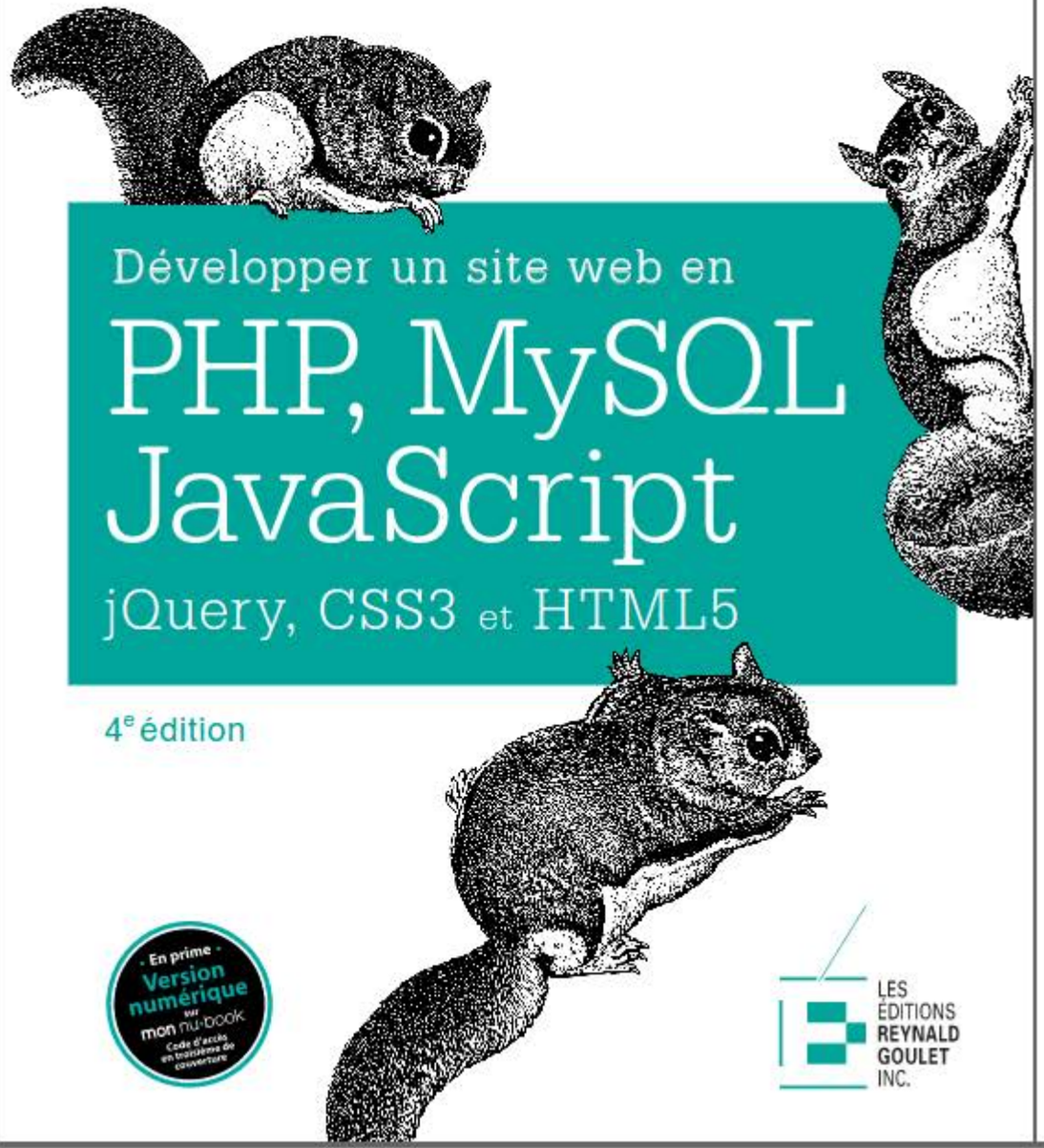
O'REILLY®

Robin Nixon

Codes sources en ligne
Cours et exercices
avec Corrigé



Développer un site web en

PHP, MySQL JavaScript

jQuery, CSS3 et HTML5

4e édition

En prime
Version numérique
sur
mon mu·book
Code d'accès en troisième de couverture

LES ÉDITIONS REYNALD GOULET INC.

4e édition

Développer un site web en

PHP, MySQL, JavaScript

jQuery, CSS3 et HTML5

Robin Nixon

Traduction et adaptation
William Piette

O'REILLY®

LES ÉDITIONS REYNALD GOULET INC.

Développer un site web en PHP, MySQL et JavaScript, avec jQuery, CSS3 et HTML5

4e édition

Traduction et adaptation : William Piette
Révision technique : Martin Villeneuve

Couverture et infographie : Ayotte Graph

Traduction en langue française de *Learning PHP, MySQL & JavaScript, 4th Edition*, ISBN 978-1-4919-1866-1,

Avec la participation du gouvernement du Canada | Canada

Avec la participation du gouvernement du Québec | Québec

Programme de crédit d'impôt pour l'édition de livres – Gestion SODEC

Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada

Imprimé au Canada
18 17 16 15 5 4 3 2 1

ISBN 978-2-89377-547-0

NOTE IMPORTANTE AU LECTEUR

L'éditeur n'offre aucune garantie de quelque forme que ce soit à l'égard des produits décrits dans ce livre et n'a effectué aucune analyse indépendante en rapport avec les informations et les produits qui y sont décrits. L'éditeur n'assume aucune responsabilité quant à l'obtention ou l'inclusion d'informations autres que celles fournies par le fabricant.

L'éditeur n'offre aucune garantie de quelque forme que ce soit quant à la justesse et à la qualité marchande du matériel fourni et se déclare exempt de toute responsabilité à cet égard. L'éditeur ne pourra en aucun cas être passible de poursuites en dommages et intérêts exemplaires résultant d'une utilisation erronée ou de la confiance portée par le lecteur sur le matériel contenu dans cet ouvrage, en tout ou en partie.

Avant-propos

La combinaison de PHP et MySQL représente l'approche la plus pratique pour la conception de sites web dynamiques orientés vers les bases de données. Elle tient parfaitement la route par rapport à des environnements de développement intégrés bien plus difficiles à apprendre, comme *Ruby on Rails*. En raison de ses racines établies en code source ouvert (au contraire du Microsoft .NET Framework concurrent), sa mise en œuvre est gratuite et constitue donc un choix très apprécié pour le développement web.

Toute personne désireuse de développer sur une plateforme Unix/Linux, ou même Windows/Apache, se doit de maîtriser ces technologies. Combinées aux technologies partenaires JavaScript, jQuery, CSS et HTML5, elles vous permettent de créer des sites web du niveau des standards de l'industrie, comme Facebook, Twitter et Gmail.

Public visé

Ce livre est destiné aux gens qui souhaitent apprendre à créer des sites web efficaces et dynamiques. Cet ouvrage se destine aussi aux webmestres et aux concepteurs graphiques qui créent déjà des sites web statiques et souhaitent porter leurs compétences à un niveau supérieur, ainsi qu'aux étudiants en conception web et en informatique, aux jeunes diplômés et aux autodidactes.

En pratique, toute personne intéressée à apprendre les fondements qui sous-tendent les technologies du Web 2.0, connues sous le nom d'Ajax, peuvent obtenir ici les bases approfondies de toutes les technologies fondamentales de PHP, MySQL, JavaScript, CSS et HTML5, ainsi que l'essentiel de la bibliothèque jQuery.

Hypothèses de travail

Le contenu de ce livre suppose que vous disposiez d'une compréhension élémentaire du HTML et que vous soyez en mesure de réaliser un site web statique tout simple, mais suppose que vous n'ayez aucune connaissance préalable de PHP, MySQL, JavaScript, CSS ni HTML5. Si c'est néanmoins le cas, bien entendu, vous pourrez avancer plus rapidement dans le livre.

Structure du livre

Les chapitres de ce livre ont été rédigés dans un ordre précis, avec une toute première introduction à toutes les technologies fondamentales que nous allons aborder, puis un guide de leur installation sur un serveur de développement afin que vous puissiez vous livrer à l'essai des exemples proposés.

La première section vise à vous familiariser avec les bases du langage de programmation PHP, avec un aperçu de la syntaxe, des tableaux, des fonctions et de la programmation orientée objet.

Ces bases du PHP en poche, vous découvrirez ensuite le système de base de données MySQL. Vous apprendrez tout, de la structure d'une base de données MySQL à la génération de requêtes de données complexes.

La phase suivante consiste à combiner PHP et MySQL pour débuter la création de vos propres pages web dynamiques par l'intégration de formulaires et d'autres fonctions HTML. Vous plongerez ensuite dans les aspects plus pratiques et intéressants du développement en PHP et MySQL par l'entremise de l'étude de toute une variété de fonctions utiles et de la gestion des témoins de connexion et des sessions, le tout en garantissant un haut niveau de sécurité.

Les quelques chapitres suivants vous inviteront à découvrir les bases de JavaScript, des simples fonctions et de la gestion d'événements à l'accès au modèle objet de document DOM (*Document object model*), en passant par la validation dans le navigateur et la gestion des erreurs. Ce sera là l'occasion de découvrir l'utilisation de la bibliothèque jQuery en JavaScript, fort appréciée.

Ce n'est qu'au terme de la compréhension de ces trois technologies fondamentales que vous pourrez ensuite découvrir comment effectuer des appels Ajax en coulisses et transformer vos sites web en environnements hautement dynamiques.

Deux chapitres suivent, où vous apprendrez à exploiter CSS pour concevoir l'aspect de vos pages web et leur apporter du style. La section suivante aborde les nouvelles fonctionnalités intégrées dans HTML5, comme la géolocalisation, l'audio, la vidéo et les canevas. Au terme de cela, vous assemblerez tout ce que vous aurez appris et constituerez un ensemble complet de programmes pour édifier un site web de réseau social complètement opérationnel.

Tout au long de ce parcours, vous découvrirez toutes sortes de conseils, de bonnes pratiques et d'astuces de programmation qui vous permettront de résoudre des erreurs difficiles à repérer. Si vous souhaitez aller plus loin dans votre découverte, des liens parsèment le livre vers des sites web contenant des détails plus approfondis sur les sujets abordés.

Anglais ou français?

Si vous examinez les offres d'emplois proposées par les institutions et les sites spécialisés, vous voyez quasi systématiquement que l'anglais courant est souhaité, sinon exigé. C'est dans cette optique que ce livre vous encourage à *aussi* apprendre à maîtriser les concepts en anglais. S'il est plus confortable pour vous de lire un texte en français, le fait de tout écrire en français ne vous rendra pas service. Les amateurs du monde du code source ouvert connaissent bien l'importance de l'anglais dans leurs collaborations au sein des communautés d'utilisateurs et de développeurs. Lorsque vous aborderez jQuery, par exemple, vous verrez que tout y est en anglais. Par conséquent, l'approche proposée dans ce livre consiste à conserver l'anglais dans les programmes, mais de commenter autant que possible en français les lignes de code. Elle se situe donc à mi-chemin entre le tout français, avec le risque que, lorsque vous voudrez aller plus loin, vous soyez perdu, et le tout anglais, qui vous ralentirait dans votre étude. Vous verrez que vous retrouverez très rapidement vos repères en connaissant le sens des fonctions et des commandes qui appartiennent à ces environnements.

Conventions en usage dans ce livre

Ce livre utilise les conventions typographiques suivantes :

Noms des options
Indiquent des intitulés de menus, options et boutons.

Termes importants
Soulignent de nouveaux termes et concepts.

Noms de fichiers
Indiquent des noms de fichiers, extensions de fichiers, chemins de fichiers, dossiers et utilitaires Unix.

Adresses URL
Indiquent des URL, des adresses de courriel.

`Caractères de largeur fixe`
Désignent des options de ligne de commande, des variables et d'autres éléments de code, des balises HTML, des macros et le contenu de fichiers.

`Caractères de largeur fixe`
Désignent des options de ligne de commande, des variables et d'autres éléments de code, des balises HTML, des macros et le contenu de fichiers insérés dans le texte.

`Caractères de largeur fixe, en gras`
Présentent les résultats des programmes ou des sections de code mises en évidence et décrites dans le texte.

`Caractères de largeur fixe, en italiques`
Désignent du texte à remplacer par des valeurs fournies par l'utilisateur.

Ce logo identifie une astuce, une suggestion ou une remarque d'ordre général.

Ce logo identifie un avertissement ou une remarque importante.

Utilisation des exemples de code

Ce livre vise votre réussite. En général, les exemples de code de ce livre sont proposés en toute liberté et vous pouvez les utiliser dans vos programmes et documentations. Il n'est pas nécessaire de communiquer avec nous pour obtenir de permission, sauf si vous reproduisez une portion significative du code des programmes. Ainsi, l'écriture d'un programme qui reprend plusieurs extraits des codes de ce livre ne nécessite aucune autorisation. En revanche, la diffusion d'un CD-ROM des exemples des livres de l'éditeur exige une autorisation. Le fait de répondre à une question en citant ce livre

et des exemples de code ne nécessite aucune permission. Par contre, l'insertion d'une part importante d'exemples de code extraits de ce livre dans votre documentation de produit exige une autorisation auprès de l'éditeur. Ce livre possède un site web d'accompagnement, sur *www.goulet.ca/nixon*, où vous pouvez télécharger tous les exemples de code de ce livre dans un seul fichier.

Nous apprécierions mais sans la réclamer, une citation pour référence de paternité. Une référence d'auteur reprend habituellement le titre, l'auteur, l'éditeur et le numéro ISBN. Par exemple, « *Développer un site web en PHP, MySQL et JavaScript, 4e édition* de Robin Nixon (Éditions Reynald Goulet). Droits d'auteur 2015 Robin Nixon, ISBN 978-2-89377-547-0 ».

À propos de l'auteur

Robin Nixon possède plus de 30 ans d'expérience d'écriture de logiciels ainsi que dans le développement de sites et d'applications web. Il a écrit plus de 500 articles à propos des ordinateurs et de la technologie, publié dans des magazines et près de 30 livres, dont beaucoup ont été traduits dans d'autres langues. Il est aussi un prolifique formateur dans des cours en vidéo sur l'internet.

Au-delà de l'informatique, ses points d'intérêts vont à la psychologie et la motivation (à propos desquelles il écrit aussi), la recherche en intelligence artificielle, de nombreux genres de musique qu'il aime non seulement écouter mais également jouer, les jeux de plateaux qu'il aime créer et dont il joue volontiers, l'étude de la philosophie et la culture, sans compter les bons plats et les boissons pour les accompagner.

Robin vit sur la côte sud-est de l'Angleterre, où il écrit à temps plein, avec ses cinq enfants et sa femme, Julie, une infirmière expérimentée et professeur d'université. Ensemble, ils stimulent également trois enfants handicapés. Vous pouvez suivre les messages (sporadiques) de Robin sur son site web, *robinnixon.com*.

Version numérique gratuite

À l'achat de ce livre en format papier, vous obtenez ***en prime*** un accès à la version numérique de ce livre durant 1 an pour les **appareils mobiles iOS**, **Android** ou sur **ordinateur Mac OS** ou **Windows**.

Dans cette version numérique Mon Nu-book, vous pouvez apporter des annotations, insérer des liens, effectuer une recherche contextuelle et plus encore.

Pour télécharger la version numérique, grattez la pastille en 3e de couverture, puis suivez les instructions.

Pour nous contacter

N'hésitez pas à envoyer vos commentaires et vos questions à propos de ce livre à l'éditeur.

Une page web est consacrée à ce livre, où nous reprenons des erratas, des exemples et toutes informations utiles. Pour y accéder, allez sur *www.goulet.ca/nixon*.

Table des matières

CHAPITRE 1

Introduction au contenu web dynamique

Le monde du web, l'internet ou la « Toile » est un réseau en constante évolution. Il a déjà franchi les premières barrières bien au-delà de sa conception au début des années 1990, lorsqu'il a vu le jour pour résoudre un problème précis. Les expériences à la pointe du progrès au CERN, le Laboratoire européen de physique des particules (aujourd'hui connu comme l'opérateur du Grand collisionneur d'hadrons), produisaient des quantités incroyables de données, si bien que la diffusion de ces données s'est vite avérée fastidieuse, parmi les scientifiques participants disséminés à travers le monde.

À l'époque, l'internet était déjà en place, avec plusieurs centaines de milliers d'ordinateurs connectés. Tim Berners Lee (qui travaillait au CERN) a conçu une méthode pour naviguer parmi ceux-ci grâce à un environnement bâti autour des liens hypertextes, ou hyperliens, qui s'est fait connaitre sous le nom d'*Hypertext transfer protocol*, ou HTTP. Il a également créé un langage de balisage appelé *Hypertext markup language*, ou HTML. Pour les assembler, il a créé les premiers navigateur et serveur web, outils dont nous lui sommes aujourd'hui encore reconnaissants.

Mais revenons un instant à l'époque, où le concept était révolutionnaire. Les seuls éléments d'interconnexion connus jusque-là nécessitaient de se connecter par modem à des systèmes de babillard électronique (ou BBS, *bulletin board systems*), hébergés chacun par un seul ordinateur, auxquels il fallait s'abonner pour communiquer et échanger des données avec d'autres utilisateurs de ces services. Par conséquent, il fallait être membre de nombreux babillards pour communiquer électroniquement et efficacement avec des confrères ou des amis.

Tim Berners Lee a changé la donne du tout au tout en une seule fois et, au milieu des années 1990, trois navigateurs principaux se concurrençaient déjà pour s'attirer les grâces de quelque cinq millions d'utilisateurs. Or, il est vite apparu évident que quelque chose manquait. Bien sûr, des pages de texte et de graphismes avec des liens hypertextes vers d'autres pages représentaient un concept génial, mais les résultats ne reflétaient pas le potentiel instantané des ordinateurs et de l'internet à rencontrer les besoins spécifiques de chaque utilisateur, avec un contenu dynamique, sur mesure. L'utilisation du web se résumait alors à une expérience fade et élémentaire, même si nous disposions déjà de textes défilants et d'images GIF animées.

Les paniers d'achat, les moteurs de recherche et les réseaux sociaux ont depuis clairement modifié la manière dont nous utilisons le web. Dans ce chapitre, nous examinons brièvement les différents composants qui forment le web et les logiciels qui permettent d'en tirer une expérience riche et dynamique.

Nous allons devoir commencer à explorer des acronymes qui, désolé, sont quasiment tous en anglais. J'essaie de les expliquer clairement, mais ne vous inquiétez pas trop de ce qu'ils signifient à première vue, car ils vous apparaitront plus clairs à mesure que vous avancerez dans votre lecture.

HTTP et HTML : les bases selon Tim Berners Lee

La norme de communication HTTP régit les requêtes et les réponses qui prennent place entre le navigateur, qui s'exécute sur l'ordinateur de l'utilisateur final, et le serveur web. La tâche du serveur consiste à accepter une requête du client et à tenter de lui répondre d'une manière intelligible, généralement par l'envoi de la page web demandée. Le serveur fournit ce service, d'où son nom de *serveur*. Le *client* constitue la contrepartie naturelle du serveur et ce terme désigne tant le navigateur web que l'ordinateur sur lequel il fonctionne.

Entre le client et le serveur existent plusieurs autres appareils, comme les routeurs, les serveurs mandataires, les passerelles et ainsi de suite. Ils endossent des rôles différents pour garantir le transfert correct des requêtes et des réponses entre le client et le serveur. Normalement, ils utilisent l'internet pour échanger ces informations.

Un serveur web gère généralement de nombreuses connexions simultanées et, quand il ne communique pas avec un client, il passe l'essentiel de son temps à écouter si des connexions entrantes lui parviennent. Dès que l'une lui parvient, le serveur envoie une réponse pour confirmer la réception de la demande.

Procédure de requête-réponse

Au niveau le plus fondamental, le processus de requête-réponse est assuré par un navigateur web qui demande à un serveur web de lui envoyer une page web, et le serveur lui envoie cette page en retour. Le navigateur se charge ensuite d'afficher correctement ladite page (figure 1-1).

Les étapes de la séquence de requête (ou demande) et de réponse se déroulent comme suit :

1. Vous entrez *http://serveur.com* dans la barre d'adresse du navigateur.
2. Le navigateur recherche l'adresse IP correspondant à *serveur.com*.
3. Le navigateur envoie une demande de la page d'accueil de *serveur.com*.
4. La requête voyage sur l'internet et parvient au serveur web *serveur.com*.

5. Le serveur web qui reçoit la requête recherche la page web correspondante sur son disque dur.
6. Le serveur trouve la page web et la renvoie au navigateur.
7. Le navigateur reçoit et affiche la page web sur l'écran.

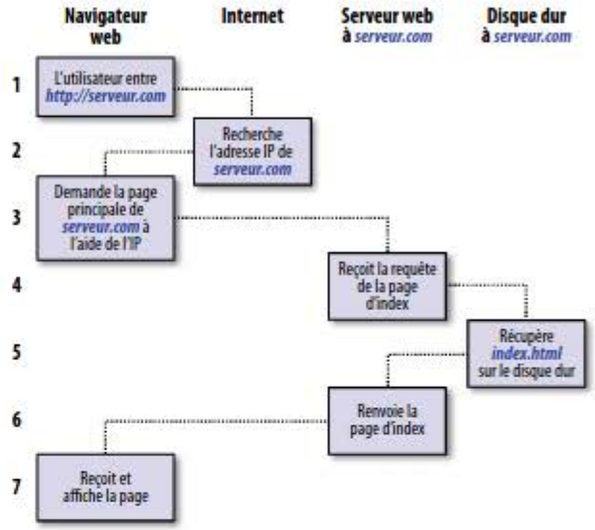


Figure 1-1. La séquence de base d'une requête-réponse entre client et serveur

Dans le cas d'une page web classique, ce processus s'établit pour chacun des objets présents dans la page, que ceux-ci soient une image, une vidéo intégrée ou un fichier Flash, et même un modèle en CSS.

Remarquez qu'à la deuxième étape, le navigateur doit rechercher l'adresse IP correspondante à *serveur.com*. Toute machine connectée à l'internet possède une adresse IP, y compris votre ordinateur. Or, nous accédons généralement aux serveurs web par leurs noms, comme *google.com*. Vous savez très probablement que le navigateur consulte un service supplémentaire de l'internet, dit de nom de domaine (DNS, *Domain name service*) pour connaitre l'adresse IP associée à un nom de domaine, et ce n'est qu'ensuite qu'il peut communiquer avec le serveur.

Dans le cas de pages web dynamiques, le processus est un peu plus complexe, puisqu'il fait intervenir PHP et MySQL à la fois dans la communication (figure 1-2).

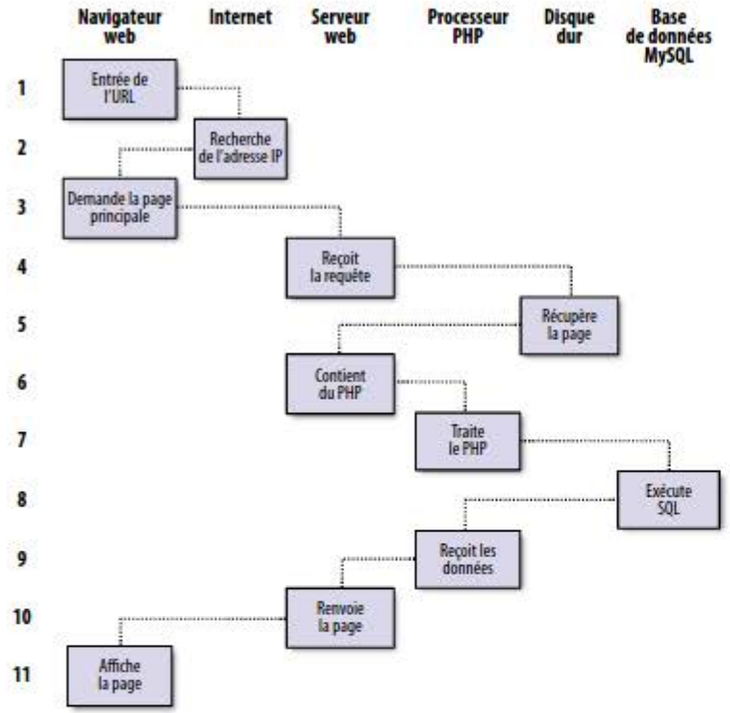


Figure 1-2. La séquence de requête-réponse dynamique entre client et serveur

1. Vous entrez *http://serveur.com* dans la barre d'adresse de votre navigateur.
2. Celui-ci recherche l'adresse IP correspondant à *serveur.com*.
3. Votre navigateur envoie une requête à cette adresse pour obtenir la page d'accueil du serveur.
4. La requête franchit l'internet et parvient au serveur web *serveur.com*.
5. Le serveur web qui reçoit la requête recherche la page web correspondante sur son disque dur
6. La page étant à présent en mémoire, le serveur web détecte que le fichier récupéré contient du script PHP et transfère la page à l'interpréteur PHP.
7. L'interpréteur PHP exécute le code PHP.

8. Certaines lignes de PHP contiennent des instructions destinées à MySQL, que l'interpréteur PHP transmet à présent au moteur de base de données MySQL.
9. Le moteur de base de données MySQL renvoie les résultats de ces instructions à l'interpréteur PHP.
10. L'interpréteur PHP renvoie les résultats du code PHP exécuté, avec en plus les données renvoyées par la base de données MySQL, au serveur web.
11. Le serveur web renvoie la page complète au client demandeur, qui l'affiche.

S'il est utile de prendre conscience de tout ce processus pour bien comprendre comment tous ces éléments travaillent de concert, en pratique, vous n'avez pas à vous inquiéter de ces détails, car ils interviennent tous de manière automatique et transparente pour vous.

Les pages HTML renvoyées au navigateur dans ces deux exemples peuvent aussi contenir du JavaScript, interprété alors localement au niveau du client et qui peut encore amorcer une autre requête, de la même manière que le font les objets intégrés, tels que les images.

Les avantages de PHP, MySQL, JavaScript, CSS et HTML5

Tout au début de ce chapitre, j'ai présenté le monde, tel qu'il était à l'époque du Web 1.0, mais il n'a pas fallu longtemps avant que le Web 1.1 surgisse, avec le développement d'améliorations apportées aux navigateurs, telles que Java, JavaScript, JScript (une variante proche de JavaScript conçue par Microsoft) et les ActiveX. Du côté du serveur, des progrès sont aussi apparus, avec les CGI (*Common gateway interface*), qui utilisent des langages de script comme Perl (une alternative au langage PHP) et les *scripts du côté serveur*, pour insérer le contenu d'un fichier (ou les résultats d'un appel système) dans un autre, de façon dynamique.

Lorsque la poussière de la tourmente de toutes ces évolutions s'est reposée, trois technologies principales se sont maintenues au-dessus des autres. Bien que Perl soit encore très apprécié en tant que langage de script et qu'il ait encore de nombreux adeptes, la simplicité de PHP et ses liens intégrés au programme de base de données MySQL lui ont valu de doubler le nombre de ses fervents utilisateurs. Ensuite, JavaScript s'est rapidement imposé comme une des composantes essentielles de l'équation qui consiste à manipuler des feuilles de style en cascade (CSS, *Cascading style sheets*) et du HTML, pour prendre à son compte la lourde charge de gérer du côté client les processus liés à Ajax. Grâce à Ajax, les pages web exécutent des tâches de gestion de données et d'envoi de requêtes aux serveurs web à l'arrière-plan, sans que l'utilisateur se rende compte de quoi que ce soit.

Il ne fait aucun doute que la symbiose naturelle entre PHP et MySQL les a propulsés à l'avant, mais une question se pose : qu'est-ce qui a attiré les développeurs vers eux, de prime abord ? La réponse est assez simple : l'aisance de les utiliser pour créer rapidement des éléments dynamiques dans des sites web. MySQL est un système de base de

données rapide et puissant, mais facile à utiliser, qui offre exactement tout ce dont un site web a besoin pour retrouver des données et les distribuer à un navigateur. Lorsque PHP vient s'allier à MySQL pour stocker, rechercher et manipuler ces données, vous avez tous les pans fondamentaux de l'édifice nécessaire pour développer des sites de réseaux sociaux et aborder le WEB 2.0.

Et, lorsque vous intégrez du JavaScript et des CSS dans la recette, vous obtenez les ingrédients pour bâtir des sites web très dynamiques et interactifs.

Utilisation de PHP

Avec PHP, il ne s'agit ni plus ni moins que d'intégrer des activités dynamiques dans des pages web. Lorsque vous attribuez à des pages l'extension de nom de fichier *.php*, elles accèdent automatiquement au langage de script. Selon le point de vue du développeur, tout ce que vous avez à faire se résume à écrire du code tel que le suivant :

```
<?php
  echo " Nous sommes aujourd'hui " . date("l") . ". ";
?>
Voici les dernières nouvelles.
```

La balise `<?php` d'entrée de jeu indique au serveur web de permettre au programme PHP d'interpréter tout le code qui suit, jusqu'à la balise `?>`. En dehors de cette structure, tout est envoyé directement sous forme de HTML. Ainsi, le texte `Voici les dernières nouvelles` est purement et simplement envoyé au navigateur. Au sein des balises PHP, la fonction intégrée `date` affiche le jour de la semaine selon la date système du serveur.

Le résultat de ce code apparait comme suit :

```
Nous sommes aujourd'hui Wednesday. Voici les dernières nouvelles.
```

Eh oui, « Wednesday » est en anglais. Nous verrons plus loin comment gérer ce détail.

De nombreuses possibilités existent pour mettre en forme et restituer des informations, que j'expliquerai dans les chapitres sur PHP. L'essentiel est de comprendre que PHP offre aux développeurs web un langage de script qui, même s'il n'est pas aussi rapide qu'un langage compilé, comme C ou tout autre langage comparable, il est extrêmement rapide et s'intègre parfaitement au balisage HTML.

Si vous avez l'intention de programmer les exemples en PHP de ce livre en même temps que je les décris, rappelez-vous d'ajouter `<?php` au début du code et `?>` après, pour que l'interpréteur PHP les traite. Pour vous faciliter cette tâche, vous pouvez vous créer un fichier nommé *exemple.php*, avec ces balises déjà préparées.

Grâce à PHP, vous disposez d'un contrôle sans limite de votre serveur web. Que vous vouliez modifier du HTML à la volée, traiter un paiement par carte de crédit, ajouter des informations sur un utilisateur dans une base de données ou récupérer des informations d'un site tiers, vous pouvez le faire dans les mêmes fichiers PHP qui contiennent le code HTML des pages.

Utilisation de MySQL

Bien entendu, cela n'a d'intérêt d'être en mesure de modifier de façon dynamique la sortie en HTML que si vous disposez aussi d'un moyen de suivre l'évolution des utilisateurs lorsqu'ils naviguent sur votre site web. Aux premiers jours du web, nombre de sites empruntaient de simples fichiers texte « à plat » pour stocker des données telles que les noms d'utilisateur et les mots de passe. Mais cette approche pouvait provoquer des problèmes, si ces fichiers n'étaient pas correctement protégés contre les corruptions causées par de nombreux accès simultanés. De plus, un fichier à plat peut grossir énormément, jusqu'au point de devenir ingérable, sans compter les difficultés que supposent les tentatives de fusionner des fichiers et d'effectuer des recherches complexes en un délai raisonnable dans un tel fichier.

C'est là que les bases de données relationnelles avec leurs requêtes structurées deviennent essentielles. Or, comme l'utilisation de MySQL est gratuite et qu'il est installé sur un grand nombre de serveurs web de l'internet, il constitue un superbe candidat pour ce rôle. Il s'agit d'un système de gestion de bases de données robuste et exceptionnellement rapide, qui se pilote à l'aide de commandes fort proches de l'anglais de tous les jours.

Le niveau le plus élevé de structure de MySQL est formé d'une base de données parmi laquelle une ou plusieurs tables existent, qui contiennent les données. Si, par exemple, vous travaillez sur une table appelée `utilisateurs`, dans laquelle vous avez créé des colonnes pour contenir le `nom`, le `prénom` et l'adresse de `courriel`, et que vous voulez ajouter un nouvel utilisateur, alors une commande dans le genre de la suivante permet de le faire :

```
INSERT INTO utilisateurs VALUES('Martin', 'Julie', 'jmartin@monsite.com');
```

Il est évident que vous aurez au préalable entré quelques autres commandes pour créer la base de données et la table, puis défini correctement les champs. Mais cette commande `INSERT` montre combien il est simple d'ajouter de nouvelles données à une base de données. Cette commande est un exemple du langage de requête structuré, ou SQL (*Structured query language*), conçu au début des années 1970 avec quelques réminiscences d'un des plus anciens langages de programmation, COBOL. SQL s'adapte cependant extrêmement bien aux requêtes sur des bases de données, ce qui explique qu'il soit encore en usage actuellement.

La recherche de données s'avère également très aisée. Si vous connaissez par exemple l'adresse de courriel d'un utilisateur et que vous souhaitez connaitre le nom et le prénom de son propriétaire, vous pouvez entrer une requête MySQL telle que la suivante :

```
SELECT nom, prénom FROM utilisateurs WHERE courriel='jmartin@monsite.com';
```

MySQL renvoie alors `Martin, Julie` et éventuellement d'autres noms et prénoms qui correspondent aussi à cette adresse de courrier électronique dans la base de données.

MySQL permet de réaliser bien plus de choses intéressantes que simplement des `INSERT` et des `SELECT`. Ainsi, vous pouvez par exemple joindre plusieurs tables en fonction de certains critères, demander les résultats selon différents ordres de tri, restreindre des

requêtes à une portion de colonne correspondant au début d'une chaine de caractères ou n'extraire que le nième résultat, et bien plus encore.

Grâce à PHP, vous envoyez ces appels directement à MySQL, sans devoir exécuter vous-même ce programme ni entrer les commandes dans l'interface en ligne de commande (appelée aussi « Terminal », « console » ou « Invite de commande »). Ceci implique que vous récupériez les résultats dans des tableaux en mémoire pour les traiter et y pratiquer de multiples recherches, dépendant chacune des précédentes, pour forer de proche en proche et atteindre la donnée dont vous avez besoin.

Pour plus de puissance, comme vous le constaterez plus loin, des fonctions supplémentaires existent dans MySQL, que vous pouvez appeler pour des opérations communes à une vitesse inégalée.

Utiliser JavaScript

La plus ancienne de ces trois technologies fondamentales exposées dans ce livre, JavaScript, a été créée pour permettre un accès par script à tous les éléments d'un document HTML. En d'autres termes, il offre une possibilité d'interaction dynamique de l'utilisateur, comme la vérification de validité d'une adresse courriel dans des formulaires d'entrée d'informations, et afficher des invites, c'est-à-dire des messages de demande de confirmation, du genre « Voulez-vous vraiment faire cela ? ». Il ne faut toutefois pas compter sur lui au niveau de la sécurité, qui doit toujours être gérée au niveau du serveur web.

Combiné avec les CSS (voir la section suivante), JavaScript représente la puissance derrière les pages web dynamiques qui changent devant vos yeux, sans imposer de retour au serveur de toute la page ni de renvoi de celui-ci.

JavaScript peut toutefois s'avérer un peu compliqué à utiliser, du fait de différences majeures dans la manière dont les concepteurs de navigateurs ont choisi de le mettre en œuvre. Ceci est essentiellement dû au fait que certains éditeurs de navigateurs ont souhaité apporter des fonctionnalités supplémentaires à leur navigateur, au détriment de la compatibilité avec ceux de leurs concurrents.

Heureusement, les développeurs ont presque tous retrouvé leur lucidité et ont compris la nécessité d'une complète compatibilité entre leurs produits, de manière à ne pas devoir multiplier les ajouts de code pour gérer les exceptions. Par chance, des solutions existent pour gérer les problèmes d'incompatibilité et, par la suite dans ce livre, nous examineront les bibliothèques et les techniques qui permettent, par leur présence, d'éluder ces différences.

Pour l'heure, voyons comment utiliser du code JavaScript accepté par tous les navigateurs :

```
<script type="text/javascript">
  document.write("Nous sommes le " + Date() );
</script>
```


Cet extrait de code indique au navigateur d'interpréter tout ce qui est présent entre les balises `script` comme du JavaScript, ce que le navigateur exécute ensuite en écrivant le texte `Nous sommes le` dans le document courant, suivi de la date, à l'aide de la fonction `Date` de JavaScript. Le résultat donne quelque chose du genre de :

```
Nous sommes le Sun Jan 01 2017 01:23:45
```

À moins d'avoir besoin de préciser une version spécifique de JavaScript, d'une manière générale, vous pouvez éluder le `type="text/javascript"` et simplement utiliser `<script>` pour débuter l'interprétation du JavaScript.

Comme évoqué précédemment, JavaScript a été développé à l'origine pour offrir un contrôle dynamique des différents éléments d'un document HTML, et c'est encore là aujourd'hui son utilisation majeure. Toutefois, JavaScript est de plus en plus exploité dans le cadre d'Ajax. Ce terme désigne le traitement qui consiste à accéder au serveur web à l'arrière-plan. Il signifiait à l'origine *Asynchronous JavaScript and XML*, mais cette expression a déjà pris un peu d'âge par rapport à la réalité actuelle.

Ajax est le principal processus derrière ce que l'on désigne aujourd'hui de Web 2.0 (terme popularisé par Tim O'Reilly, fondateur et PDG de la maison d'édition O'Reilly), où les pages web commencent à ressembler de plus en plus à des programmes autonomes classiques, parce qu'il n'est plus nécessaire de les recharger chaque fois en totalité. À la place, un appel Ajax rapide peut récupérer et mettre à jour un seul élément d'interface sur une page web, comme pour modifier votre photo sur un site de réseau social ou remplacer un bouton sur lequel vous cliquez, avec la réponse à une question. Le chapitre 17 étudie ce sujet en profondeur.

Ensuite, au chapitre 21, nous examinons copieusement le cadre d'applications jQuery, que vous pouvez utiliser pour éviter de réinventer la roue, lorsque vous devez trouver du code rapide, multinavigateur, pour manipuler vos pages web. Bien entendu, d'autres environnements existent au-delà de celui-ci, mais jQuery est largement le plus plébiscité et, du fait de son entretien permanent, est extrêmement fiable et constitue un élément de choix dans la trousse à outils de nombreux développeurs web expérimentés.

Utiliser CSS

Avec l'émergence de la norme CSS3 ces dernières années, CSS offre un niveau d'interactivité dynamique qui n'était pris en charge auparavant que par JavaScript. Ainsi, non seulement vous pouvez modifier le style de n'importe quel élément HTML pour corriger ses dimensions, couleurs, bordures, espacements et ainsi de suite, mais vous pouvez aussi ajouter des transitions et des transformations animées aux pages web à l'aide de quelques lignes de CSS.

L'utilisation de CSS peut s'avérer aussi simple que d'insérer quelques règles entre les balises `<style>` et `</style>` dans l'entête d'une page web, comme suit :

```
<style>
  p {
    text-align:justify;
    font-family:Helvetica;
  }
</style>
```

Ces règles changent l'alignement par défaut du texte de la balise `<p>`, de sorte que les paragraphes contenus dans les paragraphes sont complètement justifiés et utilisent la police Helvetica.

Comme vous le verrez au chapitre 18, il existe plusieurs façons de disposer les règles CSS et vous pouvez aussi les inclure directement entre des balises ou enregistrer un ensemble de règles dans un fichier externe pour les charger indépendamment. Cette souplesse permet non seulement d'ajuster avec précision du HTML, mais elle fournit aussi, entre autres, la possibilité de concevoir une fonctionnalité d'effet de survol de la souris pour animer les objets lorsque la souris passe au-dessus d'eux. Vous apprendrez aussi comment accéder à toutes les propriétés CSS d'un élément à partir de JavaScript, ainsi qu'en HTML.

Et enfin, voici HTML5

Même si ces ajouts aux normes du web furent très utiles, ils étaient encore insuffisants pour les développeurs plus ambitieux. Par exemple, il n'existait aucun moyen simple de manipuler des graphismes dans un navigateur web, sans faire appel à des plugiciels tels que Flash. Et la même frustration surgissait lorsqu'il fallait intégrer de l'audio et de la vidéo dans une page web. En plus, plusieurs incohérences gênantes se sont glissées dans HTML au cours de son évolution.

Donc, pour gommer tout cela, amener l'internet dans le Web 2.0 et au-delà de son itération suivante, une nouvelle norme a été créée pour le HTML afin de répondre à toutes ces lacunes : le HTML5. Son développement a débuté en 2004, lorsque le premier projet a été élaboré par la *Mozilla Foundation* et *Opera Software* (les développeurs de deux navigateurs web populaires). Mais ce n'est qu'au début de 2013 que le projet final a été soumis au W3C (*World Wide Web Consortium*), l'organisme international qui régit les standards du web.

Les neuf années nécessaires pour son développement pourraient vous laisser croire que c'est là sa spécification finale, mais ce n'est pas ainsi qu'évoluent les choses sur l'internet. Si les sites web vont et viennent à grande vitesse, les logiciels sous-jacents subissent un développement lent et soigneux. Ainsi, la recommandation stable de HTML5 n'était attendue qu'à la fin 2014. Et devinez quoi ? Les travaux évoluent déjà vers la version 5.1 et d'autres suivantes, au début de 2015. Il s'agit là d'un cycle de développement sans fin.

Cependant, si HTML5.1 est prévu pour apporter quelques améliorations pratiques (essentiellement aux canevas), le HTML5 de base est la nouvelle norme vers laquelle les développeurs web doivent désormais s'orienter et demeurera encore en place pendant

de longues années. En conséquence, apprendre tout ce que vous pouvez à son sujet vous placera en très bonne position.

HTML comporte vraiment de nombreuses nouveautés, outre quelques changements et d'autres choses qui ont disparu, mais en résumé, voici à quoi vous attendre :

Balisage
: L'apparition de nouveaux éléments tels que `<nav>` et `<footer>`, tandis que les éléments obsolètes `<font>` et `<center>` disparaissent.

De nouvelles API
: Les interfaces de programmation d'application, ou API (*application programming interface*) telles que l'élément `<canvas>` pour écrire et dessiner sur un canevas graphique, les éléments `<audio>` et `<video>`, les applications web hors ligne, les microdonnées et le stockage local.

Applications
: Apparaissent notamment deux nouvelles technologies de rendu : MathML (*Math markup language*) pour l'affichage de formules mathématiques et SVG (*Scalable vector graphics*) pour la création d'éléments graphiques en dehors du nouvel élément `<canvas>`. Cependant, comme MathML et SVG s'adressent plutôt à des spécialistes et qu'ils contiennent tant de possibilités qu'ils nécessiteraient un livre à part entière pour décrire chacun d'eux, je ne les étudie pas ici.

À partir du chapitre 22, nous aborderons ces éléments et bien d'autres encore.

Un des détails que j'aime à propos des spécifications de HTML5, c'est que la syntaxe du XHTML n'est plus obligatoire pour les éléments auto-fermants. Par le passé, il était possible d'afficher un retour à la ligne à l'aide de l'élément `<br>`. Par la suite, pour assurer la compatibilité avec le XHTML (le remplaçant prévu de HTML qui ne s'est jamais imposé), ceci a été remplacé par `<br />`, où le caractère de fermeture / a été ajouté (car tous les éléments sont supposés contenir une balise de fermeture contenant ce caractère). Maintenant que les choses ont bien décanté et que nous pouvons utiliser indifféremment les deux versions, pour des raisons de brièveté et pour économiser les frappes au clavier, j'ai adopté dans ce livre l'écriture initiale, soit `<br>`, `<hr>` et ainsi de suite.

Le serveur web Apache

Outre PHP, MySQL, JavaScript, CSS et HTML5, un sixième acteur intervient dans le web dynamique : le serveur web. Dans le cadre de cet ouvrage, il signifie le serveur web Apache. Nous avons vaguement évoqué le rôle du serveur web au cours des échanges entre le serveur HTTP et le client, mais ce serveur fait bien plus que cela en coulisses.

Ainsi, Apache ne dessert pas que des fichiers HTML ; il gère une vaste gamme de fichiers : des images, des fichiers Flash jusqu'aux fichiers audio MP3, des flux RSS (*Really simple syndication*) et ainsi de suite. Pour ce faire, pour chaque élément qu'il rencontre dans une page HTML, le client le demande au serveur, qui le lui envoie.

Ces fichiers ne doivent pas être nécessairement des fichiers statiques, comme les images GIF. Ils peuvent aussi naître de la génération par des programmes, tels que des

scripts PHP. C'est exactement ça : PHP peut même créer des images et d'autres fichiers pour vous, soit à la volée, soit à l'avance pour les livrer ultérieurement.

Pour cela, des modules existent normalement, soit précompilés dans Apache ou PHP, ou encore appelés au moment de l'exécution. L'un de ces modules est la bibliothèque GD (*Graphic draw*), que PHP utilise pour créer et gérer des graphismes.

En lui-même, Apache prend en charge une vaste gamme de modules. En plus du module PHP, les plus importants pour vos besoins de programmeur web sont ceux qui gèrent la sécurité. On peut citer aussi le module Rewrite, qui permet au serveur web de gérer tous types d'adresses URL et de les récrire selon ses propres exigences internes, puis il y a aussi le module Proxy, qui permet de servir des pages souvent demandées à partir d'une mémoire cache pour en faciliter le chargement sur le serveur.

Plus loin dans ce livre, vous apprendrez à exploiter certains de ces modules pour améliorer les fonctionnalités fournies par les trois technologies fondamentales.

À propos de l'open source

Le fait d'être (ou pas) en code source ouvert (*open source*) constitue le motif de la popularité de ces technologies, comme souvent évoqué, mais PHP, MySQL et Apache sont les trois outils les plus utilisés de leurs catégories. Ce que l'on peut dire finalement à leur sujet, c'est que le fait qu'ils soient en source ouverte signifie qu'ils ont été développés par la communauté, par des équipes de programmeurs qui ont rédigé les fonctionnalités dont eux-mêmes avaient besoin et qu'ils voulaient, tout en plaçant le code source original à la disposition de tous, pour lecture et modification. Ainsi, il est plus facile de découvrir les bogues et les brèches dans la sécurité avant qu'ils ne deviennent problématiques.

Il y a un autre avantage : la gratuité d'usage de tous ces programmes. Il n'est pas nécessaire de s'inquiéter pour l'achat de licences supplémentaires si vous devez faire évoluer votre site à une plus grande échelle et lui ajouter de nouveaux serveurs. Et il n'est pas indispensable de vérifier le budget disponible avant de décider de mettre ces produits à niveau vers leurs dernières versions.

Assembler le tout

La véritable beauté de PHP, MySQL, JavaScript (parfois aidé par jQuery ou tout autre environnement), CSS et HTML5 réside dans la merveilleuse manière dont ils œuvrent tous ensemble pour produire du contenu web dynamique. PHP assure la majeure partie du travail sur le serveur web, MySQL traite toutes les données et la combinaison de CSS et JavaScript assure la présentation de la page web. JavaScript peut aussi dialoguer avec le code PHP sur le serveur web, chaque fois qu'il faut mettre un élément à jour (soit sur le serveur, soit sur la page web). Et avec les puissantes fonctionnalités nouvelles de HTML5, comme le canevas, l'audio, la vidéo, la géolocalisation, vous pouvez rendre vos pages web fortement dynamiques, interactives et agrémentées de multimédia.

Sans code de programmation, résumons ce chapitre en examinant le processus de combinaison de quelques-unes de ces technologies dans une fonction Ajax d'usage quotidien que nombre de sites web exploitent : la vérification qu'un nom d'utilisateur souhaité existe déjà sur un site, lorsqu'un utilisateur s'inscrit pour obtenir un nouveau compte. Un bon exemple de ceci peut être observé sur Gmail (figure 1-3).

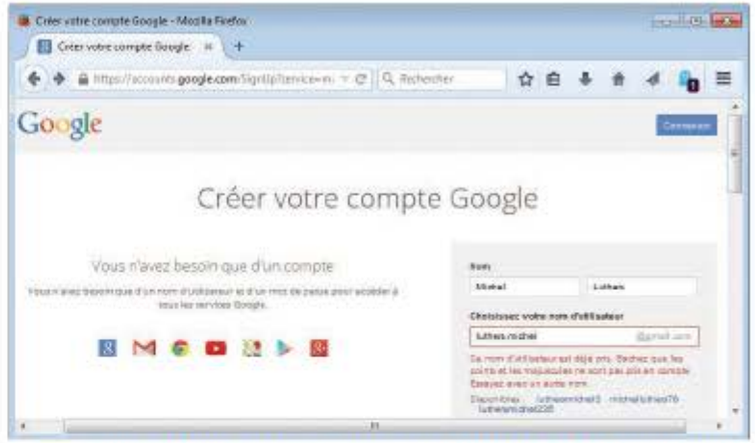


Figure 1-3. Gmail utilise Ajax pour vérifier la disponibilité des noms d'utilisateurs

Les étapes impliquées dans ce processus Ajax sont de l'ordre des suivantes :

1. Le serveur affiche le code HTML pour créer le formulaire, qui demande les détails nécessaires, comme le nom d'utilisateur, le prénom, le nom et l'adresse de courriel.
2. En même temps, le serveur associe du JavaScript au HTML pour surveiller la zone d'entrée du nom d'utilisateur et vérifier deux choses : (a) que du texte a été entré dans la zone de texte et (b) si la zone de texte n'est plus sélectionnée parce que l'utilisateur a cliqué dans une autre zone d'entrée.
3. Dès que le texte a été entré et que le champ est désélectionné, en coulisses, le code JavaScript passe le nom d'utilisateur entré à un script PHP du serveur web et attend sa réponse.
4. Le serveur recherche le nom d'utilisateur et répond en retour au JavaScript si le nom a déjà été utilisé.
5. Le JavaScript place alors une indication à côté de la zone d'entrée du nom d'utilisateur qui signifie si le nom est disponible pour l'utilisateur (éventuellement une coche verte ou, à l'inverse, une croix rouge), avec un peu de texte explicatif.

6. Si le nom d'utilisateur n'est pas disponible et que l'utilisateur soumet quand même le formulaire, JavaScript interrompt l'envoi et met de nouveau en évidence (avec peut-être un graphisme plus grand ou un message d'alerte) que l'utilisateur doit choisir un autre nom d'utilisateur.
7. Éventuellement, dans une version plus étoffée, le processus peut rechercher des alternatives disponibles en fonction du nom choisi et les proposer à l'utilisateur.

Tout ceci a lieu en coulisses et en silence, pour offrir une expérience d'utilisateur fluide et agréable. Sans Ajax, il aurait fallu soumettre la totalité du formulaire au serveur web, qui aurait renvoyé du HTML pour surligner toutes les erreurs. Cette solution fonctionne, bien entendu, mais elle est beaucoup moins propre et agréable que le traitement à la volée de champ de formulaire.

Ajax permet de réaliser bien plus de choses qu'une simple vérification et un traitement de cet ordre. Nous en découvrirons nombre d'autres dans les chapitres sur Ajax, plus loin dans ce livre.

Dans ce chapitre, vous avez eu une bonne introduction aux technologies fondamentales PHP, MySQL, JavaScript, CSS et HTML5 (ainsi qu'Apache), et vous avez vu comment elles collaborent entre elles. Au chapitre 2, nous allons aborder l'installation de votre propre serveur de développement web sur lequel vous pourrez pratiquer tout ce que vous apprendrez.

Questions

1. Énumérez au moins quatre composants nécessaires pour créer une page web totalement dynamique.
2. Que signifie *HTML* ?
3. Pourquoi *MySQL* contient-il les lettres *SQL* ? Quelle est leur signification ?
4. PHP et JavaScript sont tous deux des langages de programmation qui génèrent des résultats dynamiques pour des pages web. Quelle est leur principale différence et pourquoi les utilise -t-on ensemble ?
5. Que signifie *CSS* ?
6. Énumérez cinq nouveaux éléments introduits dans HTML5.
7. Si vous découvrez un bogue (ce qui est rare) dans un des outils en code source ouvert, comment pourriez-vous le corriger ?

Retrouvez les réponses du chapitre 1 dans l'annexe A.

CHAPITRE 2

Mettre en place le serveur de développement

Pour développer des applications web alors que vous ne disposez pas de votre propre serveur de développement, vous devrez chaque fois télécharger la moindre modification sur un serveur, hébergé quelque part sur la Toile, avant de la tester.

Même avec une connexion à très haut débit, cela peut représenter un gaspillage significatif du temps de développement. En revanche, sur un ordinateur local, le test équivaut à enregistrer une modification (ce qui se résume généralement à un simple clic sur une icône), puis à cliquer sur le bouton Actualiser du navigateur.

Un serveur de développement local offre également l'avantage d'éviter de vous préoccuper des erreurs embarrassantes et des problèmes de sécurité lorsque vous programmez et testez, tandis que sur un serveur dont l'accès est public, vous devez vous inquiéter de ce que les gens peuvent voir ou faire avec vos applications. Il est préférable de passer tout cela en revue pendant que vous êtes encore sur votre système personnel ou de petit bureau, protégé en principe par un pare-feu et d'autres mesures de protection.

Une fois que vous aurez votre propre serveur de développement, vous vous demanderez comment vous avez fait pour vous en passer auparavant. Il est très facile d'en installer un. Suivez simplement les étapes des sections suivantes, en les adaptant selon que vous utilisez un ordinateur sous Windows, OS X (Mac) ou Linux.

Dans ce chapitre, nous examinons uniquement la partie serveur de l'expérience web, telle que décrite au chapitre 1. Ensuite, pour tester les résultats de vos travaux, en particulier lorsque nous commencerons à utiliser JavaScript, CSS et HTML5 par la suite, vous devrez aussi disposer d'une instance de chacun des principaux navigateurs web sur une machine, selon vos souhaits. Autant que possible, parmi ces navigateurs principaux, vous devriez inclure Internet Explorer, Mozilla Firefox, Opera, Safari et Google Chrome. Si vous souhaitez vérifier l'aspect de vos sites sur des appareils mobiles, essayez de vous procurer (en prêt ou en location, par exemple) des téléphones et tablettes sous Apple iOS et Google Android.

Qu'est-ce que WAMP, MAMP ou LAMP ?

Les sigles WAMP, MAMP et LAMP sont les acronymes respectifs de « Windows, Apache, MySQL et PHP », « Mac, Apache, MySQL et PHP » et « Linux, Apache, MySQL et PHP ». Ces abréviations décrivent une configuration utilisable pour développer des pages web dynamiques.

WAMP, MAMP et LAMP apparaissent sous la forme d'un progiciel qui associe proprement tous les logiciels en un ensemble prédéfini et préréglé, de sorte que vous ne devez plus les installer indépendamment puis les associer ou les configurer. Cela signifie que vous ne téléchargez qu'un seul programme, répondez à quelques invites simples lors de son installation, pour obtenir un serveur web de développement prêt à l'emploi, en un minimum de temps et d'efforts.

Durant l'installation, un certain nombre de paramètres prédéfinis sont ajustés pour vous. La configuration de sécurité n'est pas aussi exigeante dans ce type d'installation qu'elle le serait sur un serveur de production, parce qu'elle est optimisée pour un usage local. En conséquence, vous ne devez jamais installer ce genre de progiciel sur un serveur de production, accessible au public.

Pour du développement et des tests de sites et d'applications web, ces configurations sont amplement suffisantes.

Si vous édifiez votre propre système de développement plutôt que de suivre la piste des WAMP, MAMP et LAMP, sachez que le téléchargement et l'intégration par vous-même des différents logiciels peuvent exiger beaucoup de temps et de nombreuses recherches pour que l'ensemble s'intègre parfaitement. Mais, si vous possédez déjà une installation de tous ces composants intégrés les uns aux autres, ils sont en mesure de fonctionner avec les exemples de ce livre.

Installer XAMPP sous Windows

Si plusieurs serveurs WAMP sont disponibles, avec des configurations légèrement différentes, parmi toutes les options en source ouverte et gratuites, la meilleure est sans doute XAMPP. Vous pouvez la télécharger à partir de *http://apachefriends.org*, comme à la figure 2-1.

Figure 2-1. Le site web de XAMPP

Il est préférable de toujours télécharger la dernière version stable (soit ici la version 5.6.3). Sur la page principale, des liens spécifiques proposent de télécharger la version spécifique pour Windows, OS X et Linux.

Depuis la publication de ce livre, certains écrans et options peuvent évoluer par rapport aux illustrations dans ce guide. Si c'est le cas, faites preuve de bon sens pour procéder d'une manière comparable aux séquences d'actions décrites.

Dès que vous avez téléchargé l'installable, exécutez-le pour obtenir le premier écran illustré à la figure 2-2. Avant d'atteindre cette fenêtre, il se peut que votre programme antivirus ou que le contrôle de compte d'utilisateur de Windows affichent quelques fenêtres d'avertissement et de confirmation, donc cliquez sur Autoriser, Oui ou OK pour poursuivre l'installation.

Figure 2-2. La fenêtre initiale d'installation de l'installeur

Cliquez sur Next, puis décochez éventuellement les cases des composants dont vous n'aurez pas besoin, comme à la figure 2-3. Le minimum indispensable pour suivre ce livre réside dans Apache, MySQL, PHP et PHPMyAdmin. Les autres fonctionnalités ne sont pas gérées dans ce livre mais vous pouvez obtenir de plus amples informations à leur propos, sur les technologies principales de XAMP à la page *https://www.apachefriends.org/faq_windows.html* en anglais).

Un clic sur Next vous conduit à l'écran de la figure 2-4, où vous devez choisir un dossier pour cette installation. Sauf si vous avez de très bonnes raisons de changer de dossier d'installation, il est préférable de conserver le dossier proposé par défaut. Dans ce livre, nous supposons que vous avez accepté ce choix prédéfini. Si le dossier que vous choisissez existe déjà et s'il n'est pas vide, vous ne pouvez pas l'utiliser.

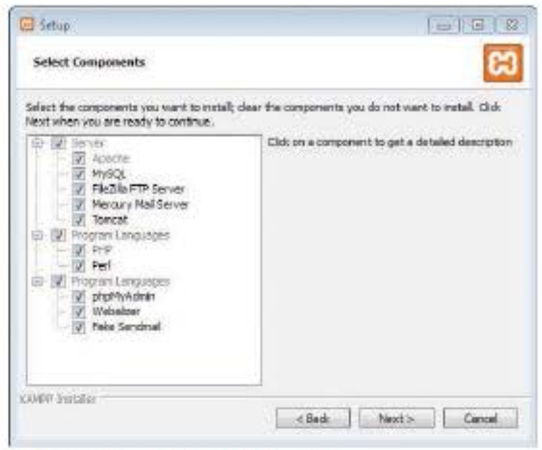


Figure 2-3. Sélectionner les composants à installer

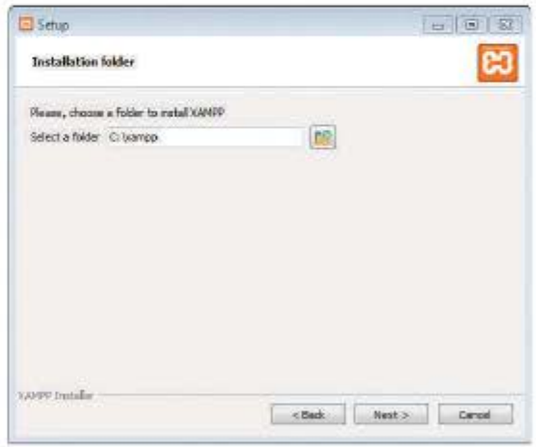


Figure 2-4. Sélectionner le dossier d'installation

Un clic sur Next conduit à l'écran de la figure 2-5, avec une case à cocher déjà cochée (que vous pouvez décocher) pour obtenir des informations supplémentaires sur des installeurs gratuits de produits associés dans une nouvelle fenêtre ou un nouvel onglet de navigateur web. Que vous décidiez ou non de recevoir ces informations, cliquez sur Next.

Figure 2-5. Des informations sont disponibles sur des produits gratuits associés

Maintenant que vous avez fourni les informations de base requises par l'installeur, vous aboutissez à l'écran de la figure 2-6. La configuration est prête à démarrer : cliquez sur Next.

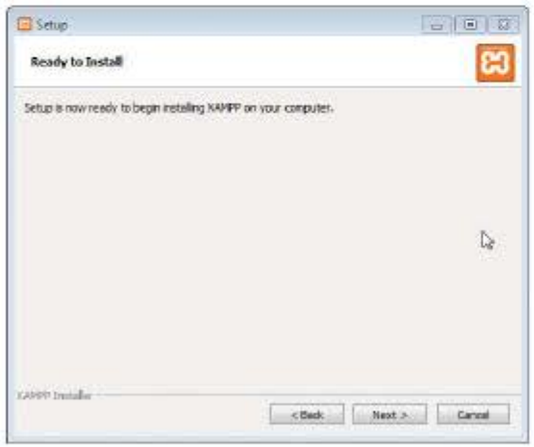


Figure 2-6. Ces premiers paramétrages effectués, XAMPP est prêt à s'installer

Lorsque vous cliquez sur Next, l'installation débute et l'écran de la figure 2-7 s'affiche durant la progression. Pendant cette installation, cliquez sur les icônes pour ouvrir une fenêtre de navigateur web avec des informations sur les produits présentés. L'ensemble du processus ne prend que quelques minutes sur la plupart des ordinateurs.

Figure 2-7. Installation en cours

Dès que l'installation est terminée, l'écran de la figure 2-8 s'affiche avec une case à cocher qui propose de démarrer maintenant le tableau de bord (*Control panel*) de XAMPP. Laissez cette case cochée et cliquez sur Finish.

À présent, vous êtes prêt à utiliser XAMPP et à le paramétrer à partir du tableau de bord, comme illustré à la figure 2-9. Le volet est affiché automatiquement si vous avez conservé la coche de la case au moment de l'achèvement de l'installation. Sinon, démarrez-le par le menu Démarrer ou sur l'écran au démarrage.

Figure 2-8. Cliquer sur Finish pour terminer l'installation

Figure 2-9. Le tableau de bord (control panel)

La première chose que je vous recommande ici est de cliquer sur Config dans le coin supérieur droit de la fenêtre pour ouvrir l'écran de la figure 2-10. En particulier, si elles ne le sont pas déjà, cochez les cases Apache et MySQL pour en garantir le démarrage automatique au démarrage de l'ordinateur. Mais vous pouvez aussi cliquer sur les boutons Start à côté d'Apache et de MySQL pour qu'ils ne démarrent que dans le cadre de votre session.

Figure 2-10. Choisir l'éditeur, les composants au démarrage (Autostart) et autres

Tant que vous êtes dans cette boite de dialogue, vous pouvez décider de modifier les réglages de ports: cliquez sur Service and Port Settings pour ouvrir la fenêtre de la figure 2-11.

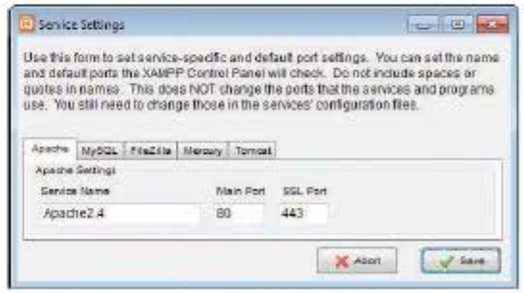


Figure 2-11. Cliquer sur Save pour terminer la configuration

Les ports prédéfinis prévus pour cette affectation sont les ports 80 pour le serveur web Apache, 443 pour SSL et 3306 pour MySQL. Si vous changez ces valeurs, n'oubliez pas de les indiquer par la suite, lorsque vous utiliserez ces services dans les exemples du livre.

Le tableau de bord est l'endroit central où vous menez l'essentiel des actions nécessaires pour gérer XAMPP, notamment l'édition et la visualisation des différents fichiers de configuration, et où vous surveillez les fichiers journaux d'accès, d'erreur et autres, le tout dans une seule interface. Ainsi, à la figure 2-12, après un clic sur le bouton Logs, à droite d'Apache, le dossier des journaux s'affiche. Remarquez que, pour obtenir le contenu illustré, il faut avoir déjà accédé à une page web de ce serveur, comme à la section suivante.

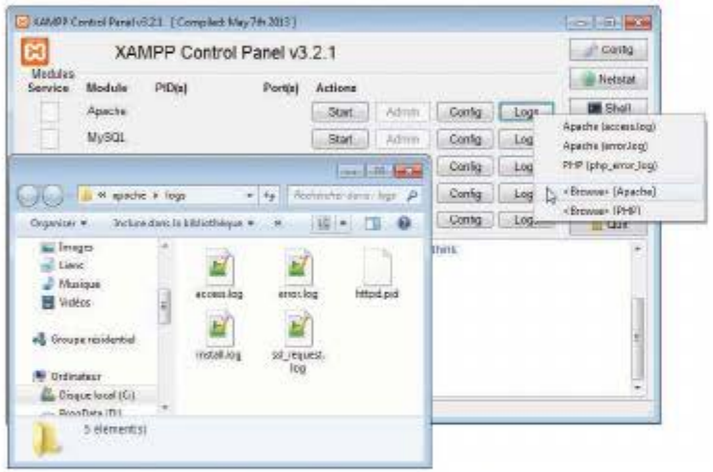


Figure 2-12. Ouverture du dossier des fichiers journaux d'Apache

Tester l'installation

À ce stade, il est important de vérifier que tout fonctionne correctement. D'abord, cliquez sur Start à droite d'Apache pour démarrer le serveur web. Essayez ensuite d'afficher la page web prédéfinie, installée dans le dossier racine des documents du serveur web (figure 2-13). Entrez une des deux adresses URL suivantes dans la barre d'adresse de votre navigateur :

```
localhost
127.0.0.1
```


Figure 2-13. Aspect de la page d'accueil de XAMPP par défaut

Le mot *localhost* dans une URL désigne l'ordinateur local, qui répond aussi à l'adresse IP 127.0.0.1, de sorte que vous pouvez utiliser indifféremment l'un ou l'autre pour accéder au document racine de votre serveur web.

Si vous avez choisi un port différent de 80 dans la configuration du tableau de bord, par exemple 8080, vous devez alors ajouter à l'URL un deux-points, suivi de cette valeur : *localhost:8080*. Vous devrez suivre cette méthode pour tester tous les exemples du livre. Ainsi, au lieu de l'URL *localhost/exemple.php*, vous devrez entrer dans la barre d'adresse du navigateur *localhost:8080/exemple.php* (ou toute autre valeur de port que vous aurez choisie précédemment).

Accéder à la racine des documents

La *racine des documents* (*document root*) est le dossier qui contient les documents web principaux d'un domaine. Ce sont ceux auxquels vous accédez automatiquement par une URL, sans indiquer de chemin, entrée dans la barre d'adresse du navigateur, comme *http://yahoo.fr* ou, dans le cas de votre serveur, *http://localhost*.

Par défaut, XAMPP utilise l'emplacement physique suivant pour ce dossier :

```
C:/xampp/htdocs
```

Pour vérifier que tout est correctement installé, créez un fichier de test, avec le classique « Bonjour tout le monde! ». Créez un petit fichier HTML à l'aide du Bloc-notes de Windows ou tout autre éditeur de texte. N'utilisez pas de logiciel qui produit du texte enrichi, tel que Microsoft Word, sauf si vous enregistrez le contenu au format Texte brut (*.txt). Entrez les quelques lignes suivantes :

```
<html>
 <head>
  <title>Un rapide test</title>
 </head>
 <body>
  Bonjour tout le monde!
 </body>
</html>
```

Dès que vous avez entré ceci, enregistrez le fichier dans le dossier racine des documents évoqué plus haut, sous le nom de fichier *test.htm*. Si vous utilisez le Bloc-notes, assurez-vous de modifier le type de fichier en Document texte (*.txt) ou Tous les fichiers (*.*). Vous pouvez aussi enregistrer le fichier avec l'extension *.html*, ce qui revient au même.

Figure 2-14. Votre première page web

Alternatives de WAMP

Lors de la mise à jour de logiciels, ces derniers se comportent parfois de manière inattendue par rapport aux attentes, et même des bogues peuvent s'inviter au passage. Donc, si vous rencontrez des difficultés que vous ne parvenez pas à résoudre, en dernier ressort, il vous reste la possibilité de prendre une autre solution disponible sur le web.

Vous pourrez toujours utiliser les exemples du livre, mais vous devrez chaque fois suivre les instructions fournies par chaque « WAMP », qui risquent de ne pas être aussi simples à suivre que celles de ce guide.

Voici une sélection des meilleurs, selon mon point de vue :

- EasyPHP : *easyphp.org*
- WAMPServer : *wampserver.com/en*
- Glossword WAMP : *glossword.biz/glosswordwamp*

Installer XAMPP sous Mac OS X

XMPP est également disponible sur OS X et vous pouvez le télécharger à partir de *http://apachefriends.org*, comme illustré précédemment à la figure 2-1.

Double-cliquez sur le fichier *.dmg* après l'avoir téléchargé, puis double-cliquez sur l'installeur et suivez la même séquence d'instructions que sous Windows. Il se peut que vous voyiez des cases à cocher pour sélectionner les fichiers du cœur du programme (*core files*), les fichiers pour développeurs (*developer files*) ou les deux.

La procédure d'installation est très semblable à celle de sous Windows, mais XAMPP s'installe sous OS X à cet emplacement : *Applications/XAMPP*.

L'installation réussie, la fenêtre de gestion, *XAMPP Manager*, s'ouvre. Pour que XAMPP puisse prendre correctement le contrôle du service web du Mac, vous devrez éventuellement arrêter tout serveur web (Apache) que le Mac exécute déjà, à l'aide de la commande suivante dans une fenêtre de Terminal :

```
sudo apachectl stop
```

Vous pouvez ensuite cliquer sur l'onglet du milieu, intitulé Manage Servers, en haut de la fenêtre et cliquez sur Start All pour démarrer les serveurs de XAMPP. Ensuite, cliquez sur l'onglet Welcome pour retourner à l'écran *Manager*, puis cliquez sur Go to Application, qui appelle la page web illustrée à la figure 2-13. L'ensemble de la suite des logiciels est à présent configuré et prêt à l'emploi.

Pour de plus amples informations sur l'installation et l'utilisation de XAMPP sur Mac, reportez-vous à *apachefriends.org/faq_osx.html* (en anglais).

À l'avenir, pour appeler le gestionnaire, ouvrez le dossier *Applications*, cherchez le dossier *XAMPP*, puis exécutez *manager-osx*.

Accéder à la racine des documents

Sur un Mac, le dossier racine des documents de XAMPP, où sont stockés les documents à servir, se trouve dans :

```
/Applications/XAMPP/htdocs
```

Pour tester l'installation, entrez un peu de HTML comme suit dans TextEdit (ou tout autre éditeur capable d'enregistrer en texte brut) et enregistrez le fichier correspondant dans la racine des documents sous le nom *test.html*. Dans la barre d'adresse de votre navigateur, entrez *localhost/test.html* et le résultat devrait être identique à celui de la figure 2-14 :

```
<html>
 <head>
  <title>Un rapide test</title>
 </head>
 <body>
  Bonjour tout le monde!
 </body>
</html>
```

Installer LAMP sous Linux

Ce livre est essentiellement voué au PC et au Mac mais son contenu fonctionne tout aussi bien sur un ordinateur sous Linux. Le seul souci, c'est qu'il existe des dizaines de variétés de Linux et que chaque variété nécessite une installation de LAMP légèrement différente des autres. Comme je ne peux les envisager toutes, j'évite d'aborder le sujet dans ce livre.

Ceci dit, nombre de variantes de Linux s'installent avec un serveur web et MySQL préinstallés, donc la probabilité que vous disposiez déjà de ces serveurs est grande. Pour vous en assurer, essayez d'entrer la ligne suivante dans un navigateur et voyez si vous obtenez une page web prédéfinie, qui confirmerait que vous avez déjà un dossier racine de documents web :

```
http://localhost
```

Si cela fonctionne, alors il est fort probable que le serveur web Apache soit installé et qu'il fonctionne. Avec un peu de chance, MySQL est déjà installé et fonctionne aussi. Le cas échéant, communiquez avec votre administrateur système pour en être certain.

Si vous n'avez pas encore de serveur web installé, une version de XAMPP est néanmoins disponible à l'adresse *apachefriends.org*.

La séquence d'installation est comparable à celle illustrée dans la section pour Windows. Si vous avez besoin d'aide, les forums pour Linux en français sont là pour cela, et il reste le graal, mais il est en anglais (vous ne pourrez pas dire que vous n'aurez pas été prévenu!) : *apachefriends.org/faq_linux.html*.

Travailler à distance

Si vous disposez d'un accès à un serveur web distant complètement configuré avec PHP et MySQL, n'hésitez pas à l'utiliser pour vos développements web. Néanmoins, sauf si vous disposez d'une connexion très haut débit, ce n'est pas vraiment la meilleure solution. L'intérêt d'une installation purement locale se situe dans l'aisance du test de chaque modification que vous apportez à vos programmes et à vos pages, agrémentée de la vitesse incomparable.

L'accès à MySQL à distance risque de ne pas être très simple. Vous avez peut-être la possibilité d'accéder par Telnet ou SSH à votre serveur de base de données pour créer des bases de données et définir les permissions à partir de la ligne de commande. Votre hébergeur vous indique la meilleure manière de faire ce genre de choses et vous fournit le mot de passe qu'il attribue à votre accès à MySQL, en plus de celui qui, bien entendu, vous donne accès en premier lieu à votre serveur et votre espace d'hébergement.

Ouvrir une session

Je recommande aux utilisateurs de Windows d'installer au minimum PuTTY, disponible sur *http://putty.org*, pour avoir accès à Telnet et SSH (rappelez-vous que SSH est beaucoup plus sécuritaire que Telnet).

Sur Mac, vous disposez déjà de SSH. Sélectionnez le dossier *Applications*, puis *Utilitaires*, puis lancez le Terminal. Dans la fenêtre du terminal, ouvrez une session sur le serveur à l'aide de SSH comme suit :

```
ssh monlogin@serveur.com
```

où *serveur.com* est le nom du serveur sur lequel vous voulez ouvrir une session et *monlogin*, le nom d'utilisateur nécessaire pour vous connecter. Une invite vous demande ensuite d'entrer le mot de passe approprié à ce nom d'utilisateur et, s'il est correct, votre session s'ouvre.

Utiliser FTP

Pour transférer des fichiers entre votre ordinateur et le serveur distant, vous avez besoin d'un programme de transfert FTP. Il en existe de nombreux qui méritent votre attention et vous perdrez beaucoup de temps à trouver celui avec les fonctionnalités qu'il vous faut.

Actuellement, je recommande systématiquement FireFTP, du fait de ses avantages :

- C'est un plugiciel du navigateur web Firefox, donc il fonctionne sur toute plateforme où fonctionne Firefox.
- L'appeler s'avère aussi simple que de sélectionner un marque-page.
- C'est l'un des programmes de transfert FTP les plus rapides et faciles à utiliser que je connaisse.

Vous vous dites peut-être que vous utilisez Internet Explorer et que FireFTP n'est pas disponible pour lui. Je vous répondrais que vous allez développer des pages web donc vous vous devez de disposer d'une copie de chacun des principaux navigateurs web, installée sur votre ordinateur, comme je l'ai suggéré au début de ce chapitre.

Pour installer FireFTP, allez sur *http://fireftp.mozdev.org* à l'aide de Firefox et cliquez sur le lien Download FireFTP. Il ne « pèse » qu'un demi-mégaoctet et s'installe très rapidement. Dès qu'il est installé, redémarrez Firefox. Vous accédez ensuite à FireFTP dans le menu Outils (figure 2-15).

Autre excellent programme de transfert en FTP, FileZilla est un logiciel en source ouverte, disponible sur *http://filezilla-project.org*, pour Windows, Linux et Mac OS X 10.5 ou ultérieur.

Bien entendu, si vous avez l'habitude d'utiliser un programme FTP, continuez de l'utiliser car le mieux est de vous en tenir à ce que vous connaissez bien.

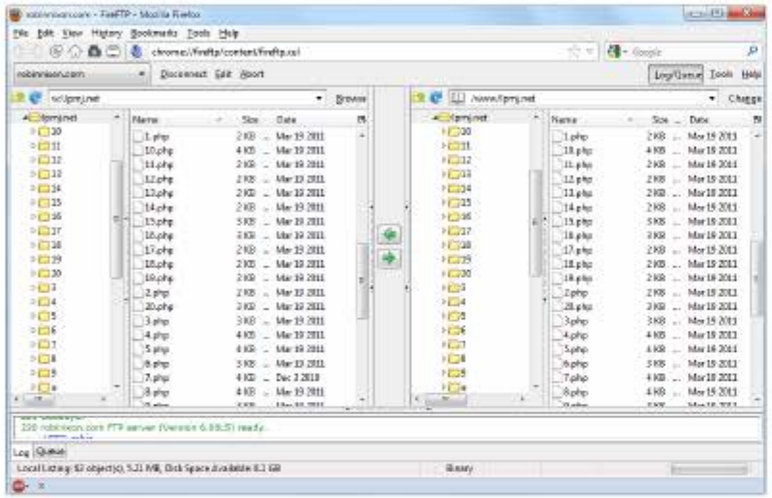


Figure 2-15. FireFTP offre un accès complet en FTP à partir de Firefox

Utiliser un éditeur de programmes

Bien qu'un éditeur de texte brut permette de rédiger du code HTML, PHP et JavaScript, des logiciels dédiés à l'écriture de programmes apportent des améliorations substantielles, puisqu'ils offrent des fonctionnalités très pratiques, telles que la coloration syntaxique. Les éditeurs de programmes actuels sont intelligents et montrent les erreurs de syntaxe avant même d'exécuter un programme. Lorsque vous aurez découvert un éditeur moderne, vous vous demanderez comment vous avez pu vous en passer autant d'années.

Nombre de bons programmes existent mais j'ai sélectionné Editra, parce qu'il est gratuit et disponible pour Mac, Windows et Linux/Unix. Pour le télécharger, allez sur *http://editra.org* et sélectionnez le lien Download dans le coin supérieur gauche de la page. Un autre lien donne accès à la documentation.

La figure 2-16 montre qu'Editra utilise la coloration syntaxique appropriée, c'est-à-dire des couleurs différentes pour clarifier le contenu du code. De plus, lorsque vous placez le curseur à côté de parenthèses ou d'accolades, Editra met en évidence la paire correspondante, ce qui permet de vérifier si vous en avez de trop ou trop peu. En fait, Editra propose bien plus encore, que vous découvrirez au fur et à mesure, et apprécierez.

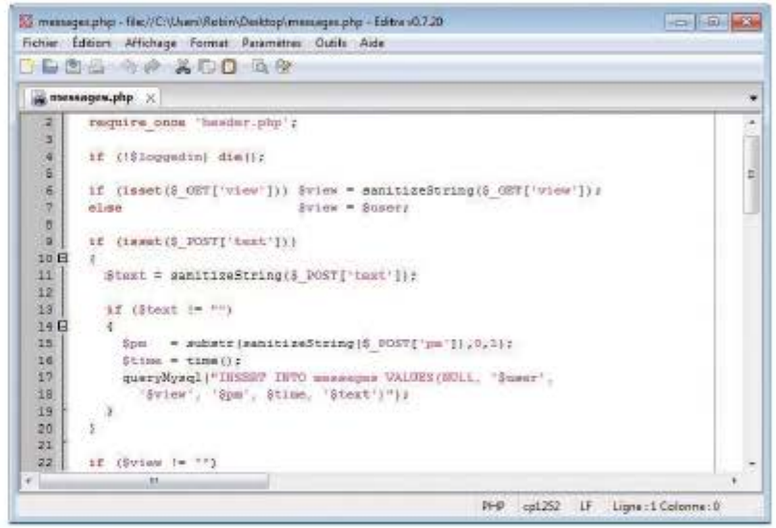


Figure 2-16. Les éditeurs de programmes surclassent largement les éditeurs de texte brut

Encore une fois, si vous avez l'habitude d'utiliser un éditeur de programmes différent, continuez car il vaut toujours mieux utiliser des logiciels que vous maitrisez.

Un éditeur de programmes : Notepad++

Petit éditeur de texte gratuit mais aussi tout à fait impressionnant, Notepad++ gère également la coloration syntaxique. Il a été conçu initialement en France (actuellement, l'équipe de développement est internationale), ce qui explique qu'il fonctionne aussi en français. Vous le trouverez à l'adresse *https://notepad-plus-plus.org/fr/*. Il fonctionne sous Windows et, avec la bibliothèque de compatibilité Wine, aussi sous Linux.

Utiliser un environnement de développement intégré

Aussi efficaces que soient devenus les éditeurs de programmes dans l'écriture de code de programmes, leurs performances pâlissent lorsqu'ils entrent en comparaison avec les *environnements de développement intégrés* (EDI), qui offrent de nombreuses fonctionnalités supplémentaires comme le débogage au sein même de l'éditeur et le test des programmes, ainsi que la description des fonctions et bien d'autres choses encore.

La figure 2-17 montre le très populaire EDI phpDesigner, avec un programme chargé dans le volet principal et, sur le côté droit, l'explorateur de code (*Code Explorer*) avec une liste des différentes classes, fonctions et variables utilisées.

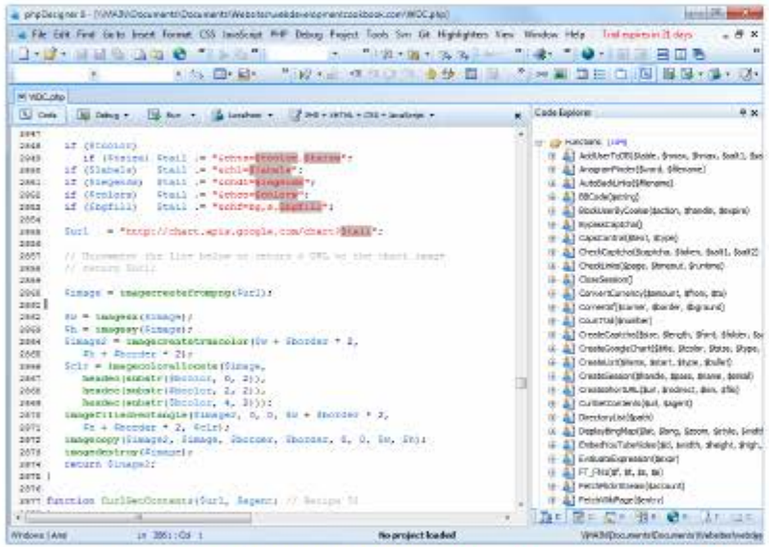


Figure 2-17. Lorsque vous utilisez un EDI telle que phpDesigner, le développement en PHP devient plus facile et rapide

Lorsque vous développez avec un EDI, vous pouvez définir des points d'arrêt (*break-point*) et ensuite exécuter tout ou des portions du code, pour arrêter l'exécution à ces points d'arrêt et examiner des informations qu'offre l'environnement à propos de l'état courant du programme, de ses variables et ainsi de suite.

Les exemples de ce livre constituent une aide à l'apprentissage de la programmation et, lorsque vous les entrez dans un EDI, vous pouvez les exécuter à la volée, sans même ouvrir un navigateur. Plusieurs EDI sont disponibles pour des platesformes diverses, dont la plupart sont commerciaux, donc payants, mais il en existe aussi quelques-uns gratuits. Le tableau 2-1 propose des EDI pour PHP les plus appréciés, avec les liens pour les télécharger.

Le choix d'un EDI est affaire de préférences personnelles donc, si vous comptez en utiliser un, je vous conseille d'en descendre deux ou plus et de les essayer d'abord, avant de choisir. Ils proposent tous une version d'évaluation ou sont gratuits, donc leur essai ne vous coutera que du temps.

Tableau 2-1. Une sélection d'EDI pour PHP, avec leur prix en dollars américains et leur compatibilité

EDI	Lien de téléchargement	Prix	Win	Mac	Lin
Eclipse PDT	http://eclipse.org/pdt/downloads/	Gratuit	✓	✓	✓
Komodo IDE	http://activestate.com/Products/komodo_ide	245 USD	✓	✓	✓
NetBeans	http://www.netbeans.org	Gratuit	✓	✓	✓
phpDesigner	http://mpsoftware.dk	39 USD	✓		
PHPEclipse	http://phpeclipse.de	Gratuit	✓	✓	✓
PhpED	http://nusphere.com	119 USD	✓		✓
PHPEdit	http://www.phpedit.com	119 USD	✓		

Prenez le temps d'installer un éditeur de programmes et (ou) un EDI avec lequel vous vous sentez à l'aise et vous serez prêt à essayer les exemples des chapitres suivants.

Équipé de ces outils, vous êtes prêt à aborder le chapitre 3, où vous commencez à explorer PHP plus en profondeur, découvrez comment faire fonctionner ensemble HTML et PHP, et la manière dont le langage PHP est structuré. Mais avant cela, voici quelques questions pour vérifier vos nouvelles connaissances.

Questions

1. Quelle est la différence entre WAMP, MAMP et LAMP ?
2. Qu'ont en commun l'adresse IP 127.0.0.1 et l'URL *http://localhost* ?
3. Quel est le but d'un programme FTP ?
4. Citez le principal inconvénient de travailler sur un serveur web distant.
5. Quel est l'intérêt d'utiliser un éditeur de programmes au lieu d'un simple éditeur de texte ?

Retrouvez les réponses du chapitre 2 dans l'annexe A.

CHAPITRE 3

Introduction à PHP

Le chapitre 1 expliquait que PHP est un langage qui permet, du côté du serveur web, de générer des résultats dynamiques, des résultats potentiellement différents chaque fois que le navigateur demande une page. Dans ce chapitre, vous allez apprendre les bases de ce langage simple mais néanmoins puissant. Ce sera le sujet de ce chapitre jusqu'au chapitre 7.

Je vous encourage à développer vos codes PHP dans un des EDI énumérés au chapitre 2, car ce genre d'outil permet de repérer rapidement les fautes de frappe et d'accélérer considérablement votre apprentissage, par rapport à d'autres éditeurs.

Nombre de ces environnements de développement permettent d'exécuter directement le code PHP pour voir les résultats évalués dans ce chapitre. Nous verrons aussi comment intégrer du PHP dans un fichier HTML, pour voir à quoi ressemblent les sorties dans une page web, c'est-à-dire de la manière dont les utilisateurs les verront. Cette étape n'est toutefois pas essentielle à ce stade.

En production, vos pages web combinent du PHP, du HTML et du JavaScript, ainsi que quelques instructions de MySQL, le tout mis en page grâce à des CSS, et possiblement avec d'autres éléments de HTML5. Mieux encore, chaque page peut mener à d'autres pages pour permettre aux utilisateurs de cliquer sur des liens et remplir des formulaires. Nous éviterons cependant cette complexité durant l'apprentissage de chaque langage. L'important pour l'heure est de nous concentrer sur l'écriture de code PHP et l'obtention des résultats attendus, ou du moins la compréhension des résultats que vous obtenez réellement !

Incorporer du PHP dans du HTML

Par défaut, les documents PHP se terminent par *.php*. Quand un serveur web rencontre cette extension dans un nom de fichier demandé, il le transfère automatiquement au processeur PHP. Bien entendu, comme les serveurs web sont configurables à souhait, certains développeurs web décident d'imposer leurs fichiers qui se terminent par *.htm* ou *.html* d'être également analysés par le processeur PHP, notamment pour cacher le fait qu'ils utilisent PHP !

Un programme PHP a la responsabilité de restituer un fichier propre, adapté à l'affichage dans un navigateur web. Au niveau le plus élémentaire, un document PHP ne « sort » que du HTML. Pour le prouver, ouvrez n'importe quel document HTML, comme *index.html*, et enregistrez-le avec l'extension *.php*, par exemple *index.php*, puis accédez à ce dernier par le navigateur : il s'affiche comme l'original.

Pour déclencher l'analyse de code PHP, donc l'insertion de commandes PHP dans un tel fichier, vous devez connaitre une nouvelle balise. En voici la première partie :

```
<?php
```

La première chose à remarquer ici, c'est que cette balise n'est pas « terminée », autrement dit, elle n'est pas « fermée ». Vous pouvez ainsi entrer des sections importantes de PHP dans cette balise et celle-ci ne se termine qu'avec la partie « fermante », qui prend la forme suivante :

```
?>
```

Le petit programme PHP suivant constitue un classique, le « Bonjour tout le monde! » :

Exemple 3-1. Appel de PHP

```
<?php
  echo "Bonjour tout le monde!";
?>
```

La manière d'utiliser cette balise est entièrement à votre discrétion, tant elle est souple. Certains programmeurs ouvrent la balise tout au début d'un document, pour la fermer tout à la fin, avec la génération du HTML au sein de commandes PHP. D'autres préfèrent ne laisser que des fragments aussi courts que possible de PHP avec ces balises, chaque fois que du script dynamique s'avère vraiment nécessaire, pour laisser le reste du document en HTML standard.

Parmi ces programmeurs, les seconds prétendent généralement que leur style de programmation génère un code plus rapide, tandis que les premiers considèrent que le gain de vitesse est trop négligeable pour justifier la complexité supplémentaire d'ouvrir et fermer PHP un peu partout dans un même document.

À mesure que vous progresserez, vous trouverez votre style préféré de développement PHP, mais pour des raisons de simplification des exemples, j'ai adopté dans ce livre l'approche qui consiste à conserver le minimum de transferts entre PHP et HTML, soit une ou deux fois dans un même document.

Tant que nous y sommes, notons une petite variante dans la syntaxe PHP. Si vous naviguez sur l'internet à la recherche d'exemples en PHP, vous trouverez aussi du code où les balises d'ouverture et de fermeture prennent l'allure suivante :

```
<?
  echo "Bonjour tout le monde!";
?>
```


Même s'il n'est pas évident que cela appelle bien l'analyseur PHP, cette syntaxe alternative est valide. Elle est toutefois fortement déconseillée, parce qu'elle n'est pas compatible avec XML et qu'elle utilise une forme obsolète (*deprecated*), c'est-à-dire qu'elle n'est plus recommandée et qu'elle risque de disparaitre dans les prochaines versions.

Si un de vos fichiers ne contient que du code PHP, vous pouvez omettre la portion fermante `?>`. Ceci constitue une bonne pratique, car elle permet de garantir que vous n'avez pas d'espace excessive induite par vos fichiers PHP. Cette précaution est particulièrement importante quand vous rédigez du code orienté objet.

Les exemples de ce livre

Afin de vous épargner le temps nécessaire à les entrer en totalité au clavier, tous les exemples de ce livre sont proposés sous forme d'une archive sur le site web d'accompagnement, à *www.goulet.ca/nixon*. Téléchargez l'archive sur votre ordinateur en cliquant sur le lien Exemples et code source dans la section Matériel d'accompagnement (figure 3-1).

Figure 3-1. Consultez les exemples de code de ce livre sur www.goulet.ca/nixon

En plus des exemples enregistrés par chapitre et par numéro d'exemple (comme *exemple3-1.php*), l'archive contient aussi un dossier intitulé *Exemples_nommes*, dans lequel vous trouverez des exemples que je vous suggère d'enregistrer sous un nom de fichier déterminé (comme pour l'exemple suivant, exemple 3-4, que vous enregistrerez sous le nom *test1.php*).

Structure de PHP

Cette section aborde quelques éléments fondamentaux. Ils ne sont pas complexes mais je vous conseille de les suivre soigneusement, pas à pas, car ils jettent les bases de tout le reste de ce livre. Comme toujours, des questions bien utiles figurent à la fin de ce chapitre, pour vous aider à évaluer ce que vous aurez appris.

Utiliser des commentaires

PHP autorise deux manières d'écrire des commentaires dans le code. La première transforme une simple ligne en commentaires à l'aide de deux barres de division en début de commentaire :

```
// Ceci est un commentaire.
```

Cette version des commentaires s'avère aussi une manière bien pratique de retirer temporairement une ligne de code d'un programme qui provoque des erreurs. Vous pouvez ainsi placer en commentaire une ligne de code de débogage quand vous n'en avez pas besoin, comme suit :

```
// echo "X vaut $x";
```

Ce style de commentaire permet aussi de commenter directement sur la même ligne une instruction pour en décrire le but, comme suit :

```
$x += 10;  // Incrémenter $x de 10.
```

Lorsque vous avez besoin de commentaires sur plusieurs lignes, un second type de commentaire existe, qui prend la forme de l'exemple 3-2.

Exemple 3-2. Commentaire sur plusieurs lignes

```
<?php
/* Ceci est une section de
   commentaires sur plusieurs
   lignes qui ne sera pas
   interprétée */
?>
```

Les caractères `/*` et `*/` vont par paire et permettent respectivement d'ouvrir et de clôturer un commentaire presque partout où vous le voulez dans votre code. La plupart du temps, voire toujours, les programmeurs utilisent cette construction pour placer en commentaire des sections entières de code qui ne fonctionnent pas correctement ou que, pour une raison ou une autre, ils ne souhaitent pas voir interprétées.

Une erreur très fréquente consiste à placer en commentaires avec ces caractères une vaste section de code qui contient déjà des commentaires avec ces mêmes caractères. Il n'est pas possible d'imbriquer des commentaires de cette manière. L'interpréteur ignore où se termine la zone de commentaires et considère que le premier `*/` clôture les commentaires, donc ce qui le suit génère un message d'erreur. D'où l'intérêt d'utiliser un éditeur de programmes ou un EDI qui met en évidence la syntaxe car ce genre d'erreur devient facile à repérer.

Syntaxe de base

PHP est un langage assez simple qui trouve ses racines dans C et Perl, même s'il ressemble plus à Java. Il est assez souple mais il exige de respecter quelques règles de syntaxe et de structure.

Points-virgules

Dans les exemples en PHP précédents, vous avez sans doute remarqué que chaque commande se termine par un point-virgule, comme suit :

```
$x += 10;
```

L'erreur la plus commune que vous rencontrerez en PHP réside dans l'oubli de ce point-virgule. Ceci entraine que PHP considère plusieurs instructions comme une seule, qu'il ne comprend pas. Il réagit en générant un message de type `parse error` ou erreur d'analyse.

Symbole $

Plusieurs langages de programmation utilisent le symbole `$`. Ainsi, si vous avez déjà programmé en langage BASIC, vous avez utilisé le `$` pour terminer des noms de variables et indiquer que ce sont des chaines de caractères.

En PHP, vous devez placer le `$` devant toutes les variables. Le `$` est indispensable pour accélérer le fonctionnement de l'analyseur PHP, car il sait alors instantanément quand il aborde une variable. Que vos variables soient des nombres, des chaines ou des tableaux (*arrays*), elles doivent prendre des formes comme celles illustrées dans l'exemple 3-3.

Exemple 3-3. Trois types différents d'affectations de variable

```
<?php
  $moncompteur = 1;
  $machaine  = "Bonjour";
  $monarray   = array("Un", "Deux", "Trois");
?>
```

Et c'est à peu près tout ce que vous devez retenir de la syntaxe. Au contraire de langages tels que Python, qui sont très stricts sur la manière de placer en retrait (*indent*) et de disposer le code, PHP vous laisse complètement libre d'utiliser des retraits et des espacements, comme bon vous semble. En pratique, utilisez raisonnablement les *espacements* et écrivez des commentaires complets pour faciliter la compréhension du code lorsque vous y revenez. Ils permettent également à d'autres programmeurs de lire plus facilement votre code.

Variables

Une jolie métaphore permet de mieux comprendre les variables en PHP : imaginez-les comme de petites (ou grandes) boites d'allumettes. C'est exactement cela : des boites d'allumettes que vous aurez repeintes et identifiées par des noms écrits sur elles.

Variables chaines de caractères

Imaginez que vous ayez une boite d'allumettes sur laquelle vous ayez écrit le mot *utilisateur*, qui représente le nom de l'utilisateur. Vous écrivez ensuite *Éric Dubé* sur un bout de papier que vous placez dans la boite (figure 3-2). Eh bien, c'est exactement le même procédé que vous suivez pour affecter une valeur de chaine de caractères, comme ceci :

```
$utilisateur = "Éric Dubé";
```

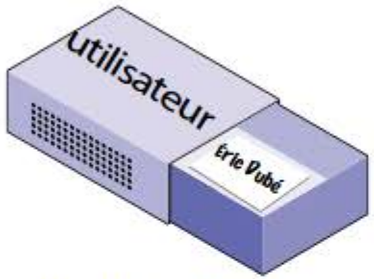


Figure 3-2. Imaginez les variables comme des boites d'allumettes contenant des éléments

Les guillemets verticaux indiquent que « Éric Dubé » est une *chaine* de caractères (*string*). Vous devez entourer chaque chaine de caractères, soit d'apostrophes verticales, soit de guillemets verticaux. Une légère nuance existe entre les deux, que nous expliquerons plus loin. Lorsque vous voulez savoir ce que contient la boite, vous l'ouvrez, prenez le bout de papier et le lisez. PHP fait exactement la même chose avec la ligne suivante :

```
echo $utilisateur;
```

Mais vous pouvez aussi affecter le contenu de la boite à une autre variable, ce qui revient à photocopier le papier et placer la copie dans une autre boite d'allumettes, comme suit :

```
$utilisateur_courant = $utilisateur;
```

Si vous êtes pressé d'essayer PHP par vous-même, vous pouvez entrer les exemples de ce chapitre dans un EDI (tel que ceux suggérés à la fin du chapitre 2) pour voir directement les résultats, ou entrer le code de l'exemple 3-4 dans un éditeur de programmes et enregistrer le fichier correspondant dans le dossier racine des documents de votre serveur (comme indiqué aussi au chapitre 2), sous le nom *test1.php*.

Exemple 3-4. Votre premier programme PHP

```
<?php // test1.php
  $utilisateur = "Éric Dubé";
  echo $utilisateur;
  echo "<br>";
  $utilisateur_courant = $utilisateur;
  echo $utilisateur_courant;
?>
```

Pour appeler ce programme, entrez ce qui suit dans la barre d'adresse du navigateur :

```
http://localhost/test1.php
```

Si pendant l'installation de votre serveur web (comme indiqué au chapitre 2), vous avez changé le port affecté au serveur en autre chose que 80, vous devez alors indiquer ce numéro de port dans l'URL de cet exemple et tous les suivants du livre. Ainsi, si vous avez choisi le port 8080, l'URL devient celle-ci :

```
http://localhost:8080/test1.php
```

Je ne le rappellerai plus par la suite donc rappelez-vous d'indiquer le numéro de port si nécessaire lors du test des exemples ou de l'écriture de votre propre code.

L'exécution de ce code produit l'affichage de deux occurrences du nom *Éric Dubé* : la première résulte de la commande `echo $utilisateur` ; la seconde résulte de la commande `echo $utilisateur_courant`.

Variables numériques

Les variables ne contiennent pas nécessairement du texte; elles peuvent aussi contenir des nombres. Revenons à l'analogie de la boite d'allumettes : pour stocker le nombre `17` dans la variable `$compteur`, cela reviendrait à placer 17 billes dans la boite et écrire sur la boite le mot compteur :

```
$compteur = 17;
```

Vous pouvez aussi utiliser un nombre *virgule flottante*, c'est-à-dire un nombre qui comporte une *virgule décimale*. Mais attention à ce niveau car, en PHP, la virgule décimale s'écrit avec un point, comme en anglais :

```
$compteur = 17.5;
```

Pour lire le contenu de la boite d'allumettes, vous l'ouvrez simplement et comptez les billes. En PHP, vous pouvez affecter la valeur de `$compteur` à une autre variable ou l'afficher dans le navigateur.

Tableaux

Que sont les tableaux ? En anglais, nous dirons des *arrays*. Pour vous les représenter, imaginez plusieurs boites d'allumettes collées ensemble, côte à côte. Par exemple, imaginez que vous devez stocker les noms de cinq joueurs de basketball dans un tableau nommé `$equipe`. Pour ce faire, vous pourriez coller cinq boites d'allumettes les unes sur les autres, écrire les noms de chacun des joueurs sur des bouts de papier et ranger chaque bout de papier dans une boite différente.

Au-dessus de l'ensemble des boites collées, vous écririez le mot équipe (figure 3-3). L'équivalent de ceci s'écrit en PHP comme suit :

```
$equipe = array('Pierre', 'Pauline', 'Jacques', 'Christian', 'Julie');
```

La syntaxe se complique un peu par rapport à ce que nous avons vu précédemment. Le code de construction d'un tableau repose sur la commande suivante,

```
array();
```

avec cinq chaines à l'intérieur. Chaque chaine est entourée d'apostrophes verticales.

Ensuite, pour savoir qui est le quatrième joueur, nous utilisons la commande suivante :

```
echo $equipe[3];  // Affiche le nom Christian
```

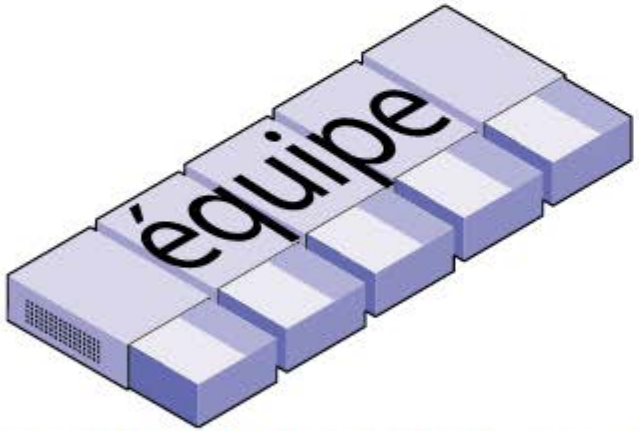


Figure 3-3. Un tableau est comme plusieurs boites d'allumettes collées ensemble

Le motif de la présence du 3 au lieu de 4 dans l'instruction précédente vient du fait que le premier élément d'un tableau en PHP porte le numéro, l'*indice*, zéro. Il en résulte que les joueurs de l'équipe portent les numéros de 0 à 4.

Tableaux à deux dimensions

Les tableaux permettent bien d'autres choses. Ainsi, au lieu de rangées simples de boites d'allumettes, vous pourriez construire une matrice bidimensionnelle ou même des matrices à trois dimensions ou plus.

Pour prendre un exemple plus explicite de tableau à deux dimensions, rappelez-vous le jeu du tic-tac-toe (ou morpion ou oxo) qui nécessite une structure de données de neuf cellules disposées en un carré de 3 cases sur 3. Pour la représenter avec des boites d'allumettes, il nous faudrait neuf boites collées en une matrice de trois rangées et trois colonnes (figure 3-4).

Figure 3-4. Un tableau multidimensionnel simulé à l'aide de boites d'allumettes

Vous pouvez à présent placer un bout de papier avec soit `x`, soit `o` dans la boite voulue à chaque mouvement des joueurs. Pour exprimer cela en PHP, vous créez un tableau contenant trois autres tableaux, comme dans l'exemple 3-5, qui montre une partie en cours.

Exemple 3-5. Définition d'un tableau bidimensionnel

```
<?php
  $oxo = array(array('x', ' ', 'o'),
               array('o', 'o', 'x'),
               array('x', 'o', ' '));
?>
```

Une fois encore, nous venons de monter d'un cran dans la complexité mais le tout demeure facile à comprendre si vous avez compris la syntaxe de base. Trois constructions `array()` résident, imbriquées dans la construction `array()` extérieure.

Ensuite, pour retourner le troisième élément de la deuxième rangée de ce tableau, nous utilisons la commande PHP suivante, qui affiche un `x` :

```
echo $oxo[1][2];
```

Retenez que les indices des tableaux (pointeurs vers les éléments au sein d'un array) débutent en zéro et non en un, de sorte que le [1] dans la commande précédente fait référence au deuxième élément des trois tableaux, et que le [2] fait référence au troisième emplacement au sein de ce tableau. La commande affiche donc l'élément de deuxième rangée et de troisième colonne.

Comme mentionné, les tableaux peuvent avoir encore plus de dimensions, simplement en créant plus de tableaux dans les tableaux. Néanmoins, ce livre ne traitera pas de tableaux de plus de deux dimensions.

Ne vous inquiétez pas si vous éprouvez des difficultés à saisir le concept des tableaux, car le chapitre 6 revient en détail sur le sujet.

Règles de dénomination des variables

Lors de la création de variables en PHP, suivez ces quatre règles :

- Les noms de variables doivent débuter par une lettre de l'alphabet ou le caractère _ (soulignement).
- Ils ne peuvent contenir que les caractères `a` à `z`, `A` à `Z`, `0` à `9` et le _ (soulignement).
- Ils ne peuvent pas contenir d'espace. Si le nom d'une variable doit contenir plusieurs mots, séparez-les par le caractère _ (par exemple `$nom_d_utilisateur`).
- Les noms de variables sont sensibles à la casse, c'est-à-dire que `$Meilleur_score` n'est pas la même variable que `$meilleur_score`.

Pour permettre l'utilisation de caractères ASCII étendus qui contiennent des accents, PHP reconnait les caractères correspondant aux octets 127 à 255 dans les noms de variables. À moins que votre code ne soit jamais maintenu que par des programmeurs habitués à ces caractères (é, è, à, ï, ç, œ, etc.), il est préférable de les éviter, parce que les programmeurs qui n'ont à leur disposition qu'un clavier anglais éprouvent des difficultés à y accéder. C'est la raison pour laquelle, au début de la section Tableau, page 42, pour représenter l'équipe de basketball, nous avons utilisé la variable `$equipe`, sans accent.

Opérateurs

Les opérateurs sont les commandes mathématiques, de chaines, de comparaison et logiques, tels que plus, moins, multiplié et divisé. En PHP, ces opérateurs ressemblent fort aux opérateurs arithmétiques. Ainsi, l'instruction suivante affiche `8` :

```
echo 6 + 2;
```

Avant de découvrir ce que PHP peut faire pour vous, attardez-vous un instant sur les différents opérateurs qu'il propose.

Opérateurs arithmétiques

Les opérateurs arithmétiques permettent d'effectuer des calculs d'algèbre classique. Les quatre opérations de base sont disponibles (addition, soustraction, multiplication et division), ainsi que le modulo, c'est-à-dire le reste de la division entière, et l'incrémentation et la décrémentation d'une valeur (tableau 3-1).

Tableau 3-1. Opérateurs arithmétiques

Opérateur	Description	Exemple
+	Addition	`$j + 1`
-	Soustraction	`$j - 6`
*	Multiplication	`$j * 11`
/	Division	`$j / 4`
%	Modulus (reste de la division entière)	`$j % 9`
++	Incrémentation	`++$j`
--	Décrémentation	`--$j`

Opérateurs d'affectation

Ces opérateurs servent à attribuer des valeurs aux variables. Ils vont du simple =, jusqu'aux `+=`, `-=` et ainsi de suite (tableau 3-2). L'opérateur `+=` ajoute la valeur située à droite de l'opérateur à celle de gauche, au lieu de totalement remplacer la valeur de gauche. Ainsi, si `$compteur` débute avec la valeur `5`, l'instruction

```
$compteur += 1;
```

met `$compteur` à 6, exactement comme l'instruction plus familière suivante :

```
$compteur = $compteur + 1;
```

Les chaines ont leur propre opérateur, le point (.), que vous verrez en détail à la section Concaténation de chaines, à la page 49.

Tableau 3-2. Opérateurs d'affectation

Opérateur	Exemple	Équivaut à
=	$j = 15	$j = 15
+=	$j += 5	$j = $j + 5
-=	$j -= 3	$j = $j - 3
*=	$j *= 8	$j = $j * 8
/=	$j /= 16	$j = $j / 16
.=	$j .= $k	$j = $j . $k
%=	$j %= 4	$j = $j % 4

Opérateurs de comparaison

Les opérateurs de comparaison sont utilisés dans une construction telle qu'une instruction `if`, où il faut comparer deux éléments. Par exemple, pour savoir si une variable que vous avez incrémentée a atteint une valeur donnée, ou si une autre variable est inférieure à une valeur définie, et ainsi de suite (tableau 3-3).

Retenez bien la différence entre `=` et `==`. Le premier est un opérateur d'affectation, tandis que le second est un opérateur de comparaison. Même les programmeurs les plus expérimentés les confondent quelquefois, lorsqu'ils doivent rédiger du code en urgence, donc soyez vigilant.

Tableau 3-3. Opérateurs de comparaison

Opérateur	Description	Exemple
==	est égal à	$j == 4
!=	est différent de	$j != 21
>	est plus grand que	$j > 3
<	est plus petit que	$j < 100
>=	est plus grand ou égal à	$j >= 15
<=	est plus petit ou égal à	$j <= 8

Opérateurs logiques

Si vous ne les avez jamais utilisés auparavant, les opérateurs logiques peuvent vous sembler un peu déconcertants. Mais ils apparaissent vite plus clairs lorsqu'il ne s'agit en fait que d'exprimer de la logique en français et de la transposer en anglais. Ainsi, vous vous dites que «s'il est plus tard que midi et plus tôt que 14 heures, alors je vais manger». Pour exprimer ceci en PHP, il suffit d'une instruction du genre de la suivante:

```
if ($heure > 12 && $heure < 14) aller_manger();
```


Ici, nous avons converti l'ensemble d'instructions pour aller réellement prendre un repas en une fonction que nous devrons créer par la suite et dénommée `aller_manger`. Le «si» de la proposition s'écrit `if`, comme en anglais. Le «alors» de la phrase est éludé parce qu'il est implicite et donc inutile.

Comme le montre l'exemple précédent, un opérateur logique permet de combiner les résultats de deux opérateurs de comparaison, tels que ceux de la section précédente. Un opérateur logique peut aussi servir d'entrée à un autre opérateur: «S'il est plus tard que midi et plus tôt que 14 heures, ou si une odeur de rôti envahit le couloir et les assiettes sont déjà sur la table.» La règle veut que, si quelque chose a une valeur vraie (`TRUE`) ou fausse (`FALSE`), ce quelque chose peut servir d'entrée dans un opérateur logique. Un opérateur logique ne peut prendre qu'une parmi deux valeurs, vrai ou faux, et produit un résultat vrai ou faux.

Tableau 3-4. Opérateurs logiques

Opérateur	Description	Exemple
&&	Et	$j == 3 && $k == 2
and	Et de priorité plus faible	$j == 3 and $k == 2
||	Ou	$j < 5 || $j > 10
or	Ou de priorité plus faible	$j < 5 or $j > 10
!	Non (ou pas)	! ($j == $k)
xor	Ou exclusif	$j xor $k

Remarquez que `&&` est généralement interchangeable avec `and`, et de même pour `||` et `or`. Mais comme `and` et `or` ont une priorité plus faible, dans certains cas, il est nécessaire d'ajouter des parenthèses pour imposer la préséance requise. En revanche, dans certains cas précis, *seuls* `and` ou `or` sont permis, comme dans l'instruction suivante, qui utilise l'opérateur `or`:

```
$html = file_get_contents($site) or die("Impossible de télécharger de $site");
```

Le plus inhabituel parmi ces opérateurs est `xor`, qui signifie *ou exclusif*. Il renvoie la valeur vraie (`TRUE`) si seulement une des valeurs comparées est vraie, mais faux (`FALSE`) si les deux valeurs comparées sont simultanément `TRUE` ou si elles sont simultanément `FALSE`. Pour comprendre à quoi sert le `xor`, imaginez que vous décidez de préparer votre propre nettoyant pour ustensiles ménagers. L'ammoniac est un bon nettoyant, de même que l'eau de javel, donc vous sélectionnez l'un d'eux dans votre potion nettoyante. Mais votre potion ne peut contenir les deux car leur mélange présente un danger. Pour exprimer ceci en PHP, vous écririez:

```
$ingredient = $ammoniac xor $javel;
```

Dans cet exemple, si `$ammoniac` ou `$javel` est `TRUE`, `$ingredient` devient `TRUE`. Mais s'ils sont tous les deux `TRUE`, ou tous les deux `FALSE`, alors `$ingredient` devient `FALSE`.

Affectation de variable

La syntaxe pour affecter une valeur à une variable est toujours *variable = valeur*. Ou, pour affecter la valeur d'une variable à une autre variable, elle devient *autre variable = variable*.

Quelques autres opérateurs d'affectation s'avèreront utiles, comme le suivant, que vous avez déjà vu :

```
$x += 10;
```

Celui-ci indique à l'analyseur PHP d'ajouter la valeur de droite (ici, la valeur `10`) à la variable de gauche, soit `$x`. De la même manière, la soustraction peut s'écrire :

```
$y -= 10;
```

Incrémentation et décrémentation de variable

L'ajout de 1 (l'incrémentation) et la soustraction de 1 (décrémentation) à une variable sont des opérations d'une telle fréquence que PHP fournit des opérateurs spéciaux pour cela. Vous pouvez utiliser une des formes suivantes pour remplacer les opérateurs `+=` et `-=`, quand la valeur de droite vaut 1 :

```
++$x;
--$y;
```

Si vous combinez cela à un test (instruction `if`), vous pouvez écrire, par exemple :

```
if (++$x == 10) echo $x;
```

Ceci indique à PHP d'incrémenter *d'abord* la valeur de `$x`, puis de la comparer à 10 et, s'ils sont égaux, alors d'afficher sa valeur. Mais si, au contraire, vous souhaitez que PHP incrémente (ou, dans l'exemple suivant, décrémente) une variable *après* en avoir testé la valeur, vous écrirez alors :

```
if ($y-- == 0) echo $y;
```

Ceci donne un résultat d'une différence subtile. Supposez que `$y` ait la valeur `0` avant l'exécution de cette instruction. La comparaison renvoie le résultat `TRUE` et `$y` n'est mis à `-1` qu'après la comparaison. Alors, que va afficher la commande `echo`, `0` ou `-1` ? Essayez de deviner, puis entrez l'instruction dans le processeur PHP pour vérifier. Comme cette combinaison d'instructions engendre de la confusion, il faut la considérer comme un exemple simplement pédagogique mais comme un mauvais style de programmation à éviter.

Pour résumer, le fait qu'une variable soit incrémentée ou décrémentée avant ou après le test dépend de l'emplacement de l'opérateur d'incrémentation ou de décrémentation avant ou après la variable.

Au fait, la réponse à la devinette précédente est `-1`, parce que `$y` est décrémenté après l'instruction `if` mais avant la commande `echo`.

Concaténation de chaines

Concaténer des chaines de caractères signifie les placer bout à bout. L'opérateur point (.) permet d'ajouter une chaine à la fin d'une autre, comme dans cet exemple :

```
echo "Vous avez " . $msgs . " messages.";
```

En supposant que la variable `$msgs` contient la valeur `5`, le résultat de cette ligne est :

```
Vous avez 5 messages.
```

De la même manière que vous ajoutez une valeur à une variable numérique à l'aide de l'opérateur `+=`, vous pouvez ajouter (*append*) une chaine à la fin d'une autre à l'aide de `.=` comme suit :

```
$journal .= $nouvelle_breve;
```

Ici, si `$journal` contient des informations sur des actualités et `$nouvelle_breve` une nouvelle information brève d'actualité, la commande ajoute cette dernière à la fin de la première, de sorte que `$journal` contient désormais les deux chaines de texte en une seule.

Types de chaines

PHP traite deux types de chaines de caractères qui se distinguent par les guillemets ou les apostrophes qui les entourent. Lorsque vous voulez affecter une chaine littérale et en préserver le contenu exact, utilisez les apostrophes verticales :

```
$info = 'Préfixez les variables avec un $ comme suit: $variable';
```

Dans ce cas, chaque caractère compris entre les apostrophes est attribué à `$info`. Si vous aviez mis des guillemets verticaux autour de la chaine, PHP aurait tenté d'évaluer `$variable` comme une variable, pour en recopier le contenu au lieu du nom de variable.

Par conséquent, lorsque vous voulez inclure le contenu d'une variable au sein d'une chaine, utilisez les guillemets verticaux autour de la chaine :

```
echo "Cette semaine $compteur personnes ont consulté votre profil.";
```

Vous comprendrez plus tard que cette syntaxe offre l'avantage d'une forme simplifiée de concaténation qui ne nécessite pas de point, ni de fermer les guillemets pour les rouvrir aussitôt et ainsi de suite, pour concaténer une chaine avec une autre. Ce mécanisme s'appelle la *substitution de variable*. Vous découvrirez que certaines applications l'utilisent abondamment, tandis que d'autres pas du tout.

Caractères d'échappement

Une chaine peut parfois contenir des caractères de signification particulière qui risquent d'entrainer une interprétation impropre. C'est le cas de l'apostrophe, très courante en français. La ligne de code ne fonctionne pas parce que la deuxième apostrophe rencontrée, dans « l'orthografe » est considérée par l'interpréteur PHP comme la fin de la chaine, donc il rejette la suite du texte comme étant une erreur :

```
$texte = 'Pour moi, l'orthografe est un vrai suplice!'; // Syntaxe incorrecte
```

Pour corriger cela, ajoutez une barre oblique inverse (\) juste avant l'apostrophe qui fait partie du texte, pour indiquer à PHP de la traiter comme un caractère littéral et de ne pas l'interpréter :

```
$texte = 'Pour moi, l\'orthografe est encore un vrai suplice!';
```

Utilisez cette astuce dans à peu près toutes les situations où PHP risque de retourner une erreur en essayant d'interpréter un caractère. Ainsi, dans le code suivant, la chaine entourée de guillemets verticaux au sein même de la chaine est analysée correctement :

```
$texte = "Elle écrivit dessus, \"Retour à l'expéditeur\".";
```

Remarquez qu'ici, la barre oblique inverse n'est pas indispensable devant l'apostrophe de « l'expéditeur » (mais elle aurait pu y être), parce que la chaine complète est entourée de guillemets et que l'apostrophe ne marque donc pas la fin de la chaine.

En outre, les caractères d'échappement permettent d'insérer des caractères spéciaux dans des chaines, comme des tabulations, des nouvelles lignes, des retours chariot. Ceux-ci sont représentés, comme vous l'aurez peut-être deviné, respectivement par \t, \n et \r. Suit un exemple d'insertion de tabulations pour définir un entête de tableau. Il ne sert que d'exemple pour illustrer les caractères d'échappement, car les pages web disposent de bien d'autres et meilleurs moyens de gérer la présentation :

```
$entete= "Date\tNom\tPaiement";
```

Ces caractères précédés d'une barre oblique inverse ne fonctionnent que dans les chaines entourées de guillemets. Dans les chaines entourées d'apostrophes, comme l'analyse est « littérale », ils apparaitraient sous l'horrible forme \t. Dans une chaine entourée d'apostrophes, seule l'apostrophe précédée de \ et la barre oblique inverse elle-même (\\) sont reconnues comme des séquences d'échappement.

Commandes sur plusieurs lignes

Il est parfois nécessaire d'écrire des commandes dont le contenu ne peut tenir que sur plusieurs lignes. Un echo d'un long texte, par exemple, nécessiterait normalement une succession de commandes echo (ou print) qui exigeraient beaucoup de temps pour éclater le texte en segments plus courts, sans compter les erreurs que cela engendrerait. Pour éviter cela, PHP offre deux facilités. La première consiste à placer plusieurs lignes entre guillemets, comme l'illustre l'exemple 3-6. Il est aussi possible d'affecter les variables, comme dans l'exemple 3-7.

Exemple 3-6. Une instruction echo de plusieurs lignes

```
<?php
  $auteur = "Steve Ballner";

  echo "Développeurs, Développeurs, développeurs, développeurs,
  développeurs, développeurs, développeurs, développeurs, développeurs!

  - $auteur.";
?>
```


Exemple 3-7. Affectation d'une chaine de plusieurs lignes

```
<?php
  $auteur = "Bill Gates";

  $texte = "Mesurer la progression de la programmation en termes de lignes de code
  revient à mesurer la progression de la construction d'un avion en termes de poids.

  - $auteur.";
?>
```

PHP offre une autre possibilité de séquence sur plusieurs lignes à l'aide de l'opérateur <<<, souvent qualifié de *document-ici* (*here-document*, ou *heredoc*), c'est-à-dire un moyen d'indiquer une chaine littérale, qui conserve les ruptures de lignes et autres espaces (y compris les retraits) dans le texte. L'exemple 3-8 en montre un échantillon.

Exemple 3-8. Autre possibilité d'instruction echo sur plusieurs lignes

```
<?php
  $auteur = "Brian W. Kernighan";

  echo <<<_END
  Le débogage est deux fois plus difficile que d'écrire le code pour
  la première fois. Par conséquent, si vous rédigez le code de la façon
  la plus intelligente qui soit, vous n'êtes, par définition, pas assez
  intelligent pour le déboguer.

  - $auteur.
_END;
?>
```

Ce code indique à PHP de sortir tout ce qui est compris entre les deux balises _END, comme une seule chaine entourée de guillemets, avec la petite nuance que les guillemets et apostrophes n'ont plus besoin de barres obliques inverses pour leur échappement. Cela signifie qu'il est possible pour un développeur, par exemple, de rédiger des sections entières de code HTML directement dans du code PHP et de n'y remplacer que les portions dynamiques par des variables en PHP.

Il est important de retenir que la balise fermante _END; *doit* apparaitre contre la marge gauche d'une nouvelle ligne et qu'elle doit être la *seule* présente sur cette ligne. Rien, ni un commentaire, ni même une seule espace ne doit figurer sur cette ligne. Une fois qu'un bloc de plusieurs lignes est ainsi clôturé, libre à vous de réutiliser la même balise.

Retenez : l'utilisation de la construction heredoc <<<_END … _END; n'impose plus d'ajouter de caractère \n de nouvelle ligne. Il suffit de presser [Entrée] et de débuter une nouvelle ligne. Au contraire de l'usage dans les chaines entourées de guillemets ou d'apostrophes, vous avez toute liberté d'utiliser les apostrophes et guillemets au sein d'un *heredoc*, sans les faire précéder par la barre oblique inverse.

L'exemple 3-9 montre comment exploiter la même syntaxe pour attribuer plusieurs lignes à une variable.

Exemple 3-9. Affectation à une variable d'une chaine sur plusieurs lignes

```
<?php
  $auteur = "Scott Adams";

  $out = <<<_END
  Les gens ordinaires croient que tant qu'il n'y a pas rupture, il ne faut pas réparer.
  L'ingénieur croit que, s'il n'y a pas rupture, il n'y a pas encore assez
  de fonctionnalités.

  - $auteur.
_END;
  echo $out;
?>
```

La variable `$out` reçoit tout le contenu compris entre les deux balises. Si vous concaténez au lieu d'affecter, utilisez `.=` à la place de `=` pour ajouter la chaine à la suite de `$out`.

Assurez-vous de ne pas placer de point-virgule à la fin de la première occurrence de `_END` car cela clôturerait le bloc multiligne avant même son début et provoquerait un message d'erreur d'analyse, `parse error`. La seule place autorisée du point-virgule se situe après la balise `_END` de fermeture, mais il demeure permis d'utiliser des points-virgules dans le bloc, au même titre que des caractères normaux.

J'ai choisi la balise `_END` dans ces exemples parce qu'il est peu probable qu'elle apparaisse ailleurs dans le code PHP et qu'elle est de ce fait unique. Vous avez tout le loisir d'utiliser la balise de votre choix, comme `_SECTION1`, `_OUTPUT` et ainsi de suite. De plus, pour différencier des balises comme celle-ci des variables et des fonctions, l'usage indique de manière générale de les préfixer par un caractère de soulignement et de les mettre en capitales (majuscules), mais vous pouvez très bien décider de ne pas le faire.

Une telle disposition de texte sur plusieurs lignes comme ici représente plus un moyen de faciliter la lecture du code PHP, car lorsqu'il s'étale dans une page web, les règles de mise en forme ont préséance et les espaces sont supprimés (mais `$auteur` est toujours remplacé par la valeur de cette variable).

Donc, si vous chargez ces exemples de sortie sur plusieurs lignes dans un navigateur, ils ne s'affichent pas sur plusieurs lignes, car tous les navigateurs considèrent les retours à la ligne simplement comme des espaces. En revanche, lorsque vous utilisez la fonctionnalité Afficher la source du navigateur, vous constatez que les retours à la ligne sont correctement disposés et que le contenu s'affiche sur plusieurs lignes.

Typage de variable

PHP s'avère un langage très souple en termes de typage. Ceci signifie qu'il n'impose pas de déclarer les variables avant de les utiliser et qu'il convertit toujours les variables dans le *type* requis par leur contexte avant d'y accéder.

Ainsi, si vous créez un nombre à plusieurs chiffres, vous pouvez en extraire le $n^{ième}$ chiffre, simplement en considérant que c'est une chaine. Le petit extrait de code

suivant, de l'exemple 3-10, multiplie les nombres 12345 et 67890, renvoie le résultat 838102050, qu'il place dans la variable `$nombre`.

Exemple 3-10. Conversion automatique de nombre en chaine

```
<?php
  $nombre = 12345 * 67890;
  echo substr($nombre, 3, 1);
?>
```

Au moment de l'affectation, `$nombre` est une valeur numérique mais à la deuxième ligne, un appel à la fonction PHP `substr` demande de renvoyer un caractère à partir de `$nombre`, à la quatrième position (rappelez-vous que les décalages, comme les indices de tableaux, débutent à zéro). Pour cela, PHP convertit `$nombre` en une chaine de neuf caractères, pour que `substr` puisse y accéder et renvoyer le caractère, 1 dans ce cas-ci.

La même chose se vérifie pour la conversion d'une chaine en un nombre et ainsi de suite. Dans l'exemple 3-11, la variable `$pi` est définie sous forme d'une valeur de chaine, automatiquement convertie en un nombre à virgule flottante à la troisième ligne, pour calculer l'aire du cercle, dont la valeur est de 78,5398175.

Exemple 3-11. Conversion automatique d'une chaine en nombre

```
<?php
  $pi    = "3.1415927";
  $rayon = 5;
  echo $pi * ($rayon * $rayon);
?>
```

En pratique, tout cela signifie que vous n'avez pas à vous inquiéter outre mesure des types de vos variables. Affectez-leur les valeurs qui ont un sens pour vous et laissez PHP les convertir si nécessaire. Ensuite, lorsque vous souhaitez récupérer vos valeurs, demandez-les simplement, par exemple avec une instruction `echo`.

Constantes

Les *constantes* ressemblent à des variables car elles contiennent des informations auxquelles vous accédez par la suite. Leur seule différence par rapport aux variables est représentée par leur appellation : elles sont constantes. En d'autres termes, dès que vous les définissez avec une valeur, celle-ci demeure invariable jusqu'à la fin du programme.

Un exemple d'utilisation d'une constante réside dans la mémorisation de l'emplacement de la *racine* (*root*) de votre serveur, c'est-à-dire le dossier qui contient les fichiers principaux de votre site web. Pour définir une telle constante, utilisez l'écriture suivante :

```
define("ROOT_LOCATION", "/usr/local/www/");
```

Ensuite, pour lire le contenu de la constante, faites-y référence comme pour toute variable normale (mais sans la préfixer d'un symbole dollar) :

```
$dossier = ROOT_LOCATION;
```

À partir de là, lorsque vous devrez exécuter votre code PHP sur un autre serveur dont le dossier racine diffère, vous n'aurez plus qu'une seule ligne de code à corriger.

Les deux seules choses à retenir à propos des constantes sont que vous *ne* devez *pas* les préfixer du symbole dollar ($), au contraire des autres variables, et que vous ne pouvez les définir qu'à l'aide de la fonction define.

Il est généralement conseillé de n'utiliser que des caractères en capitales pour nommer les constantes, surtout si d'autres personnes doivent aussi lire votre code.

Constantes prédéfinies

PHP prédéfinit des dizaines de constantes auxquelles vous n'accéderez généralement pas en tant que débutant en programmation PHP. Cependant, quelques-unes parmi elles, appelées *constantes magiques*, peuvent vous être utiles. Les noms de ces constantes magiques possèdent deux caractères de soulignement à leur début et à leur fin, pour éviter que vous ne nommiez accidentellement une de vos constantes avec un nom déjà pris. Le tableau 3-5 les énumère. Les concepts présentés dans ce tableau seront détaillés dans des chapitres ultérieurs.

Tableau 3-5. Constantes magiques en PHP

Constante magique	Description
__LINE__	Le numéro de ligne courante dans le fichier.
__FILE__	Le chemin complet du fichier. Si elle sert au sein d'un include, elle renvoie le nom du fichier inclus. En PHP 4.0.2, __FILE__ contient un chemin absolu dont les liens symboliques sont résolus, tandis que dans les versions antérieures, il peut contenir un chemin relatif dans certaines circonstances.
__DIR__	Le dossier du fichier courant. Si elle sert au sein d'un include, elle renvoie le nom du dossier du fichier inclus. Elle équivaut à dirname(__FILE__). Le nom de ce dossier ne possède pas de barre oblique à sa fin, sauf s'il s'agit du dossier racine. (Ajouté dans PHP 5.3.0.)
__FUNCTION__	Le nom de la fonction. (Ajouté dans PHP 4.3.0.) En PHP 5, renvoie le nom de la fonction courante, telle qu'elle a été déclarée (sensible à la casse). En PHP 4, ce nom apparait toujours en minuscules.
__CLASS__	Le nom de la classe. (Ajouté dans PHP 4.3.0.) En PHP 5, renvoie le nom de la classe courante telle qu'elle a été déclarée (sensible à la casse). En PHP 4, ce nom apparait toujours en minuscules.
__METHOD__	Le nom de la méthode courante de classe. (Ajouté dans PHP 5.0.0. Elle renvoie le nom de la méthode de classe courante, telle qu'elle a été déclarée (sensible à la casse).
__NAME_SPACE__	Le nom de l'espace de noms courant (sensible à la casse). Cette constante est définie au moment de la compilation. (Ajouté dans PHP 5.3.0.)

En pratique, l'usage de ces constantes s'avère surtout utile à des fins de débogage, lorsque vous devez insérer une ligne de code pour vérifier si l'exécution du programme atteint cette ligne :

```
echo "Ceci est la ligne " . __LINE__ . " du fichier " . __FILE__;
```

Cette ligne de code affiche dans le navigateur web le numéro de la ligne et le nom du fichier (avec son chemin complet) du script en cours d'exécution.

Différence entre les commandes echo et print

Jusqu'ici, vous avez utilisé la commande echo de plusieurs manières pour afficher du texte depuis le serveur dans le navigateur. Dans certains cas, vous avez affiché une chaine littérale. Dans d'autres cas, vous avez concaténé des chaines et même évalué des variables. Vous avez vu qu'il est possible d'étaler les résultats sur plusieurs lignes en sortie.

Il existe une alternative à echo, également utile : print. Ces deux commandes sont très similaires mais print est une construction du genre fonction, qui prend un seul paramètre et possède une valeur de retour (toujours égale à 1), tandis qu'echo est une construction qui fait purement partie du langage PHP. Comme ces commandes sont toutes deux des constructions, aucune d'elles ne nécessite de parenthèses.

Sans trop entrer dans les détails, la commande echo est légèrement plus rapide que print dans l'affichage général de texte, car elle ne renvoie pas de valeur. En revanche, du fait qu'elle n'est pas implémentée comme une fonction, echo ne peut faire partie d'une expression plus complexe, tandis que print le peut. L'exemple suivant sort à l'aide de print soit TRUE, soit FALSE, selon la valeur de la variable, chose que vous ne pourriez réaliser de la même manière avec echo, car cela provoquerait un message parse error :

```
$b ? print "TRUE" : print "FALSE";
```

Le point d'interrogation constitue ici un moyen d'interroger la variable $b pour voir si elle est vraie (valeur TRUE) ou fausse (valeur FALSE). Si $b est vraie, la partie de code située à gauche du deux-points (:) est exécutée ; sinon, la partie située à droite du : est exécutée.

D'une manière générale, les exemples de ce livre utilisent echo et je vous conseille d'en faire autant, jusqu'au stade du développement PHP où vous découvrirez la nécessité de faire appel à print.

Fonctions

Les *fonctions* permettent de séparer des sections de code qui réalisent des tâches particulières. Si, par exemple, vous devez souvent rechercher une date et la renvoyer dans un format bien précis, c'est là un bon exemple de tâche qu'une fonction peut remplir. Le code de cette tâche ne tient peut-être que sur trois lignes mais, si vous devez les reproduire des dizaines de fois dans un programme, celui-ci devient inutilement lourd et complexe. Donc vous avez tout intérêt à le convertir en une fonction. En plus, si vous décidez par la suite de changer le format des données, vous ne devrez plus le modifier qu'une seule fois, dans la fonction.

Placer ce code dans une fonction, non seulement réduit la taille du code source et le rend plus lisible, mais lui ajoute certaines fonctionnalités, parce qu'il est possible de passer des paramètres aux fonctions pour faire varier leur fonctionnement et elles peuvent aussi renvoyer une valeur au code appelant.

Pour créer une fonction, déclarez-la de la manière décrite dans l'exemple 3-12 suivant.

Exemple 3-12. Une déclaration de fonction simple

```
<?php
  function longdate($timestamp)
  {
    return date("l F jS Y", $timestamp);
  }
?>
```

Cette fonction prend (ou reçoit au moment de l'appel) en entrée un *timestamp* Unix, c'est-à-dire un horodatage, soit un nombre entier qui représente une date et une heure à la seconde près, sous la forme du nombre de secondes écoulées depuis le 1er janvier 1970 à 00 h 00, puis appelle la fonction PHP `date` avec une chaine qui impose un certain format pour renvoyer une date sous la forme longue anglaise *Tuesday May 2nd 2017*. Le nombre de paramètres définis entre les parenthèses de `longdate` dépend de vous; nous avons choisi d'en indiquer simplement un. Les accolades entourent tout le code à exécuter lors de l'appel de la fonction.

Pour sortir la date du jour à l'aide de cette fonction, il suffit de placer l'appel suivant dans votre code :

```
echo longdate(time());
```

Cet appel exploite la fonction intégrée de PHP `time` pour obtenir le timestamp courant et le passer à la nouvelle fonction `longdate`, qui renvoie (*return*) la chaine adéquate à la commande `echo` pour affichage. Si vous devez afficher la date d'il y a 17 jours, il vous suffit d'écrire l'appel suivant :

```
echo longdate(time() - 17 * 24 * 60 * 60);
```

Comme un timestamp est en secondes, cette ligne passe à `longdate` le timestamp Unix courant, moins le nombre de secondes écoulées depuis 17 jours, soit 17 jours × 24 heures × 60 minutes × 60 secondes.

Les fonctions peuvent accepter plusieurs paramètres et retourner plusieurs résultats, à l'aide de techniques que nous verrons en détail dans les chapitres suivants.

Portée des variables

Lorsque vous développez un très long programme, il se peut que vous commenciez à manquer de noms de variables. Heureusement, PHP permet de décider de la *portée* d'une variable. En d'autres termes, vous pouvez indiquer que vous voulez n'utiliser une variable `$temp` qu'à l'intérieur d'une fonction bien déterminée et oublier que vous l'ayez jamais utilisée quand la fonction renvoie le contrôle. En pratique, c'est le mode de fonctionnement par défaut des variables PHP.

Au contraire, vous pouvez informer PHP qu'une variable est de portée globale et donc accessible de partout ailleurs dans le programme.

Variables locales

Les *variables locales* sont celles créées à l'intérieur d'une fonction et accessibles uniquement au sein de cette fonction. Ce sont généralement des variables temporaires, qui servent à stocker des résultats de traitement ou de calculs partiels avant le retour de la fonction.

Les arguments d'une fonction constituent autant de variables locales. À la section précédente, nous avons défini une fonction qui accepte un paramètre nommé `$timestamp`. Celui-ci n'a de signification que dans le corps de la fonction; vous ne pouvez ni lire ni modifier sa valeur en dehors de la fonction.

Pour un autre exemple de variable locale, reprenons la fonction `longdate` et modifions-la légèrement, comme dans l'exemple 3-13 suivant.

Exemple 3-13. Légère modification de la fonction longdate

```
<?php
  function longdate($timestamp)
  {
    $temp = date("j M Y", $timestamp);
    return "Nous sommes le $temp";
  }
?>
```

Cette fois, nous affectons la valeur retournée par la fonction `date` à la variable temporaire `$temp`, pour l'insérer dans la chaine renvoyée par la fonction. Mais dès que la fonction retourne son résultat, la valeur de `$temp` est effacée, comme si elle n'avait jamais existé.

Pour voir les effets de la portée des variables, examinons la variante de code de l'exemple 3-14. Cette fois, `$temp` est créée avant l'appel de la fonction `longdate`.

Exemple 3-14. La tentative d'accéder à $temp dans la fonction échoue

```
<?php
  $temp = "Nous sommes le ";
  echo longdate(time());

  function longdate($timestamp)
  {
    return $temp . date("j M Y", $timestamp);
  }
?>
```

Cependant, comme `$temp` n'a jamais été créée dans la fonction `longdate` ni passée en paramètre, celle-ci ne peut y accéder. En conséquence, l'extrait de code affiche seulement la date, sans le texte prédécesseur. En réalité, le code affiche le message d'erreur `Notice: Undefined variable: temp`.

Par défaut, les variables créées au sein d'une fonction sont locales à cette fonction, tandis que les variables créées en dehors de toute fonction ne sont accessibles que par le code en dehors de toute fonction.

Des moyens existent pour réparer l'exemple 3-14, comme dans l'exemple 3-15 et l'exemple 3-16.

Exemple 3-15. La récriture pour accéder à $temp dans sa portée locale règle le problème

```
<?php
  $temp = "Nous sommes le ";
  echo $temp . longdate(time());

  function longdate($timestamp)
  {
    return date("j M Y", $timestamp);
  }
?>
```

L'exemple 3-15 déplace la référence à `$temp` en dehors de la fonction, de sorte que la référence apparait dans la même portée que la définition de la variable.

Exemple 3-16. Autre solution : passer $temp en argument de la fonction

```
<?php
  $temp = "Nous sommes le ";
  echo longdate($temp, time());

  function longdate($texte, $timestamp)
  {
    return $texte . date("j M Y", $timestamp);
  }
?>
```

La solution de l'exemple 3-16 passe `$temp` à la fonction `longdate` sous forme d'un argument supplémentaire. La fonction peut donc le lire dans une variable temporaire nommée `$texte` et sortir le résultat attendu.

L'oubli du principe de la portée d'une variable entraine une erreur très commune, donc rappelez-vous comment fonctionne ce principe pour faciliter le débogage de certains problèmes assez obscurs. Retenez qu'à moins que vous ayez déclaré une variable autrement, sa visibilité se limite à sa portée locale : soit dans la fonction courante, soit dans le code en dehors de toute fonction, selon l'endroit où vous l'avez créée en premier lieu et en fonction de l'endroit d'où vous l'appelez, soit à l'intérieur, soit à l'extérieur d'une fonction.

Variables globales

Certains cas nécessitent de disposer d'une variable à portée *globale*, car la totalité du code doit pouvoir y accéder. Certaines données peuvent également s'avérer vastes et complexes et vous préférez éviter de les passer en argument de fonctions.

Pour déclarer une variable de portée globale, utilisez le mot-clé `global`. Supposons que vous avez dans votre site web un moyen pour vos utilisateurs de s'identifier et d'ouvrir une session (*to log-in*), que vous souhaitez que tout votre code sache s'il interagit avec un utilisateur identifié ou un simple visiteur anonyme. Une manière d'atteindre ce but consiste à créer une variable globale telle que `$is_logged_in` :

```
global $is_logged_in;
```

À présent, votre fonction d'ouverture de session doit simplement mettre la variable à `1` lors d'une identification réussie, ou `0` en cas d'échec. Comme la porte de cette variable est globale, toutes les lignes de code du programme peuvent y accéder.

Remarquez au passage que la formulation `$is_quelquechose` en anglais résulte d'une convention fréquente chez les programmeurs. Nous essayons de répondre à la question : l'utilisateur est-il connecté ? Cette question est assez difficile à exprimer en français en code de programme (`&est_connecte`), car les caractères accentués sont fortement déconseillés, tandis que la signification est triviale en anglais : `$is_logged_in`.

N'utilisez les variables globales qu'avec la plus grande prudence. Je vous recommande de n'en créer que si vous n'avez absolument aucun autre moyen d'atteindre le résultat souhaité. En général, les programmes découpés en petites portions de code et de données bien distinctes boguent beaucoup moins et sont plus faciles à maintenir. Si vous avez un programme d'un millier de lignes de code (un jour, vous en aurez) et si vous découvrez qu'une variable globale contient une valeur incorrecte, à votre avis, combien de temps mettrez-vous pour trouver la ligne de code qu'il a réglée incorrectement ?

Plus vous créez de variables globales, plus vous vous exposez au risque de réutiliser le même nom qu'une de ces variables au niveau local, ou de penser que vous l'avez utilisée en local alors qu'en fait, elle a déjà été déclarée en global. Toutes sortes de bogues étranges peuvent naitre de telles situations et ils sont difficiles à repérer.

J'adopte parfois la convention qui consiste à écrire toutes les variables globales en capitales, de la même manière que les constantes, de sorte que je peux en repérer la portée au premier coup d'œil.

Variables statiques

À la section Variables locales de la page 57, j'indiquais que la valeur d'une variable disparait à la fin de l'exécution de sa fonction. Même si une fonction s'exécute de nombreuses fois, elle repart chaque fois avec une copie toute fraiche de ses variables et les réglages précédents n'ont plus aucun effet.

Voici un cas intéressant. Et si vous aviez une variable locale au sein d'une fonction, à laquelle vous ne voulez pas que d'autres parties du code aient accès, mais dont vous voulez conserver la valeur lors de l'exécution suivante de la fonction ? Dans quel but ? Pour éventuellement compter le nombre d'appels de cette fonction. La solution consiste à déclarer la variable comme statique, avec le mot-clé `static`, comme dans l'exemple 3-17.

Exemple 3-17. Une fonction avec une variable statique

```
<?php
  function test()
  {
    static $compteur = 0;
    echo $compteur;
    $compteur++;
  }
?>
```

Ici, la toute première ligne du corps de la fonction `test` crée une variable statique nommée `$compteur` et l'initialise à la valeur `0`. La ligne suivante affiche la valeur de cette variable ; la dernière l'incrémente.

Lors de l'appel suivant de la fonction, comme `$compteur` a déjà été déclarée comme statique, la première ligne de la fonction est ignorée. La valeur incrémentée précédemment de `$compteur` s'affiche, puis la variable est incrémentée de nouveau.

Si vous envisagez d'utiliser des variables statiques, remarquez que vous ne pouvez pas leur affecter le résultat d'une expression dans leur définition. Il n'est permis de les initialiser qu'à l'aide de valeurs prédéfinies (exemple 3-18).

Exemple 3-18. Valeurs permise et interdites dans la déclaration d'une variable statique

```
<?php
  static $int = 0;          // Autorisé
  static $int = 1+2;        // Interdit (génère un parse error)
  static $int = sqrt(144); // Interdit
?>
```

Variables superglobales

Depuis PHP 4.1.0, plusieurs variables sont désormais disponibles. Elles sont considérées comme des variables *superglobales*, ce qui signifie qu'elles sont fournies par l'environnement de PHP et qu'elles sont globales au sein du programme et donc accessibles absolument n'importe où.

Ces superglobales contiennent une pléthore d'informations à propos du programme en cours d'exécution et de son environnement (tableau 3-6). Elles sont structurées comme des tableaux associatifs, sujet décrit au chapitre 6.

Tableau 3-6. Variables superglobales de PHP

Superglobale	Contenu
$GLOBALS	Toutes les variables actuellement définies dans la portée globale du script. Les noms des variables constituent les clés du tableau.
$_SERVER	Des informations concernant, par exemple les entêtes, les chemins et l'emplacement de script. Les entrées de ce tableau sont créées par le serveur web et aucune garantie n'existe que le serveur web les fournisse.

Superglobale	Contenu
$_GET	Variables passées au script courant par l'entremise de la méthode HTTP Get.
$_POST	Variables passées au script courant par l'entremise de la méthode HTTP Post.
$_FILES	Éléments chargés dans le script courant par l'entremise de la méthode HTTP Post.
$_COOKIE	Variables transmises au script courant par l'entremise des témoins de connexion (cookies) HTTP.
$_SESSION	Variables de session accessibles au script courant.
$_REQUEST	Le contenu des informations envoyées par le navigateur; correspond par défaut à $_GET, $_POST et $_COOKIE.
$_ENV	Variables transmises au script courant par l'entremise de la méthode d'environnement.

Toutes les superglobales (sauf `$GLOBALS`) sont nommées en capitales avec un premier caractère de soulignement; par conséquent, évitez de nommer vos propres variables de la même manière pour éviter toute confusion.

Pour illustrer leur utilisation, examinons une information que nombre de sites emploient. Parmi les nombreuses informations fournies par les variables superglobales, figure l'URL de la page dont vient l'utilisateur et qui l'a amené à accéder à la page web courante. Pour connaitre cette information de référence, utilisez une instruction comme celle-ci :

```
$vient_de= $_SERVER['HTTP_REFERER'];
```

C'est aussi simple que cela. Cela dit, si l'utilisateur est allé directement sur cette page, en entrant par exemple l'URL directement dans la barre d'adresse de son navigateur, alors `$vient_de` contient une chaine vide.

Super-globales et sécurité

Un petit conseil de prudence est indispensable avant que vous n'utilisiez les variables superglobales, parce qu'elles sont souvent exploitées par les pirates pour trouver des failles et pénétrer dans un site web. Les pirates chargent `$_POST`, `$_GET` ou d'autres superglobales à l'aide de code malveillant, tel que des commandes Unix ou MySQL, qui risquent d'endommager ou d'afficher des données sensibles si vous y accédez de manière naïve.

Par conséquent, vous devez toujours désinfecter les superglobales avant de les utiliser. Une manière de le faire consiste à passer par la fonction PHP `htmlentities`. Celle-ci convertit tous les caractères en entités HTML. Par exemple, les caractères plus petit que (<) et plus grand que (>) sont convertis en leurs équivalents HTML, respectivement `<` et `>`, ainsi que les guillemets, les apostrophes, les barres obliques inverses et ainsi de suite, qu'il est possible de restituer sans risque.

En conséquence, une meilleure façon d'accéder à `$_SERVER` (et aux autres superglobales) serait la suivante :

```
$vient_de = htmlentities($_SERVER['HTTP_REFERER']);
```

L'utilisation de la fonction `htmlentities` en guise de nettoyage constitue une pratique de programmation importante en toutes circonstances où des données de l'utilisateur ou de tiers doivent être traitées, et non simplement les superglobales.

Vous avez déjà de solides connaissances en PHP. Au chapitre 4, vous commencerez à utiliser ce que vous avez appris pour édifier des expressions et contrôler le flux de programme, autrement dit, pour programmer réellement.

Avant d'aller plus loin, voici quelques questions pour vérifier si vous avez assimilé le contenu de ce chapitre.

Questions

1. Quelle est la balise à utiliser pour demander à PHP de commencer à interpréter du code de programme ? Et quelle est la forme abrégée de cette balise ?
2. Quels sont les deux types de balises de commentaires ?
3. Quel caractère doit figurer à la fin de chaque instruction en PHP ?
4. Quel est le symbole utilisé pour préfixer toute variable PHP ?
5. Que peut mémoriser une variable ?
6. Quelle est la différence entre `$variable = 1` et `$variable == 1` ?
7. Qu'est-ce qui vous permet de supposer que le caractère de soulignement est autorisé dans les noms de variables (`$utilisateur_courant`), tandis que le tiret ne l'est pas (`$utilisateur-courant`) ?
8. Les noms de variables sont-ils sensibles à la casse ?
9. Est-il permis d'utiliser des espaces dans les noms de variables ?
10. Comment fait-on pour convertir le type d'une variable en un autre type (par ex. une chaine en un nombre) ?
11. Quelle est la différence entre `++$j` et `$j++` ?
12. Les opérateurs `&&` et `and` sont-ils interchangeables ?
13. Comment faites-vous pour créer une instruction `echo` ou une affectation répartie sur plusieurs lignes ?
14. Pouvez-vous redéfinir une constante ?
15. Dans une chaine de caractères, comment peut-on convertir un guillemet ou une apostrophe en un caractère littéral, non interprété par PHP ?
16. Quelle est la différence entre les commandes `echo` et `print` ?
17. Quel est le but des fonctions ?
18. Comment faites-vous pour rendre une variable accessible partout dans un programme en PHP ?
19. Si vous générez des données au sein d'une fonction, quelles sont les deux manières de faire pour passer ces données au reste du programme ?
20. Quel résultat obtenez-vous en combinant une chaine et un nombre ?

Retrouvez les réponses du chapitre 3 dans l'annexe A.

CHAPITRE 4

Expressions et contrôle de flux en PHP

Le chapitre précédent a survolé un certain nombre de sujets que ce chapitre examine plus en profondeur, comme décider d'un choix (le branchement) et rédiger des expressions complexes. Au chapitre précédent, je voulais me concentrer sur la syntaxe la plus fondamentale et les opérations de base de PHP mais je n'ai pu éviter d'évoquer des concepts plus avancés. À présent, je peux aborder ce que j'ai laissé en suspens, pour vous permettre d'utiliser correctement ces puissantes fonctionnalités de PHP.

Dans ce chapitre, vous acquerrez l'expérience approfondie du fonctionnement pratique de la programmation en PHP et du contrôle du flux d'exécution d'un programme.

Expressions

Commençons par la partie la plus fondamentale de n'importe quel langage de programmation : les *expressions*.

Une expression est une combinaison de valeurs, de variables, d'opérateurs et de fonctions qui produit une valeur. À l'école secondaire ou au lycée, nous apprenons des expressions d'algèbre telles que celle-ci :

```
y = 3(abs(2x) + 4)
```

Cette expression se traduit en PHP par :

```
$y = 3 * (abs(2 * $x) + 4);
```

La valeur renvoyée, (`y` ou `$y`, dans ce cas-ci) peut être un nombre, une chaine ou une *valeur booléenne* (appelée ainsi en hommage à George Boole, un mathématicien et philosophe du 19e siècle). Pour l'instant, vous êtes familier avec les deux premiers types de valeurs et je vais vous expliquer ce qu'est le troisième.

VRAI ou FAUX ?

Une valeur booléenne peut prendre une parmi deux valeurs possibles : vrai (`TRUE`) ou faux (`FALSE`). Ainsi, l'expression `20 > 9` (20 plus grand que 9) est *vraie*, tandis que l'expression `5 == 6` (5 est égal à 6) est *fausse*.

(Vous pouvez combiner des opérations booléennes à l'aide d'opérateurs tels que AND, OR et XOR, que nous verrons plus loin dans ce chapitre.)

Remarquez que j'utilise des lettres capitales pour les noms TRUE et FALSE, car ces deux valeurs sont des constantes prédéfinies en PHP. Mais vous pouvez aussi utiliser les versions en bas de casse (minuscules) si vous le préférez, car elles sont également prédéfinies. En pratique, les versions en bas de casse sont plus stables, car PHP interdit de les redéfinir, tandis que leurs pendants en capitales peuvent être redéfinis, petit détail que vous devez toujours conserver à l'esprit lorsque vous importez du code de quelqu'un d'autre.

L'exemple 4-1 montre quelques expressions simples : les deux que j'ai citées, puis deux autres. Chaque ligne affiche une lettre de a à d, suivie d'un deux-points, puis du résultat de l'expression. La balise <br> sert ici à forcer le retour à la ligne et donc à étaler les résultats sur quatre lignes en HTML.

À présent que nous sommes pleinement dans l'ère du HTML5 et que XHTML n'est plus pressenti pour remplacer HTML, il n'est plus nécessaire d'utiliser la forme autofermante <br /> de la balise <br> ni aucun élément vide (ceux sans balise de fermeture), car la / est désormais facultative. Par conséquent, j'ai choisi d'utiliser le style le plus simple dans ce livre. Si vous avez déjà rédigé des balises HTML non vides autofermantes (comme), sachez qu'elles ne fonctionneront pas en HTML5, parce qu'il ignore la /, et vous devrez les remplacer (par exemple) par ... . Vous devez toutefois toujours utiliser la forme <br /> de la syntaxe HTML lorsque vous faites appel à du XHTML.

Exemple 4-1. Quatre expressions booléennes simples

```
<?php
  echo "a: [" . (20 > 9) . "]<br>";
  echo "b: [" . (5 == 6) . "]<br>";
  echo "c: [" . (1 == 0) . "]<br>";
  echo "d: [" . (1 == 1) . "]<br>";
?>
```

Le résultat de ce code est le suivant :

```
a: [1]
b: []
c: []
d: [1]
```

Remarquez que l'évaluation des deux expressions a: et d: donne TRUE, qui correspond à la valeur 1. Mais l'évaluation de b: et c: donne FALSE et n'affiche aucune valeur, parce qu'en

PHP, la constante FALSE est prédéfinie à NULL, c'est-à-dire rien. Pour vérifier cela par vous-même, entrez le code de l'exemple 4-2.

Exemple 4-2. À quoi correspondent les valeurs TRUE et FALSE ?

```
<?php // test2.php
  echo "a: [" . TRUE  . "]<br>";
  echo "b: [" . FALSE . "]<br>";
?>
```

Le résultat est le suivant :

```
a: [1]
b: []
```

Remarquez que dans certains langages, FALSE peut être défini à 0, voire à -1, donc il vaut toujours mieux vérifier ces définitions dans chaque langage.

Valeurs littérales et variables

La forme la plus simple d'une expression est celle du *littéral*, ou *libellé*, autrement dit de la *valeur littérale*. Il indique une valeur que l'on peut interpréter ou évaluer littéralement, comme le nombre 73 et la chaîne "Salut". Une expression peut aussi n'être qu'une simple variable, qui s'évalue à la valeur qui lui a été affectée. Ce sont deux types d'expressions parce qu'ils renvoient une valeur.

L'exemple 4-3 illustre trois littéraux et deux variables, qui revoient tous une valeur, quel que soit son type.

Exemple 4-3. Valeurs littérales et variables

```
<?php
  $monnom = "Bruno";
  $monage  = 37;

  echo "a: " . 73       . "<br>"; // Littéral numérique
  echo "b: " . "Salut"  . "<br>"; // Littéral chaine
  echo "c: " . FALSE    . "<br>"; // Littéral constant
  echo "d: " . $monnom  . "<br>"; // Variable chaine
  echo "e: " . $monage  . "<br>"; // Variable numérique
?>
```

Comme vous pouvez vous y attendre, vous voyez une valeur de retour de toutes ces expressions, sauf pour le c:, qui renvoie l'évaluation de FALSE, c'est-à-dire rien, en tant que résultat :

```
a: 73
b: Salut
c:
d: Bruno
e: 37
```

La combinaison avec des opérateurs permet de créer des expressions plus complexes qui, évaluées, produisent des résultats utiles.

La combinaison de constructions d'affectation ou de contrôle de flux de programme avec des expressions constitue une *instruction*. L'exemple 4-4 en illustre des exemples. La première affecte le résultat de l'expression `366 - $numero_jour` à la variable `$jours_avant_nouvel_an`, et la seconde affiche un message convivial seulement si l'évaluation de l'expression `$jours_avant_nouvel_an < 30` donne `TRUE`.

Exemple 4-4. Une expression et une instruction

```
<?php
  $numero_jour          = 340;              // Affectation par valeur
  $jours_avant_nouvel_an = 366 - $numero_jour; // Affectation par expression

  if ($jours_avant_nouvel_an < 30)          // Condition
  {
    echo "Le Nouvel An est pour bientôt.";  // Instruction
  }
?>
```

Opérateurs

PHP propose toute une gamme d'opérateurs qui couvrent de nombreux besoins : arithmétiques, de chaine, logiques, mais aussi d'affectation, de comparaison et bien d'autres (tableau 4-1).

Tableau 4-1. Types d'opérateurs de PHP

Opérateur	Description	Exemple
Arithmétique	Mathématiques de base	`$a + $b`
Tableau	Union de tableaux	`$a + $b`
Affectation	Multiplication	`$a = $b + 23`
Niveau des bits	Manipuler les bits dans des octets	`12 ^ 9`
Comparaison	Comparer deux valeurs	`$a < $b`

Opérateur	Description	Exemple
Exécution	Exécuter des commandes système	`` `ls -al` ``
Incrémentation/décrémentation	Ajouter ou soustraire 1	`$a++`
Logique	Booléen	`$a and $b`
Chaine	Concaténation	`$a . $b`

Chaque opérateur prend un nombre d'opérandes différent :

- Les opérateurs *unaires*, comme l'incrémentation (`$a++`) ou la négation (`-$a`), ne prennent qu'un seul opérande.
- Les opérateurs *binaires*, qui représentent l'essentiel des opérateurs de PHP, dont l'addition, la soustraction, la multiplication et la division.
- Un seul opérateur *ternaire*, qui adopte la forme `$test ? $si_vrai : $si_faux`. Il s'agit d'une instruction `if` écrite en une seule ligne, qui effectue un choix entre les portions `$si_vrai` et `$si_faux`, en fonction de l'état de l'évaluation de `$test`.

Préséance des opérateurs

Si tous les opérateurs avaient la même préséance, ou priorité les uns sur les autres, ils seraient traités dans l'ordre de leur analyse. En pratique, nombre d'opérateurs ont la même préséance, comme l'illustre l'exemple 4-5.

Exemple 4-5. Trois expressions équivalentes

```
1 + 2 + 3 - 4 + 5
2 - 4 + 5 + 3 + 1
5 + 2 - 4 + 1 + 3
```

Ici, alors que les nombres et l'opérateur qui les précède ont été redistribués, le résultat de chacune de ces expressions est et reste la valeur 7, parce que les opérateurs plus et moins ont la même préséance. Nous pouvons faire de même avec la multiplication et la division (exemple 4-6).

Exemple 4-6. Trois autres expressions équivalentes

```
1 * 2 * 3 / 4 * 5
2 / 4 * 5 * 3 * 1
5 * 2 / 4 * 1 * 3
```

Le résultat demeure chaque fois 7,5. En revanche, les choses se compliquent quand nous mélangeons des opérateurs de préséances différentes dans une expression, comme dans l'exemple 4-7.

Exemple 4-7. Trois autres expressions mais avec des opérateurs de préséances différentes

```
1 + 2 * 3 - 4 * 5
2 - 4 * 5 * 3 + 1
5 + 2 - 4 + 1 * 3
```

Sans la préséance des opérateurs, ces trois expressions donneraient respectivement 25, -29 et 12. Mais comme la multiplication et la division ont préséance sur l'addition et la soustraction, tout se passe comme si des parenthèses encadraient certaines portions de ces expressions, pour prendre l'aspect illustré à l'exemple 4-8, si elles étaient visibles.

Exemple 4-8. Les trois mêmes expressions avec les parenthèses implicites

```
1 + (2 * 3) - (4 * 5)
2 - (4 * 5 * 3) + 1
5 + 2 - 4 + (1 * 3)
```

En clair, PHP doit évaluer d'abord les sous-expressions entre parenthèses pour déduire les expressions semi-complètes de l'exemple 4-9.

Exemple 4-9. Les mêmes expressions après évaluation des sous-expressions entre parenthèses

```
1 + (6) - (20)
2 - (60) + 1
5 + 2 - 4 + (3)
```

Le résultat final de ces expressions donne respectivement -13, -57 et 6, ce qui diffère radicalement des résultats 25, -29 et 12 que nous aurions déduits si la préséance des opérateurs n'existait pas.

Bien entendu, pour contourner la préséance prédéfinie des opérateurs, vous pouvez insérer vos propres parenthèses et ainsi forcer les résultats initiaux, si la préséance des opérateurs n'existait pas (exemple 4-10).

Exemple 4-10. Imposer l'évaluation de gauche à droite

```
((1 + 2) * 3 - 4) * 5
(2 - 4) * 5 * 3 + 1
(5 + 2 - 4 + 1) * 3
```

Les parenthèses correctement placées, nous retrouvons les valeurs 25, -29 et 12, respectivement.

Le tableau 4-2 énumère les opérateurs de PHP, par préséance décroissante.

Tableau 4-2. Préséance (décroissante) des types d'opérateurs de PHP

Opérateurs	Type
`()`	Parenthèses
`++ --`	Incrémentation/décrémentation
`!`	Logique
`* / %`	Arithmétique
`+ - .`	Arithmétique et chaine
`<< >>`	Niveau des bits
`< <= > >= <>`	Comparaison
`== != === !==`	Comparaison
`&`	Niveau des bits (et références)
`^`	Niveau des bits
`|`	Niveau des bits
`&&`	Logique
`||`	Logique
`? :`	Ternaire
`= += -= *= /= .= %= &= != ^= <<= >>=`	Affectation
`and`	Logique
`xor`	Logique
`or`	Logique

Associativité

Nous avons vu que le traitement des expressions s'effectue de gauche à droite, avec les exceptions de préséance des opérateurs, lorsqu'elle s'applique. Pourtant, certains opérateurs exigent un traitement de droite à gauche. Ce sens de traitement s'appelle l'*associativité* de l'opérateur. Dans le cas de certains opérateurs, aucune associativité ne s'applique.

L'associativité revêt de l'importance dans les cas où vous n'imposez aucune préséance, donc vous devez être attentif aux actions par défaut des opérateurs, détaillées au tableau 4-3, qui énumère les opérateurs et leur associativité.

Dans ce tableau, nous utilisons le terme cast. Il représente une conversion de type imposée, « explicite ». PHP peut traiter automatiquement un nombre comme une chaine lorsque c'est nécessaire mais dans certains cas, il s'avère indispensable d'imposer une *conversion de type* : c'est la notion de cast.

Tableau 4-3. Associativité des opérateurs

Opérateur	Description	Associativité
`CLONE NEW`	Créer un nouvel objet	Aucune
`< <= >= == != === !== <>`	Soustraction	Aucune
`!`	Non logique	Droite
`~`	Négation au niveau des bits	Droite
`++ --`	Incrémentation, décrémentation	Droite
`(int)`	Cast en entier	Droite
`(double) (float) (real)`	Cast en nombre flottant	Droite
`(string)`	Cast en chaine	Droite
`(array)`	Cast en tableau	Droite
`(object)`	Cast en objet	Droite
`@`	Inhibition de rapport d'erreur	Droite
`= += -= *= /=`	Affectation	Droite

Opérateur	Description	Associativité
.= %= &= |= ^= <<= >>=	Affectation	Droite
+	Addition et plus unaire	Gauche
-	Soustration et négation	Gauche
*	Multiplication	Gauche
/	Division	Gauche
%	Modulo	Gauche
.	Concaténation de chaines	Gauche
<< >> & ^ |	Opérations au niveau des bits	Gauche
?:	Opérateur ternaire	Gauche
|| && and or xor	Opérateurs logiques	Gauche
,	Séparateur (virgule)	Gauche

L'associativité des opérateurs permet d'écrire des choses très intéressantes comme dans l'exemple 4-11, qui illustre l'utilisation de l'opérateur d'affectation pour régler trois variables à la valeur `0`.

Exemple 4-11. Instruction d'affectation multiple

```
<?php
  $niveau = $score = $duree = 0;
?>
```

Cette affectation multiple n'est possible que parce que la première évaluation se situe dans la partie la plus à droite de l'expression, pour se poursuivre de droite à gauche.

En tant que débutant en PHP, pour éviter d'éventuels pièges liés à l'associativité des opérateurs, isolez et imbriquez vos sous-expressions à l'aide de parenthèses pour imposer l'ordre d'évaluation. Cela permet aussi à d'autres programmeurs qui doivent éventuellement maintenir votre code de comprendre ce qui s'y passe exactement.

Opérateurs relationnels

Les opérateurs relationnels testent deux opérandes et renvoient un résultat booléen valant soit `TRUE`, soit `FALSE`. Trois types d'opérateurs relationnels existent : d'*égalité*, de *comparaison* et *logiques*.

Opérateurs d'égalité

Nous avons déjà rencontré quelques fois dans ce chapitre l'opérateur relationnel d'égalité `==` (deux signes égal accolés), qu'il est essentiel de ne pas confondre avec l'opérateur d'affectation, `=` (un seul signe égal). Dans l'exemple 4-12, la première instruction affecte une valeur et la seconde teste l'égalité.

Exemple 4-12. Affectation de valeur et test d'égalité

```
<?php
  $mois = "mars";

  if ($mois == "mars") echo "Voilà le printemps";
?>
```

Vous constatez qu'en renvoyant `TRUE` ou `FALSE`, l'opérateur relationnel d'égalité permet de tester des conditions à l'aide, par exemple d'une instruction `if`. Mais l'histoire ne s'arrête pas là car PHP est un langage à typage faible, c'est-à-dire que, si les deux opérandes d'une expression d'égalité sont de types différents, PHP les convertit en un type qui lui parait le plus sensé.

Par exemple, il convertit des chaines ne contenant que des nombres en des nombres lorsqu'il doit les comparer à un nombre. Dans l'exemple 4-13, `$a` et `$b` sont deux chaines différentes donc vous vous attendez probablement à ce qu'aucune des instructions `if` n'affiche de résultat.

Exemple 4-13. Opérateurs d'égalité et d'identité

```
<?php
  $a = "1000";
  $b = "+1000";

  if ($a == $b)  echo "1";
  if ($a === $b) echo "2";
?>
```

Pourtant, lorsque vous exécutez le code de l'exemple, vous constatez qu'il affiche le nombre `1`, ce qui signifie que la première instruction `if` est évaluée à `TRUE`. Cela est dû au fait que les deux chaines ont d'abord été converties en nombres et que `1000` a la même valeur numérique que `+1000`.

Par contraste, la deuxième instruction `if` fait appel à l'opérateur relationnel d'*identité*, composé de trois signes égal successifs, qui empêche PHP de convertir automatiquement les types. Donc `$a` et `$b` sont comparées en tant que chaines et, comme elles sont différentes, aucun résultat n'apparait.

Dans le même ordre d'idée que l'utilisation de parenthèses pour imposer l'ordre de préséance des opérations, dès que vous avez un doute quant à la conversion de types par PHP, vous pouvez utiliser l'opérateur d'identité pour désactiver ce comportement.

De même que l'opérateur d'égalité permet de tester l'égalité d'opérandes, il est possible d'en vérifier la non-égalité à l'aide de `!=`, l'opérateur d'inégalité. En guise d'illustration, l'exemple 4-14 reprend et modifie légèrement l'exemple 4-13, pour remplacer les opérateurs d'égalité et d'identité par leurs inverses respectifs.

Exemple 4-14. Opérateurs d'inégalité et de non-identité

```
<?php
  $a = "1000";
  $b = "+1000";

  if ($a != $b)  echo "1";
  if ($a !== $b) echo "2";
?>
```

Comme vous pouvez vous y attendre, la première instruction `if` n'affiche pas le nombre `1`, parce que le code demande si les variables `$a` et `$b` ne sont pas égales sur le plan numérique.

La seconde instruction `if` entraine l'affichage du `2` car `a$` et `$b` ne sont pas égales dans leur type d'opérande actuel, donc le résultat du test est `TRUE`: elles ne sont pas les mêmes.

Opérateurs de comparaison

Les opérateurs de comparaison permettent de tester davantage que l'égalité ou l'inégalité. PHP offre `>` (plus grand que), `<` (plus petit que), `>=` (plus grand ou égal à) et `<=` (plus petit ou égal à) et l'exemple 4-15 en propose l'illustration.

Exemple 4-15. Les quatre opérateurs de comparaison

```
<?php
  $a = 2; $b = 3;

  if ($a > $b)  echo "$a est plus grand que $b<br>";
  if ($a < $b)  echo "$a est plus petit que $b<br>";
  if ($a >= $b) echo "$a est plus grand ou égal à $b<br>";
  if ($a <= $b) echo "$a est plus petit ou égal à $b<br>";
?>
```

Dans cet exemple où `$a` vaut `2` et `$b` vaut `3`, ceci s'affiche:

```
2 est plus petit que 3
2 est plus petit ou égal à 3
```

Essayez cet exemple par vous-même, modifiez les valeurs de `$a` et `$b` pour constater les résultats. Essayez de leur donner la même valeur pour voir ce qui se produit.

Opérateurs logiques

Les opérateurs logiques produisent comme résultat soit `TRUE`, soit `FALSE`. C'est la raison pour laquelle on les connait aussi sous l'appellation d'*opérateurs booléens*. Il y en a quatre (tableau 4-4).

Tableau 4-4. Les opérateurs logiques

Opérateur logique	Description
AND	(Et) TRUE si les deux opérandes sont à TRUE
OR	(Ou) TRUE si au moins un des deux opérandes vaut TRUE
XOR	(Ou exclusif) TRUE si seulement un des deux opérandes vaut TRUE
NOT	(Non) TRUE si l'opérande vaut FALSE, FALSE si l'opérande vaut TRUE

L'exemple 4-16 illustre le fonctionnement de ces opérateurs. Remarquez que PHP exige le symbole `!` à la place de `NOT`. De plus, ces opérateurs s'utilisent indifféremment en capitales ou en bas de casse.

Exemple 4-16. Les opérateurs logiques en action

```
<?php
  $a = 1; $b = 0;

  echo ($a AND $b) . "<br>";
  echo ($a or $b)  . "<br>";
  echo ($a XOR $b) . "<br>";
  echo !$a         . "<br>";
?>
```

Cet exemple génère respectivement `NULL`, `1`, `1` et `NULL`, ce qui signifie que seulement les deuxième et troisième conditions sont vraies, c'est-à-dire en pratique qu'elles s'évaluent à `TRUE`. Rappelez-vous que `NULL` signifie «rien» et représente une valeur `FALSE` dans une commande `echo`. L'instruction `AND` ne restitue `TRUE` que si les deux opérandes valent simultanément `TRUE`, or `b$` vaut `0`, ce qui équivaut à `FALSE`. Enfin, la quatrième instruction effectue un `NOT` sur la valeur de `a$`, pour la transformer de `1`, qui équivaut à `TRUE`, en `FALSE`. Pour prolonger l'expérimentation avec ce code, faites varier les valeurs de `a$` et `b$` avec 1 et 0, et voyez les résultats.

Lorsque vous rédigez des programmes en PHP, rappelez-vous que les opérateurs `AND` et `OR` ont une préséance plus faible que les autres versions de ces opérateurs, `&&` et `||`, respectivement. Dans des expressions complexes, il est parfois préférable d'utiliser `&&` et `||` pour cette même raison.

L'opérateur `OR` peut quelquefois provoquer des problèmes inattendus dans les instructions `if`, car le second opérateur n'est pas évalué si le premier est déjà évalué à `TRUE`. Dans l'exemple 4-17, la fonction `getnext` (prendre le suivant) n'est jamais appelée lorsque `$termine` vaut `1`.

Exemple 4-17. Utilisation de l'opérateur OR dans une instruction

```
<?php
  if ($termine == 1 or getnext() == 1) exit;
?>
```

Si vous devez absolument appeler `getnext` dans l'instruction `if`, quelle que soit la valeur de `$termine`, il vaut mieux récrire ce code comme dans l'exemple 4-18.

Exemple 4-18. Récriture de l'instruction if...OR pour garantir l'appel de getnext

```
<?php
  $gn = getnext();

  if ($termine == 1 or $gn == 1) exit;
?>
```

Dans cette version-ci, le code de la fonction `getnext` est exécuté avant le `if` et renvoie une valeur mémorisée dans `$gn`, utilisée ensuite dans l'instruction `if`.

Une autre possibilité consiste à inverser les deux clauses pour faire apparaitre en premier lieu le test sur `getnext` avant celui de `$termine` dans l'instruction `if`.

Le tableau 4-5 illustre toutes les variations des résultats des opérateurs logiques. Ce genre de tableau porte le nom de *table de vérité* des opérateurs. Notez que `!TRUE` donne `FALSE` et que `!FALSE` donne `TRUE`.

Tableau 4-5. Toutes les expressions logiques possibles en PHP

Entrées		Opérateurs et résultats		
a	b	AND	OR	XOR
TRUE	FALSE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	FALSE	FALSE
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	FALSE

Conditions

Les *conditions* modifient le flux d'exécution d'un programme. Elles permettent de poser des questions à propos de certaines choses et les réponses orientent l'exécution selon des voies différentes. Les conditions constituent un des fondements des pages web dynamiques et aussi la première raison d'être de l'utilisation de PHP, parce qu'elles facilitent la génération de résultats différents à chaque affichage d'une page.

Trois types de conditions sans boucle existent : l'instruction `if`, l'instruction `switch` et l'opérateur ternaire `?`. La condition *sans boucle* signifie que les actions amorcées par l'instruction sont exécutées, puis le flux de programme poursuit sa route, tandis que les conditions à boucle, que nous verrons sous peu, exécutent du code plusieurs fois, avant qu'une condition soit atteinte.

Instruction if

Une manière de voir le flux de programme consiste à imaginer une autoroute à sens unique sur laquelle vous roulez. Il s'agit essentiellement d'une ligne droite mais, de temps en temps, vous rencontrez des panneaux de signalisation qui vous indiquent par où aller.

Dans le cas de l'instruction `if`, un panneau de déviation apparait, qui vous indique d'effectuer un détour dans le cas où une certaine condition est vraie (s'il y a un accident plus loin, par exemple), ou `TRUE`. Si c'est le cas, vous détournez votre auto de l'autoroute pour y revenir ensuite et poursuivre votre route. Sinon, si la condition n'est pas vraie, n'est pas `TRUE` (il n'y a pas d'accident), vous ignorez ce détour et continuez votre route normalement (figure 4-1).

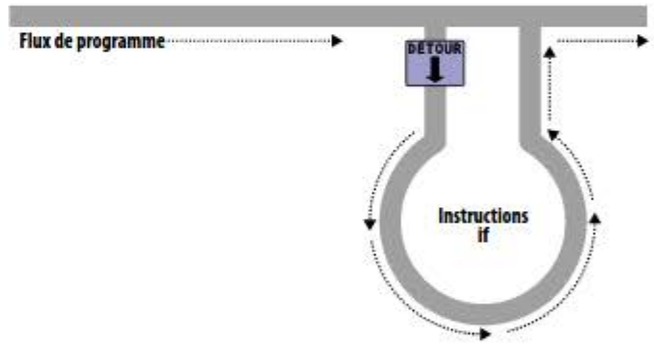


Figure 4-1. Le flux d'exécution du programme s'apparente à une autoroute à simple bande de circulation

Le contenu de la condition du `if` peut être toute expression PHP valide, notamment une égalité, une comparaison, un test de `0` et de `NULL`, et même les valeurs retournées par des fonctions, que ce soient des fonctions intégrées ou celles que vous aurez écrites vous-même.

Les actions à réaliser lorsque la condition du `if` vaut `TRUE` viennent généralement se placer entre deux accolades, `{ }`. S'il ne s'agit que d'une seule instruction, vous pouvez omettre les accolades et la placer sur la même ligne que la condition. Cependant, si vous utilisez toujours les accolades, vous vous éviterez la pénible tâche de traquer des bogues

difficiles à retrouver, comme lorsque vous ajoutez une ligne supplémentaire à une condition, qui ne s'évalue pas à cause d'un manque d'accolades. Remarquez toutefois que, dans le but de réduire l'espace et d'améliorer la clarté, nombre d'exemples du livre ne respectent pas cette suggestion et éludent les accolades pour les instructions simples.

L'exemple 4-19 part de l'hypothèse que vous êtes en fin de mois, que vous avez payé toutes vos factures et que vous effectuez un peu de gestion de votre compte bancaire.

Exemple 4-19. Instruction if avec des accolades

```
<?php
  if ($solde_banque < 100)
  {
    $argent        = 1000;
    $solde_banque  += $argent;
  }
?>
```

Dans cet exemple, vous vérifiez votre solde pour voir s'il est inférieur à 100 (quelle que soit votre devise monétaire, disons des) dollars. Si c'est le cas, vous vous payez vous-même 1 000 dollars et les ajoutez au solde. (Si gagner de l'argent était aussi simple !)

Si le solde bancaire est de 100 dollars ou plus, les instructions conditionnelles sont ignorées et le flux d'exécution du programme saute aux lignes suivantes (non montrées).

Dans ce livre, les accolades ouvrantes se situent toujours sur une nouvelle ligne. Certaines personnes préfèrent la placer à droite, la fin de la ligne de l'expression de condition ; d'autres préfèrent débuter une ligne avec l'accolade ouvrante. C'est une affaire de préférences personnelles et PHP accepte les deux, parce qu'il vous permet de disposer vos espacements (espaces, nouvelles lignes et tabulations) comme vous le voulez. Cependant, votre code sera beaucoup plus facile à relire et à déboguer si vous respectez les mêmes types de retraits pour chaque niveau de condition à l'aide d'une tabulation, ou plus suivant le niveau.

Instruction else

Lorsqu'une condition n'est pas vraie, vous voulez poursuivre vers le flux du code de programme principal mais vous souhaitez aussi parfois exécuter du code en plus, pour faire autre chose que ce que fait la partie `TRUE`. C'est là que l'instruction `else`, qui signifie « sinon », entre en jeu. Grâce à elle, vous pouvez définir une autre déviation sur l'autoroute (figure 4-2).

Dans une instruction `if...else`, l'exécution de la première instruction conditionnelle a lieu si la condition s'évalue à `TRUE`. Sinon, si la condition s'évalue à `FALSE`, c'est la seconde qui s'exécute. Une des deux options doit être exécutée. En aucune circonstance, les deux ne peuvent s'exécuter simultanément, ni aucune des deux. L'exemple 4-20 montre l'utilisation de la structure `if...else`.

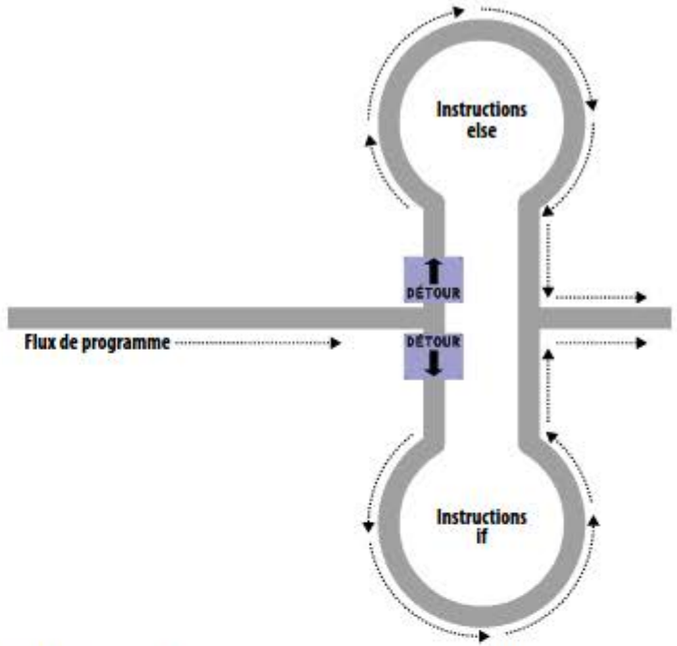


Figure 4-2. L'autoroute a à présent un détour if et un détour else

Exemple 4-20. Instruction if...else avec des accolades

```
<?php
  if ($solde_banque < 100)
  {
    $argent        = 1000;
    $solde_banque += $argent;
  }
  else
  {
    $epargne      += 50;
    $solde_banque -= 50;
  }
?>
```

Dans cet exemple, maintenant que vous avez constaté que vous disposez de 100 $ ou plus à la banque, l'instruction `else` s'exécute, par laquelle vous virez une partie de cet argent vers votre compte d'épargne.

Comme dans le cas de l'instruction `if`, si une instruction `else` ne possède qu'une seule instruction conditionnelle, vous pouvez décider d'éluder les accolades. Les accolades sont toutefois recommandées en tout temps, d'abord parce qu'elles facilitent la compréhension du code, ensuite parce qu'elles permettent d'ajouter facilement de nouvelles instructions par la suite.

Instruction elseif

Certains cas vous amènent à définir plusieurs possibilités qui peuvent se produire, en fonction d'une séquence de conditions. L'instruction `elseif` est faite pour cela. Elle correspond à « sinon, si », c'est-à-dire que c'est un `else` suivi d'un `if` et d'une expression conditionnelle, puis du code conditionnel. L'exemple 4-21 illustre une construction `if...elseif...else` complète.

Exemple 4-21. Instruction if...elseif...else avec des accolades

```
<?php
  if ($solde_banque < 100)
  {
    $argent        = 1000;
    $solde_banque += $argent;
  }
  elseif ($solde_banque > 200)
  {
    $epargne      += 100;
    $solde_banque -= 100;
  }
  else
  {
    $epargne      += 50;
    $solde_banque -= 50;
  }
?>
```

Dans cet exemple, une instruction `elseif` est venue s'insérer entre les instructions `if` et `else` de l'exemple précédent. Elle vérifie si le solde bancaire dépasse 200 $ et, si c'est le cas, décide que vous avez les moyens d'épargner 100 $ ce mois-ci.

Bien que je pousse l'analogie un peu trop loin, vous pouvez imaginer cela comme un ensemble de détours à plusieurs voies (figure 4-3).

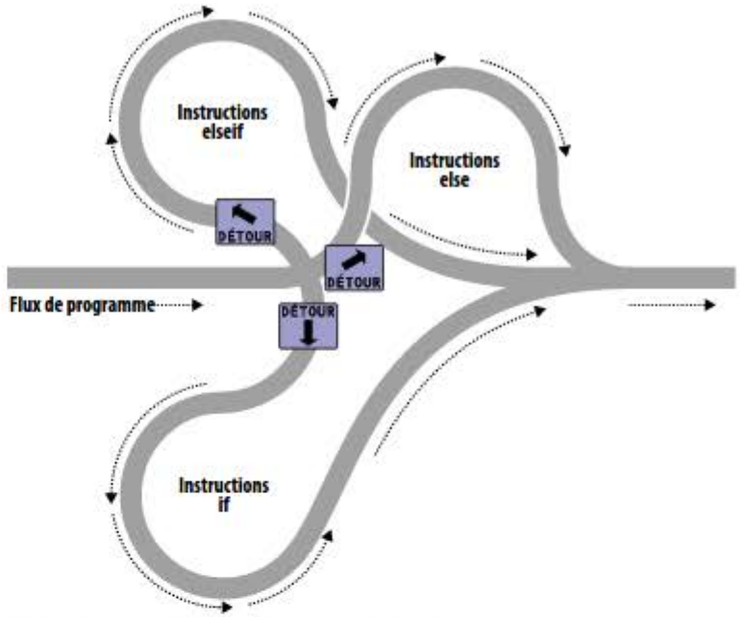


Figure 4-3. L'autoroute avec les détours if, elseif et else

L'instruction `else` clôture une instruction soit `if...else`, soit `if...elseif...else`. Vous pouvez ignorer le `else` final s'il n'est pas nécessaire mais vous ne pouvez pas placer de `else` avant un `elseif`, ni placer de `elseif` avant une instruction `if`.

Le nombre de `elseif` qui se succèdent est à votre discrétion mais, à mesure que le nombre d'instructions `elseif` croit, vous auriez probablement intérêt à mettre en place une instruction `switch`, si elle vous convient. Nous l'examinons à la section suivante.

Instruction switch

L'instruction `switch` s'avère utile dans le cas où une variable ou le résultat d'une expression peut prendre plusieurs valeurs, qui déclenchent chacune une action différente.

Prenons l'exemple d'un système de menus dirigé par PHP qui passe une simple chaine au code de menu principal selon la demande de l'utilisateur. Supposons que les options possibles soient Accueil, À propos, Nouvelles, Se connecter et Liens, et que nous définissons la variable `$page` avec une de ces valeurs, en fonction de l'entrée de l'utilisateur.

Si nous écrivons ce code à l'aide d'une instruction `if...elseif...else`, nous aboutissons à un code du genre de celui de l'exemple 4-22.

Exemple 4-22. Instruction if...elseif...else avec plusieurs lignes

```
<?php
  if     ($page == "Accueil")       echo "Vous avez choisi Accueil";
  elseif ($page == "À propos")      echo "Vous avez choisi À propos";
  elseif ($page == "Nouvelles")     echo "Vous avez choisi Nouvelles";
  elseif ($page == "Se connecter") echo "Vous avez choisi Se connecter";
  elseif ($page == "Liens")         echo "Vous avez choisi Liens";
?>
```

Avec une instruction switch, ce code prend l'allure de l'exemple 4-23.

Exemple 4-23. Instruction switch

```
<?php
  switch ($page)
  {
    case "Accueil":
        echo "Vous avez choisi Accueil";
        break;
    case "À propos":
        echo "Vous avez choisi À propos";
        break;
    case "Nouvelles":
        echo "Vous avez choisi Nouvelles";
        break;
    case "Se connecter":
        echo "Vous avez choisi Se connecter";
        break;
    case "Liens":
        echo "Vous avez choisi Liens";
        break;
  }
?>
```

La variable `$page` n'est mentionnée qu'une fois au début de l'instruction `switch`, à la suite de quoi la commande `case` vérifie les correspondances. Lorsqu'une correspondance est trouvée, l'instruction conditionnelle associée est exécutée. Dans un programme réel, il est clair que vous auriez ici du code pour afficher ou aller à la page adéquate, au lieu de simplement afficher le choix de l'utilisateur.

Dans les commandes `case` de l'instruction `switch`, n'utilisez pas d'accolades car elles commencent par un deux-points et se terminent par l'instruction `break`. En revanche, la totalité du corps de l'instruction `switch`, avec les différentes commandes `case`, doit être entourée d'accolades.

Rupture

Lorsque vous voulez interrompre l'exécution d'une instruction `switch`, parce qu'une des options a été rencontrée et ses instructions exécutées, placez une commande `break`. Cette commande indique à PHP de sortir de la structure du `switch` et de sauter à la première instruction qui la suit.

Si nous ne plaçons pas de commandes `break` dans l'exemple 4-23, alors le contenu des cinq blocs `case` s'exécute en enfilade lorsque le `case` de l'Accueil s'évalue à `TRUE`. Si `$page` a la valeur `"Nouvelles"`, alors toutes les commandes `case` suivantes s'exécutent à partir du `case` correspondant. Ceci est délibéré et permet quelles astuces de programmation avancée mais, d'une manière plus générale, retenez que vous devez en principe placer un `break` à la fin du bloc de code de chaque `case`. En pratique, l'oubli de l'instruction `break` constitue une erreur assez courante.

Action par défaut

Une exigence habituelle des instructions `switch` impose de rediriger l'exécution vers une action par défaut si aucune des conditions `case` n'est satisfaite. Ainsi, dans le cas du code de menu de l'exemple 4-23, vous devriez ajouter le code de l'exemple 4-24, immédiatement avant l'accolade finale.

Exemple 4-24. Instruction par défaut à ajouter à l'exemple 4-23

```
default:
    echo "Choix non reconnu";
    break;
```

Bien qu'ici la commande `break` ne soit pas indispensable, puisque cette sous-instruction finale est la dernière et que l'exécution se poursuit automatiquement après l'accolade finale, si vous décidez un jour de déplacer l'instruction `default` plus haut dans le code, elle devrait en contenir une pour empêcher le flux d'exécution d'entrer dans les instructions suivantes. En général, la pratique la plus sure consiste donc à toujours inclure la commande `break`.

Syntaxe alternative

Si vous préférez, vous pouvez remplacer la première accolade de l'instruction `switch` par un deux-points et l'accolade fermante finale par la commande `endswitch` (exemple 4-25). Cette pratique n'est toutefois pas usuelle et n'est mentionnée ici qu'au cas où vous liriez du code d'une autre personne.

Exemple 4-25. Syntaxe alternative de l'instruction switch

```
<?php
  switch ($page):
    case "Accueil":
        echo "Vous avez choisi Accueil";
        break;

    // etc.

    case "Liens":
        echo "Vous avez choisi Liens";
        break;
  endswitch;
?>
```

Opérateur ?

Un moyen d'éviter la verbosité (autrement dit, les « blablabla » inutiles) des instructions `if` et `else` consiste à faire appel à l'opérateur ternaire, `?`, plus compact. Celui-ci est plutôt inhabituel dans la mesure où il prend trois opérandes, au lieu de deux, plus habituels.

Nous l'avons déjà évoqué brièvement au chapitre 3, lors de l'examen de la différence entre les instructions `print` et `echo`, dans un exemple de type d'opérateur qui fonctionne bien avec `print` mais pas avec `echo`.

L'opérateur `?` reçoit une expression à évaluer, ainsi que deux instructions à exécuter : l'une si l'évaluation de l'expression donne `TRUE`, l'autre si elle donne `FALSE`. L'exemple 4-26 illustre du code utilisable pour écrire un avertissement relatif au niveau de carburant d'un véhicule sur son tableau de bord numérique.

Exemple 4-26. Utilisation de l'opérateur ?

```
<?php
  echo $carburant <= 5 ? "Faire le plein" : "Carburant suffisant";
?>
```

Cette instruction signifie que, s'il reste cinq litres ou moins de carburant (autrement dit si `$carburant` vaut 5 ou moins), la chaine `Faire le plein` est renvoyée à l'instruction `echo`, sinon, la chaine `Carburant suffisant` est renvoyée. L'exemple 4-27 illustre l'affectation de la valeur renvoyée par l'instruction `?` à une variable.

Exemple 4-27. Affectation du résultat conditionnel de l'opérateur ? à une variable

```
<?php
  $suffisant = $carburant <= 5 ? FALSE : TRUE;
?>
```


Cette fois, `$suffisant` reçoit la valeur `TRUE` seulement s'il reste plus de cinq litres de carburant, sinon elle reçoit la valeur `FALSE`.

Si l'opérateur `?` vous paraît un peu compliqué, libre à vous de le remplacer par une instruction `if...else`, mais vous avez tout intérêt à vous familiariser avec lui parce que vous le verrez dans le code d'autres personnes. L'instruction est quelquefois difficile à lire parce qu'elle mélange plusieurs occurrences d'une même variable. Ainsi, une formulation telle que la suivante n'est pas rare dans les programmes :

```
$enregistre = $enregistre >= $nouveau ? $enregistre : $nouveau;
```

Une décomposition de cette instruction permet de mieux comprendre ce code :

```
$enregistre =                       // Définit la valeur de $enregistre à...
              $enregistre >= $nouveau // Comparer $enregistre à $nouveau
          ?                         // Oui, la comparaison est vraie...
              $enregistre           // ... alors affecter la valeur de $enregistre
          :                         // Non, la comparaison est fausse...
              $nouveau;             // ... alors affecter la valeur de $nouveau
```

Il s'agit là d'un moyen concis de suivre la plus grande valeur rencontrée au fur et à mesure de la progression d'un programme. Vous mémorisez la plus grande valeur dans `$enregistre` et la comparez à `$nouveau`, chaque fois que vous obtenez une nouvelle valeur. Les programmeurs familiers avec l'opérateur `?` le considèrent plus convivial que l'instruction `if` pour de telles comparaisons courtes. Quand il ne sert pas à écrire du code compact, il intervient habituellement pour prendre des décisions à la volée, comme pour vérifier qu'une variable a bien été définie, avant de la passer à une fonction.

Boucles

Un des grands intérêts des ordinateurs réside dans leur capacité à répéter inlassablement et rapidement des tâches de calcul. Il arrive souvent qu'un programme doive répéter encore et encore la même séquence de code, jusqu'à ce que quelque chose de précis se produise, comme l'entrée d'une valeur par l'utilisateur ou l'arrivée à la fin naturelle de la boucle. Les structures de boucles de PHP offrent le moyen parfait de réaliser cela.

Pour comprendre comment cela fonctionne, examinez la figure 4-4. Pour reprendre l'analogie de l'autoroute qui nous a permis d'illustrer les instructions `if`, ici, le détour comporte en plus une section de boucle dont, lorsqu'un véhicule y entre, il ne peut sortir que sous certaines conditions adéquates du programme.

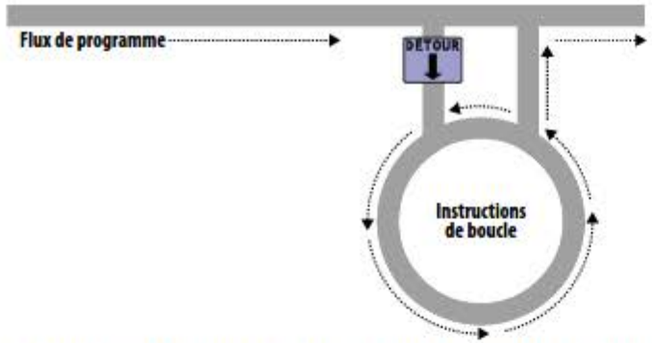


Figure 4-4. Représentation d'une boucle comme une portion d'autoroute de programme

Boucle while

Reprenons l'exemple du tableau de bord numérique évoqué à l'exemple 4-26 pour le convertir en une boucle qui vérifie en permanence le niveau de carburant pendant la conduite, avec une boucle `while` (exemple 4-28).

Exemple 4-28. Une boucle while

```
<?php
  $carburant = 50;

  while ($carburant > 5)
  {
        // Continuer de rouler …
        echo "Carburant suffisant";
  }
?>
```

En pratique, au lieu de ce message, nous afficherions plutôt un voyant vert qu'un texte, mais l'important est de placer l'indicateur positif de niveau de carburant dans la boucle `while`. Au fait, si vous testez cet exemple vous-même, remarquez qu'il affiche la chaine de texte à l'infini, jusqu'à ce que vous cliquiez sur le bouton de fermeture du navigateur.

Comme pour les instructions `if`, des accolades entourent les instructions au sein des instructions `while`, sauf s'il n'y en a qu'une.

L'exemple 4-29 illustre une autre boucle `while` qui affiche la table de multiplication de 12.

Exemple 4-29. Une boucle while pour afficher la table de multiplication de 12

```
<?php
  $compteur = 1;

  while ($compteur <= 12)
  {
    echo "$compteur fois 12 égale " . $compteur * 12 . "<br>";
    ++$compteur;
  }
?>
```

Ici, la variable `$compteur` est initialisée à `1`, puis la boucle `while` débute avec l'expression de comparaison `$compteur <= 12`. Cette boucle continue son exécution jusqu'à ce que la variable dépasse `12`. Les résultats de ce code donnent :

```
1 fois 12 égale 12
2 fois 12 égale 24
3 fois 12 égale 36
et ainsi de suite…
```

À l'intérieur de la boucle a lieu l'impression de la chaine obtenue à partir de la valeur du `$compteur`, suivie de `$compteur` multiplié par `12`, puis d'une balise `<br>` pour imposer un retour à la ligne et améliorer la présentation. La ligne suivante incrémente `$compteur`, puis l'accolade fermante indique à PHP de revenir au début de la boucle.

À ce stade, le test de `$compteur` vérifie s'il est plus grand que `12`. Il ne l'est pas, car il a cette fois la valeur `2`. Au terme de 11 tours de boucle, il a la valeur `13`. Lorsque c'est le cas, le code au sein de la boucle est ignoré et l'exécution se poursuit après l'accolade fermante de la boucle, soit ici, à la fin du programme.

Si l'instruction `++$compteur` (qui aurait pu ici s'écrire indifféremment `$compteur++`) n'était pas présente, cette boucle suivrait le même chemin que le premier exemple de boucle précédent : elle ne s'arrêterait jamais et afficherait à l'infini le résultat de `1 * 12`.

Il existe une façon plus claire d'écrire cette boucle, que vous apprécierez certainement. Examinez l'exemple 4-30.

Exemple 4-30. Version réduite de l'exemple 4-29

```
<?php
  $compteur = 0;

  while (++$compteur <= 12)
    echo "$compteur fois 12 égale " . $compteur * 12 . "<br>";
?>
```

Cet exemple montre qu'il est possible de retirer l'instruction `++$compteur` du corps de la boucle pour la placer directement dans l'expression conditionnelle de la boucle. Dans ce cas-là, PHP rencontre la variable `$compteur` au début de chaque itération de la

boucle et, comme elle est préfixée de l'opérateur d'incrémentation, lui ajoute d'abord 1, pour ensuite la comparer à la valeur 12. Remarquez que, cette fois, la variable $compteur doit être initialisée à 0, non plus 1, parce qu'elle est incrémentée dès l'entrée dans la boucle. Si vous la laissez à 1, seuls les résultats des valeurs 2 à 12 s'affichent.

Boucle do...while

Petite variante de la boucle while, la boucle do...while s'impose dans le cas où il est nécessaire d'exécuter un bloc de code au moins une fois, pour ensuite seulement vérifier une condition. L'exemple 4-31 montre une version modifiée du code de la table de multiplication de 12, pour emprunter cette structure.

Exemple 4-31. Une boucle do...while pour afficher la table de multiplication par 12

```
<?php
  $compteur = 1;

  do
        echo "$compteur fois 12 égale " . $compteur * 12 . "<br>";
  while (++$compteur <= 12);
?>
```

Remarquez d'abord que nous reprenons l'initialisation de $compteur à 1, et non plus à 0, parce que le bloc de code est exécuté immédiatement, sans possibilité d'incrémenter la variable. Ceci mis à part, le code est fort semblable au précédent.

Bien entendu, si le corps de la boucle contient plus d'une instruction, n'oubliez pas de les entourer d'accolades, comme dans l'exemple 4-32.

Exemple 4-32. Expansion de l'exemple 4-31 pour utiliser des accolades

```
<?php
  $compteur = 1;

  do {
        echo "$compteur fois 12 égale " . $compteur * 12;
        echo "<br>";
  } while (++$compteur <= 12);
?>
```

Boucle for

Le dernier type d'instruction de boucle, la boucle for, est aussi le plus puissant car il conjugue les possibilités de définir les variables à l'entrée dans la boucle, avec le test de conditions à chaque itération de la boucle et avec la modification des variables après chaque itération.

Exemple 4-33. Génération de la table de multiplication de 12 à l'aide d'une boucle for

```
<?php
  for ($compteur = 1 ; $compteur <= 12 ; ++$compteur)
        echo "$compteur fois 12 égale " . $compteur * 12 . "<br>";
?>
```

Le code en est réduit à une seule instruction for contenant une simple instruction conditionnelle. Examinons-le en détail. Chaque instruction for prend trois paramètres :

- une expression d'initialisation ;
- une expression conditionnelle ;
- une expression de modification.

Ceux-ci sont séparés par des points virgules selon la syntaxe for (*expr1* ; *expr2* ; *expr3*). Au début de la première itération de la boucle, l'expression d'initialisation est exécutée, soit dans l'exemple, $compteur est initialisé à la valeur 1. Ensuite, à chaque itération de la boucle, l'expression de condition (ici, $compteur <= 12) est testée et l'entrée dans le corps de la boucle n'a lieu que si cette condition s'évalue à TRUE. Enfin, à l'issue de chaque itération, l'expression de modification est exécutée. Dans l'exemple, la variable $compteur est incrémentée.

Cette structure supprime toute exigence d'un contrôle de boucle dans le corps de celle-ci, pour ne laisser dans le corps que les instructions que vous voulez exécuter dans la boucle.

Rappelez-vous de placer des accolades pour entourer le corps de la boucle si celui-ci en contient plus d'une, comme illustré dans l'exemple 4-34.

Exemple 4-34. La boucle for de l'exemple 4-33 avec des accolades

```
<?php
  for ($compteur = 1 ; $compteur <= 12 ; ++$compteur)
  {
    echo "$compteur fois 12 égale " . $compteur * 12;
    echo "<br>";
  }
?>
```

Maintenant, quand faut-il préférer un certain type de boucle for ou while, l'un par rapport à l'autre ? La boucle for est clairement conçue autour d'une seule valeur qui change régulièrement, soit habituellement, une valeur qui s'incrémente, comme quand vous passez une liste de choix d'utilisateur et voulez traiter chaque choix en retour. Mais vous pouvez transformer la variable de la manière que vous souhaitez. Une forme plus complexe de l'instruction for permet même d'effectuer plusieurs opérations dans chacun des trois paramètres, comme suit :

```
for ($i = 1, $j = 1 ; $i + $j < 10 ; $i++, $j++)
{
  // ...
}
```

Cette formulation est complexe et peu recommandée aux débutants. L'important est de distinguer les virgules des points-virgules qui séparent les trois paramètres. Dans chaque paramètre, plusieurs instructions peuvent exister mais vous devez les séparer par des virgules. Ainsi, dans l'exemple précédent, les premier et troisième paramètres contiennent chacun deux instructions :

```
$i = 1, $j = 1 // Initialiser $i et $j
$i + $j < 10   // Condition de terminaison
$i++ , $j++    // Modifier $i et $j au terme de chaque itération
```

Donc, le principal élément à retenir de cet exemple, c'est que vous devez séparer les trois sections de paramètres par des points-virgules et non des virgules, tandis que les virgules séparent les instructions au sein d'un même paramètre.

Enfin, quand faut-il préférer une instruction `while` à une instruction `for` ? Lorsque la condition ne dépend pas d'une variable simple, au changement régulier. Par exemple, lorsque vous devez surveiller la présence d'un événement fortuit, comme vérifier une entrée ou la présence d'une erreur et clôturer la boucle lorsqu'elles apparaissent, utilisez plutôt une instruction `while`.

Rupture d'une boucle

Tout comme l'instruction `break` permet de sortir d'une instruction `switch`, il est possible de sortir d'une boucle à l'aide de cette même commande. Cette étape peut être indispensable lorsque, par exemple, une des instructions de la boucle renvoie une erreur et que la boucle ne peut continuer de s'exécuter proprement.

Un cas d'espèce de ce genre apparait lorsque l'écriture dans un fichier renvoie une erreur, par exemple parce que le disque est plein (exemple 4-35).

Exemple 4-35. Écriture dans un fichier à partir d'une boucle for et capture d'erreur

```
<?php
  $fp = fopen("texte.txt", 'wb');

  for ($j = 0 ; $j < 100 ; ++$j)
  {
        $ecrit = fwrite($fp, "donnée");

        if ($ecrit == FALSE) break;
  }

  fclose($fp);
?>
```


Voici probablement l'exemple le plus complexe de programme que vous ayez vu jusqu'ici mais, rassurez-vous, vous êtes prêt à l'apprendre. Nous verrons les commandes de gestion de fichier dans un chapitre ultérieur mais, pour l'heure, tout ce que vous devez retenir, c'est que la première ligne ouvre le fichier *texte.txt* en écriture et en mode binaire, qu'elle renvoie un pointeur vers ce fichier dans la variable `$fp`, qui sert ensuite de référence au fichier ouvert.

La boucle entre en jeu pour itérer 100 fois (de 0 à 99) et écrire autant de fois la chaine `"donnée"` dans ce fichier. Après chaque écriture, la variable `$ecrit` reçoit la valeur retournée par la fonction `fwrite`, qui représente le nombre de caractères correctement écrits. S'il se produit une erreur à ce niveau, `fwrite` retourne la valeur `FALSE`.

Le comportement de `fwrite` permet d'aisément tester le contenu de la variable `$ecrit`, pour vérifier si elle est réglée à `FALSE` et, en conséquence, de provoquer la rupture de la boucle avec `break`, et passer à l'instruction suivante, qui ferme le fichier.

Si vous cherchez à améliorer le code, la ligne

```
if ($ecrit == FALSE) break;
```

peut s'écrire de manière plus concise avec l'opérateur `NOT`, comme suit :

```
if (!$ecrit) break;
```

La paire d'instructions du corps de la boucle peut encore se réduire en une seule instruction :

```
if (!fwrite($fp, "donnée")) break;
```

La commande `break` est encore plus puissante que vous ne l'imaginez car, si du code contient plus d'un niveau de profondeur, dont vous devez sortir, il suffit de faire suivre la commande `break` d'un nombre indiquant le nombre de niveaux dont il faut sortir, comme ceci :

```
break 2;
```

Instruction continue

L'instruction `continue` agit un peu comme l'instruction `break`, sauf qu'elle indique à PHP d'arrêter de traiter le corps de la boucle et de passer directement à l'itération suivante. Donc, au lieu de quitter la totalité de la boucle, PHP ne quitte que l'itération courante.

Cette approche est utile dans des cas où il est inutile de poursuivre l'exécution de la boucle courante et que vous souhaitez économiser des cycles du processeur ou éviter une erreur, en sautant directement à l'itération suivante de la boucle. L'exemple 4-36 illustre l'utilisation de l'instruction `continue` pour éviter une erreur de division par zéro lorsque la variable `$j` a la valeur `0`.

Exemple 4-36. Interception d'erreur de division par zéro à l'aide de continue

```
<?php
  $j = 10;

  while ($j > -10)
  {
    $j--;

    if ($j == 0) continue;

    echo (10 / $j) . "<br>";
  }
?>
```

Ce code affiche le calcul de `10` divisé par `$j` pour toutes les valeurs de `$j` comprises entre `10` et `-10`, sauf quand `$j` vaut `0`. Quand `$j` vaut `0`, l'instruction `continue` provoque le saut immédiat à l'itération suivante.

Conversion implicite et explicite de type (cast)

PHP est un langage de typage faible qui permet de déclarer une variable et son type simplement en l'utilisant. Il convertit automatiquement les valeurs d'un type à un autre lorsque c'est nécessaire. Il s'agit de la *conversion implicite de type* (*implicit casting*).

Cependant, la conversion implicite de PHP ne correspond pas toujours à ce que vous souhaitez. Dans l'exemple 4-37, remarquez que les opérandes de la division sont des entiers (*integer*). Par défaut, PHP convertit le résultat en nombre à virgule flottante pour donner la valeur la plus précise possible, 4,66 en l'occurrence.

Exemple 4-37. Cette expression renvoie un nombre à virgule flottante

```
<?php
  $a = 56;
  $b = 12;
  $c = $a / $b;

  echo $c;
?>
```

Et si nous avions préféré obtenir un entier pour `$c` ? Plusieurs manières permettent d'obtenir le résultat voulu, dont l'une consiste à imposer au résultat de `$a/$b` une conversion de type en une valeur entière à l'aide du type de conversion entier (`int`), comme suit :

```
$c = (int) ($a / $b);
```

Ceci s'appelle la *conversion explicite de type*. Remarquez que, pour garantir que ce soit la valeur de la totalité de l'expression qui subisse la conversion de type, nous devons placer

des parenthèses autour de la division avec ses opérandes, sinon, seule la variable `$a` serait convertie en entier, ce qui ne servirait à rien, puisque la division par `$b` générerait encore toujours un nombre à virgule flottante.

Le tableau 4-6 énumère les types vers lesquels les conversions explicites de type sont possibles. Vous pouvez généralement éviter les conversions de type en appelant une des fonctions intégrées de PHP. Par exemple, pour obtenir une valeur entière, utilisez la fonction `intval`. Comme dans bien des sections du livre, celle-ci est là essentiellement pour vous permettre de comprendre du code rédigé par des tiers, que vous risquez de rencontrer.

Tableau 4-6. Conversions de type (cast) de PHP

Type cible	Description
`(int) (integer)`	Conversion en un entier, par suppression du point décimal
`(bool) (boolean)`	Conversion en un booléen
`(float) (double) (real)`	Conversion en un nombre à virgule (point) flottante
`(string)`	Conversion en une chaine de caractères
`(array)`	Conversion en un tableau
`(object)`	Conversion en un objet

Liaison dynamique en PHP

Comme PHP est un langage de programmation et que les résultats qu'il produit peuvent différer complètement pour chaque utilisateur, il est possible de générer un site web tout entier à partir d'une seule page web en PHP. Chaque fois que l'utilisateur clique sur quelque chose, les détails sont envoyés à la même page web, qui décide de ce qu'il faut faire ensuite selon les différents témoins de connexion (*cookies*) et autres détails de session enregistrés.

Mais s'il est possible de construire tout un site de cette manière, ce n'est pas recommandé, car le code source croit considérablement et devient rapidement ingérable, puisqu'il doit tenir compte de toutes les actions possibles de l'utilisateur.

Il faut plutôt éclater le développement du site en différentes parties. Par exemple, une partie traite l'inscription pour accéder au site, avec les vérifications nécessaires pour valider une adresse de courrier électronique, la détermination de si un nom d'utilisateur a déjà été pris, et ainsi de suite.

Un deuxième module pourrait bien accepter les connexions des utilisateurs avant de les traiter en dehors de la partie principale du site. Ensuite, vous auriez un module de messagerie avec la possibilité pour les utilisateurs de déposer des commentaires, un module contenant des liens et d'autres informations utiles, un autre pour permettre le versement d'images, et d'autres encore.

Dès que vous disposez d'un moyen pour suivre un utilisateur tout au long du parcours de votre site par l'entremise de cookies ou de variables de session (que nous examinerons de plus près dans des chapitres ultérieurs), vous pouvez découper votre site web en sections raisonnables de code PHP, toutes autonomes, et donc vous faciliter la tâche pour développer chaque nouvelle fonctionnalité et entretenir les anciennes.

La liaison dynamique dans toute sa splendeur

La plateforme de blogue WordPress (figure 4-5) est certainement l'une des applications pilotées en PHP les plus prisées sur le web d'aujourd'hui. Selon votre point de vue de blogueur ou de simple lecteur de blogue, vous n'avez peut-être pas pris conscience que chaque section principale est gérée par son propre fichier PHP et que toute une armada de fonctions génériques partagées sont placées dans des fichiers distincts pour les inclure dans les pages PHP principales au gré des nécessités.

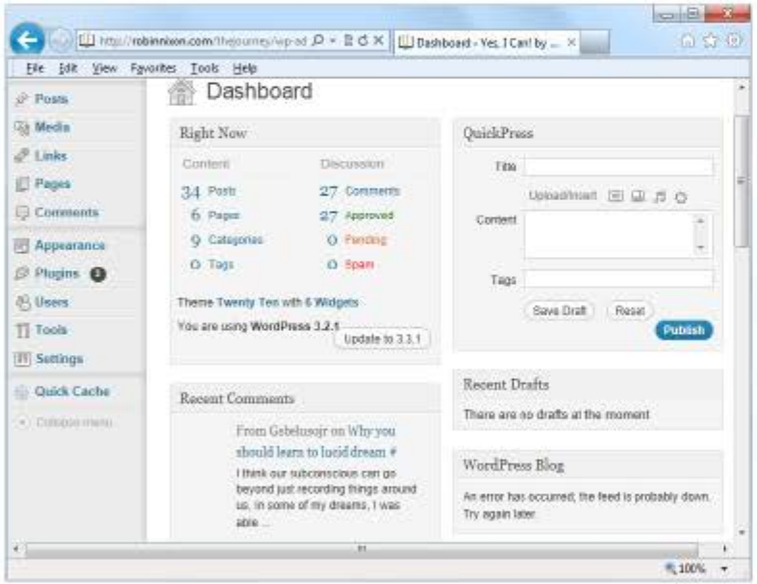


Figure 4-5. La plateforme de blogue WordPress est écrite en PHP

L'unité de l'ensemble de la plateforme est assurée par un suivi de session en coulisses, de sorte que vous savez plus ou moins quand vous transitez d'une sous-section à une autre. Par conséquent, en tant que développeur, si vous voulez affiner WordPress, il est facile de trouver le fichier particulier dont vous avez besoin, de le modifier, de le tester et de le déboguer sans dénaturer ce qui figure autour, puisque les autres portions du programme n'y sont pas liées. La prochaine fois que vous utilisez WordPress, surveillez la barre d'adresse de votre navigateur, surtout si vous gérez un blogue, et vous remarquerez la présence de quelques-uns des différents fichiers qu'il utilise.

Ce chapitre a détaillé nombre de notions fondamentales, de sorte que, dès à présent, vous devriez être en mesure de rassembler le tout pour rédiger vos premiers petits programmes personnels en PHP. Avant de vous livrer à cet exercice et de passer au chapitre suivant sur les fonctions et les objets, testez vos nouvelles connaissances en répondant aux questions suivantes.

Questions

1. Quelles valeurs sous-jacentes réelles se cachent derrière `TRUE` et `FALSE` ?
2. Quelles sont les deux formes les plus simples d'expressions ?
3. Quelle est la différence entre les opérateurs unaires, binaires et ternaire ?
4. Quel est le meilleur moyen d'imposer votre propre préséance aux opérateurs ?
5. Que signifie l'*associativité d'un opérateur* ?
6. Dans quelles circonstances utilisez-vous l'opérateur `===` d'identité ?
7. Nommez trois types d'instructions conditionnelles.
8. Quelle commande utilisez-vous pour éluder la suite de l'itération courante pour passer à la suivante ?
9. Pourquoi dit-on que la boucle `for` est plus puissante que la boucle `while` ?
10. Comment les instructions `if` et `while` interprètent-elles les expressions conditionnelles de types de données différents ?

Retrouvez les réponses du chapitre 4 dans l'annexe A.

CHAPITRE 5

Fonctions et objets en PHP

Parmi les attentes fondamentales exprimées à propos d'un langage de programmation, il y a la possibilité de stocker des données quelque part, un moyen pour diriger le flux d'exécution d'un programme et un certain nombre d'éléments tels que l'évaluation d'expression, la gestion de fichier et l'affichage de texte. PHP a tout cela mais aussi des outils comme les `else` et `elseif` pour se faciliter la tâche. Même avec tout cela dans notre trousse à outils, la programmation peut s'avérer maladroite et fastidieuse, surtout s'il faut récrire des portions de code très semblables chaque fois que nous en avons besoin.

C'est à ce stade que les fonctions et les objets entrent en jeu. Comme vous le devinez, une fonction est un ensemble d'instructions qui remplissent une certaine fonction et, accessoirement, renvoie une valeur. Vous extrayez une section de code que vous avez utilisée plus d'une fois, la placez dans une fonction et appelez la fonction par son nom là où vous en avez besoin dans le reste du programme.

Les fonctions présentent de nombreux avantages par rapport au code à la volée, contigu. Par exemple, elles :

- impliquent moins de frappes au clavier ;
- réduisent les risques d'erreurs de syntaxe et autres erreurs de programmation ;
- réduisent le temps de chargement des fichiers de programme ;
- restreignent le temps d'exécution, parce que chaque fonction est compilée une seule fois, quel que soit le nombre de ses appels ;
- acceptent des arguments et peuvent donc s'adapter aussi bien à la généralité qu'à des cas spéciaux.

Les objets prolongent encore plus avant ce concept. Un *objet* reprend une ou plusieurs fonctions, ainsi que les données qu'elles utilisent, en une seule et unique structure appelée *classe*.

Ce chapitre explique tout ce qu'il y a à connaitre à propos des fonctions, de leur définition et leur appel jusqu'au passage d'arguments dans un sens et dans l'autre. Sur base de ces connaissances, vous allez créer des fonctions et les utiliser dans vos propres objets (où ces fonctions prennent le nom de *méthodes*).

Fonctions en PHP

PHP fournit lui-même des centaines de fonctions intégrées prêtes à l'emploi, ce qui en fait un langage très riche. Pour utiliser une fonction, il suffit de l'appeler par son nom. Ainsi, vous avez déjà vu la fonction `print` à l'œuvre :

```
print("Ceci est une pseudo-fonction");
```

Les parenthèses indiquent à PHP que vous faites référence à une fonction, sinon il pourrait croire qu'il s'agit d'une constante. Vous risquez de recevoir un avertissement comme celui-ci :

```
Notice: Use of undefined constant fname - assumed 'fname'
```

Suivi de la chaine de texte `fname`, il laisse supposer que vous avez dû vouloir placer une chaine littérale dans le code. (Les choses deviennent vraiment confuses si votre code comporte une constante nommée `fname`, auquel cas, PHP en affiche la valeur.)

À vrai dire, `print` est une pseudo-fonction, appelée communément une construction. La différence par rapport à une fonction se situe dans le fait que vous pouvez omettre les parenthèses :

```
print "print n'exige pas de parenthèses".
```

En revanche, vous devez placer des parenthèses après toute fonction que vous appelez, même si elles sont vides (c'est-à-dire que vous ne lui passez aucun paramètre).

Les fonctions peuvent prendre un nombre quelconque d'arguments, y compris aucun. Par exemple, la fonction `phpinfo`, illustrée ci-après, affiche un grand nombre d'informations à propos de l'installation courante de PHP et n'exige aucun argument. La figure 5-1 illustre le résultat de l'appel de cette fonction.

```
phpinfo();
```

La fonction `phpinfo` est extrêmement pratique et utile pour obtenir des informations à propos de l'installation courante de PHP, mais ces informations sont aussi très utiles aux éventuels pirates. Par conséquent, ne laissez jamais d'appel à cette fonction trainer dans du code destiné à la production et au web.

Figure 5-1. La sortie produite par la fonction phpinfo intégrée de PHP.

L'exemple 5-1 montre quelques fonctions intégrées qui attendent un ou plusieurs arguments.

Exemple 5-1. Trois fonctions de chaines de caractères

```
<?php
    echo strrev(" .suot ,ruojnoB"); // Chaine inversée
    echo str_repeat("Hip ", 3);     // Répétition de chaine
    echo strtoupper("hourra!");     // Chaine en capitales
  ?>
```

L'exemple exploite trois fonctions de chaines pour afficher le résultat suivant :

```
Bonjour, tous. Hip Hip Hip HOURRA!
```

Vous constatez que la fonction `strrev` a inversé l'ordre des caractères dans la chaine, `str_repeat` a répété trois fois la chaine `"Hip "` (le nombre de fois apparait en second argument) et `strtoupper` a converti `"hourra"` en capitales.

Définir une fonction

La syntaxe générale d'une définition de fonction est la suivante :

```
function nom_de_fonction ([paramètre [, ...]])
{
  // Instructions
}
```

La première ligne de la syntaxe indique les éléments suivants :

- Une définition commence par le mot `function`.
- Suit un nom, qui doit commencer par une lettre ou un caractère de soulignement, suivi d'un nombre quelconque de lettres, de chiffres et de soulignements.
- Les parenthèses sont obligatoires.
- Le paramètre, ou les paramètres séparés par des virgules, sont facultatifs. Les crochets ne font pas partie de l'instruction mais signifient par convention que les paramètres qu'ils entourent peuvent être ignorés.

Les noms de fonctions ne sont pas sensibles à la casse, de sorte que les chaines suivantes font référence indifféremment à la même fonction `print` : `PRINT`, `Print` ou `PrInT`.

L'accolade ouvrante marque le début des instructions à exécuter lors de l'appel de la fonction, tandis qu'une accolade correspondante ferme le bloc de code. Ces instructions peuvent contenir une ou plusieurs instructions `return`, qui imposent à la fonction l'arrêt de son exécution et retournent au code appelant. Si une valeur est placée à côté de l'instruction `return`, le code appelant peut la récupérer, comme nous allons le voir.

Retourner une valeur

Examinons une fonction simple qui convertit le nom complet d'une personne en bas de casse (caractères minuscules), puis met en capitale la première lettre de chaque nom et prénom.

Nous avons déjà vu la fonction intégrée de PHP `strtoupper` dans l'exemple 5-1. Pour notre fonction, nous allons tirer parti de sa contrepartie, `strtolower` :

```
$bas_de_casse = strtolower("Une Chaine quelConque avec même de la Ponctuation");
echo $bas_de_casse;
```

Le résultat de cette expérience donne :

```
une chaine quelconque avec même de la ponctuation
```

Les noms en bas de casse manquent cependant d'élégance. Nous voulons que la première lettre de chaque mot soit mise en capitale. (Nous ne nous préoccupons pas des cas subtils comme Jean-Claude ou Chu-En-Laï dans cet exemple.) Par chance, PHP fournit aussi une fonction `ucfirst` qui sert exactement à cela : mettre la première lettre d'une chaine en capitale :


```
$ucfixe = ucfirst("une chaine quelconque avec même de la ponctuation");
echo $ucfixe;
```

Le résultat donne :

```
Une chaine quelconque avec même de la ponctuation
```

À partir de là, nous pouvons nous livrer à notre première conception de programme : pour obtenir un mot avec sa première lettre en capitale, nous appelons d'abord `strtolower` sur le mot, puis `ucfirst`. Pour ce faire, il faut une imbrication de l'appel de `strtolower` dans `ucfirst`. Voyons pourquoi, car il est très important de comprendre l'ordre d'évaluation du code.

Admettons que nous effectuions un simple appel à la fonction `print` :

```
print(5-8);
```

L'expression `5-8` est évaluée en premier, avec le résultat `-3`. (Comme indiqué au chapitre précédent, PHP convertit le résultat en une chaine pour l'afficher.) Si l'expression contient une fonction, cette fonction est évaluée également en premier lieu :

```
print(abs(5-8));
```

PHP suit plusieurs étapes dans l'exécution de cette courte instruction :

1. Il évalue `5-8` pour produire `-3`.
2. Il appelle la fonction `abs` pour convertir `-3` en `3`.
3. Il convertit le résultat en une chaine et l'affiche à l'aide de la fonction `print`.

Tout cela fonctionne parce que PHP évalue chaque élément à la suite, de l'intérieur vers l'extérieur. Le même procédé s'applique lorsque nous écrivons l'appel suivant :

```
ucfirst(strtolower("Une Chaine quelConque avec même de la Ponctuation"))
```

PHP passe la chaine d'abord à `strtolower` et son résultat seulement ensuite à `ucfirst`, pour produire le résultat (déjà obtenu avec les fonctions prises séparément) :

```
Une chaine quelconque avec même de la ponctuation
```

Définissons à présent une fonction (exemple 5-2) qui prend trois noms et les convertit en bas de casse, avec une première capitale.

Exemple 5-2. Nettoyer un nom complet

```
<?php
  echo fixe_noms("WILLIAM", "henry", "gatES");

  function fixe_noms($n1, $n2, $n3)
  {
    $n1 = ucfirst(strtolower($n1));
    $n2 = ucfirst(strtolower($n2));
```

```
    $n3 = ucfirst(strtolower($n3));
    return $n1 . " " . $n2 . " " . $n3;
  }
?>
```

Vous vous surprendrez à écrire souvent ce genre de code, parce que les utilisateurs ont tendance à laisser la touche de verrouillage active sur leur clavier, ou à insérer accidentellement des lettres capitales aux mauvais endroits ou même à oublier complètement les capitales. L'exemple produit le résultat suivant :

```
William Henry Gates
```

Retourner un tableau

Nous avons vu des fonctions qui renvoient une valeur simple. Il est aussi possible de renvoyer plusieurs valeurs à partir d'une fonction.

La première consiste à les renvoyer sous forme de tableau (*array*). Au chapitre 3, vous avez vu qu'un tableau s'apparente à une série de variables collées entre elles en une rangée. L'exemple 5-3 illustre la manière de renvoyer des valeurs d'une fonction à l'aide d'un tableau.

Exemple 5-3. Retourner plusieurs valeurs dans un tableau

```
<?php
  $noms = fixe_noms("WILLIAM", "henry", "gatES");
  echo $noms[0] . " " . $noms[1] . " " . $noms[2];

  function fixe_noms($n1, $n2, $n3)
  {
    $n1 = ucfirst(strtolower($n1));
    $n2 = ucfirst(strtolower($n2));
    $n3 = ucfirst(strtolower($n3));

    return array($n1, $n2, $n3);
  }
?>
```

Cette méthode a le mérite de conserver les trois noms séparés, au lieu de les concaténer en une seule chaine, de sorte que vous pouvez faire référence à un utilisateur par son prénom ou son nom de famille sans devoir extraire ce nom de la chaine renvoyée.

Ne pas passer les arguments par référence

Dans les versions de PHP antérieures à la 5.3.0, l'usage voulait de préfixer une variable du symbole & pour indiquer à l'interpréteur de passer à la fonction une référence à la variable au lieu de sa simple valeur. Ceci accordait à une fonction un accès à la variable

(pour écrire d'autres valeurs et les renvoyer modifiées à l'appelant), ce qui pouvait laisser place à un risque contre la sécurité et, surtout, laisse la place à des bogues extrêmement difficiles à repérer. Cela constitue aussi l'antithèse de la programmation orientée objet (POO).

Le passage d'arguments par référence est obsolète depuis PHP 5.3.0 et a été supprimé dans PHP 5.4.0. Par conséquent, vous ne pouvez plus utiliser cette fonctionnalité ailleurs que dans des sites hérités, anciens et encore en activité. Et même dans ces cas-là, il vous est recommandé de récrire le code qui passe des paramètres par référence, parce qu'il provoque un plantage avec erreur fatale dans les nouvelles versions de PHP.

Dans le cas où vous seriez appelé à maintenir du code ancien, vous devez savoir comment il fonctionne, pour construire les routines (programmes) de remplacement nécessaires. Ce concept peut s'avérer difficile à comprendre, donc reprenons l'analogie de la boite d'allumettes du chapitre 3.

Imaginez qu'au lieu de prendre un bout de papier dans une boite d'allumette, de le lire, de le copier sur un autre bout de papier, de déposer l'original de retour dans la boite et de passer la copie à une fonction (ouf!), vous attachiez un bout de fil au morceau de papier original et que vous passiez l'autre extrémité du fil à la fonction (figure 5-2).

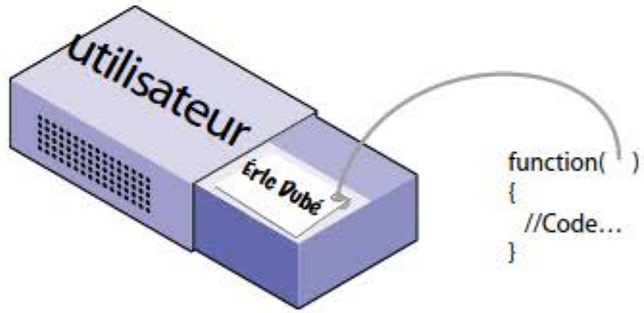


Figure 5-2. Imaginez une référence comme un fil attaché à une variable

À présent, la fonction peut suivre le fil pour trouver la donnée à laquelle elle doit accéder. Ce mécanisme évite tout le travail nécessaire pour créer une copie de la variable à l'usage de la seule fonction. De plus, la fonction a la possibilité de modifier le contenu de la variable.

Selon ce mécanisme, nous récririons l'exemple 5-3 pour passer des références à tous les paramètres, de sorte que la fonction puisse les modifier directement (exemple 5-4).

Exemple 5-4. Retour de valeurs de la fonction par référence

```
<?php
  $a1 = "WILLIAM";
  $a2 = "henry";
  $a3 = "gatES";

  echo $a1 . " " . $a2 . " " . $a3 . "<br>";
  fixe_noms($a1, $a2, $a3);
  echo $a1 . " " . $a2 . " " . $a3;

  function fixe_noms(&$n1, &$n2, &$n3)
  {
    $n1 = ucfirst(strtolower($n1));
    $n2 = ucfirst(strtolower($n2));
    $n3 = ucfirst(strtolower($n3));
  }
?>
```

Au lieu de passer directement les chaines à la fonction, vous les affectez à des variables et les affichez pour voir leurs valeurs d'avant. Ensuite, vous appelez la fonction comme précédemment, mais vous placez le symbole & en préfixe devant chacun des paramètres dans la définition de la fonction, ce qui indique à PHP de passer seulement les références des variables.

À ce stade, les variables `$n1`, `$n2` et `$n3` sont attachées par les « fils » qui mènent aux valeurs de `$a1`, `$a2` et `$a3`. Autrement dit, il n'y a qu'un seul groupe de valeurs, mais deux ensembles de noms de variables sont autorisés à accéder à ces mêmes valeurs.

Par conséquent, la fonction `fixe_noms` n'a plus qu'à attribuer de nouvelles valeurs à `$n1`, `$n2` et `$n3` pour mettre à jour les valeurs de `$a1`, `$a2` et `$a3`. Le programme donne les résultats suivants :

```
WILLIAM henry gatES
William Henry Gates
```

Dans les deux affichages, les instructions `echo` n'utilisent les valeurs que de `$a1`, `$a2` et `$a3`.

J'insiste à nouveau sur le fait que cette pratique n'est plus admise en PHP et que vous devez donc récrire tout code qui repose sur le passage de paramètres par référence. Parfois il s'agit simplement de supprimer les symboles `&`, parce qu'ils n'étaient pas nécessaires au départ. Dans d'autres cas, il faut faire appel à des variables globales, comme dans l'exemple suivant.

Renvoi de variables globales

Le meilleur moyen de donner à une fonction un accès à une variable créée à l'extérieur consiste à déclarer cette variable comme globale au sein de la fonction. Le mot clé `global` suivi du nom de la variable permet à la totalité du code d'y accéder (exemple 5-5).

Exemple 5-5. Renvoi de valeurs dans des variables globales

```
<?php
  $a1 = "WILLIAM";
  $a2 = "henry";
  $a3 = "gatES";

  echo $a1 . " " . $a2 . " " . $a3 . "<br>";
  fixe_noms();
  echo $a1 . " " . $a2 . " " . $a3;
  function fixe_noms()
  {
    global $a1; $a1 = ucfirst(strtolower($a1));
    global $a2; $a2 = ucfirst(strtolower($a2));
    global $a3; $a3 = ucfirst(strtolower($a3));
  }
?>
```

À ce stade, il n'est même plus nécessaire de passer de paramètres à la fonction et celle-ci n'en attend pas. Une fois déclarées, ces variables demeurent globales et accessibles à tout le reste du programme, y compris aux autres fonctions.

Pour conserver une portée aussi locale que possible, évitez les variables globales et essayez plutôt de retourner des tableaux ou des variables par association. Sinon, vous commencerez à perdre certains des avantages des fonctions.

Retour sur la portée des variables

Voici un récapitulatif rapide de ce que vous avez appris au chapitre 3 :

- Les *variables locales* sont accessibles uniquement à la portion de code où elles sont définies. Si elles sont définies hors d'une fonction, elles ne sont accessibles qu'à tout le code en dehors des fonctions, des classes et ainsi de suite. Si elles sont définies au sein d'une fonction, seule cette fonction peut y accéder et leur valeur est perdue quand l'exécution de la fonction se termine.
- Les *variables globales* sont accessibles à toutes les parties du code.
- Les *variables statiques* sont accessibles uniquement au sein de la fonction qui les déclare, mais leur valeur est conservée, d'un appel au suivant de la fonction.

Inclure et exiger des fichiers

À mesure que vous progressez dans la programmation en PHP, vous vient rapidement l'envie de constituer une bibliothèque (*library*) de fonctions que vous pensez réutiliser ultérieurement. Vous utiliserez aussi des bibliothèques conçues par d'autres développeurs.

Il n'est pas nécessaire de copier et coller ces fonctions dans vos programmes. Il suffit de les enregistrer dans des fichiers distincts et d'utiliser quelques commandes pour y faire référence. Deux types de commandes permettent de réaliser ces appels : `include` et `require`.

Instruction include

La commande `include` indique à PHP d'aller chercher un fichier déterminé et d'en charger le contenu. Cela revient presque à copier-coller le fichier inclus dans le fichier programme courant, au point d'insertion indiqué. L'exemple 5-6 montre comment inclure un fichier nommé *library.php*.

Exemple 5-6. Inclure un fichier PHP

```
<?php
  include("library.php");

  // Votre code vient ici
?>
```

Inclure une seule fois : include_once

Chaque fois que vous ajoutez une directive `include`, elle provoque le chargement dudit fichier, encore et encore, même si vous l'avez déjà inclus. Supposons que vous disposez d'une bibliothèque de fonctions tellement utiles que vous l'incluez dans un de vos programmes. Mais vous incluez aussi une autre bibliothèque qui fait elle-même appel à la première, *library.php*. Du fait de l'imbrication, vous avez chargé deux fois la même bibliothèque. Ceci engendre des messages d'erreur car vous essayez de définir les mêmes constantes ou fonctions à plusieurs reprises. Pour éviter ces problèmes ; utilisez plutôt `include_once`, comme illustré dans l'exemple 5-7.

Exemple 5-7. Inclure un fichier PHP une seule fois

```
<?php
  include_once("library.php");

  // Votre code vient ici
?>
```

À ce stade, si PHP rencontre un autre `include` ou `include_once` du même fichier, alors qu'il a déjà été chargé, il l'ignore complètement. Pour déterminer si le fichier a déjà été inclus, PHP compare le chemin de fichier absolu, après avoir résolu tous les chemins relatifs et trouvé le fichier dans le chemin (*path*) d'`include`.

D'une manière générale, il est préférable de n'utiliser que des `include_once` et d'ignorer l'instruction include de base. De cette façon, vous n'aurez jamais de mauvaise surprise due à plusieurs inclusions d'un même fichier.

Exiger et exiger une seule fois : require et require_once

Le problème d'`include` et `include_once` se situe dans le fait que PHP ne fait que tenter d'inclure le fichier demandé. L'exécution du programme se poursuit même si le fichier n'a pas été trouvé.

Lorsqu'il est absolument indispensable d'inclure un fichier, exigez-le : `require`. Pour la même raison que `include_once` est préférable à `include`, je recommande d'utiliser de manière générale la version `require_once` chaque fois que vous devez exiger un fichier (exemple 5-8).

Exemple 5-8. Exiger un fichier PHP une seule fois

```
<?php
  require_once("library.php");

  // Votre code vient ici
?>
```

Compatibilité de versions de PHP

PHP bénéficie d'un processus de développement permanent et très dynamique, ce qui implique qu'il en existe plusieurs versions. Lorsque vous devez vérifier si une fonction déterminée est disponible pour l'utiliser dans un programme, la fonction `function_exists` en vérifie l'existence parmi toutes les fonctions prédéfinies et celles définies par l'utilisateur.

L'exemple 5-9 vérifie si la fonction `array_combine`, spécifique à PHP 5, existe bien.

Exemple 5-9. Vérification de l'existence d'une fonction déterminée

```
<?php
  if (function_exists("array_combine"))
  {
    echo "La fonction existe";
  }
  else
  {
    echo "La fonction n'existe pas - Créez la vôtre";
  }
?>
```

Une telle manière de programmer permet de tirer parti des fonctionnalités des nouvelles versions de PHP mais permet également au programme de fonctionner sur les anciennes versions, du moment que vous répliquez les fonctionnalités manquantes. Vos fonctions seront bien entendu plus lentes que celles intégrées, mais votre code sera bien plus *portable*, c'est-à-dire utilisable sous plusieurs environnements PHP différents.

La fonction `phpversion` permet en outre de connaitre la version de PHP courante sous laquelle votre programme fonctionne au moment de son exécution. Le résultat obtenu varie selon la version mais est comparable à ceci :

```
5.6.3
```

Objets en PHP

De la même façon que les fonctions ont représenté une grande avancée dans la puissance de programmation dès les premiers jours de l'histoire de l'informatique, où parfois les seuls moyens de navigation possibles se résumaient à de très basiques instructions GOTO et GOSUB, la *programmation orientée objet* (*POO*) a emmené l'utilisation des fonctions à un tout autre niveau, sans égal jusqu'alors.

Une fois que vous avez pris le pli et que vous vous êtes habitué à condenser des portions de code réutilisables sous forme de fonctions, il ne vous faut pas effectuer un bond si énorme pour envisager le regroupement de fonctions et de leurs données dans des objets.

Reprenons l'exemple du site de réseau social et de ses nombreuses parties. Une partie regroupe toutes les fonctions liées à la gestion des utilisateurs, c'est-à-dire le code nécessaire pour permettre à de nouveaux utilisateurs de s'inscrire et aux utilisateurs déjà inscrits de modifier leurs coordonnées. En PHP standard, vous devriez créer quelques fonctions pour traiter cela et intégrer quelques appels à la base de données MySQL pour assurer le suivi de tous les utilisateurs.

Imaginez combien plus facile ce serait de créer un objet pour représenter l'utilisateur courant. Pour y parvenir, vous pourriez créer une classe, appelée par exemple `Utilisateur`, qui contiendrait tout le programme nécessaire pour gérer les utilisateurs, avec toutes les variables indispensables pour manipuler les données au sein de la classe. Ensuite, chaque fois que vous devriez manipuler les données d'un utilisateur, il vous suffirait de créer un objet de cette classe `Utilisateur`.

Vous pourriez traiter ce nouvel objet comme si c'était un véritable utilisateur. Ainsi, vous pourriez passer à l'objet un nom, un mot de passe et une adresse de courrier électronique, lui demander si tel utilisateur existe déjà, pour, si ce n'est le cas, créer un nouvel utilisateur avec ces attributs. Vous pourriez même disposer d'un objet messagerie instantanée ou encore un autre pour gérer le fait que deux utilisateurs sont amis.

Terminologie

Lors de la création d'un programme destiné à gérer des objets, vous devez concevoir un modèle composé de données et de code, appelé *classe*. Chaque nouvel objet basé sur cette classe s'appelle une *instance* (ou occurrence) de cette classe.

Les données associées à un objet sont nommées les *propriétés* de cet objet, tandis que les fonctions qu'il renferme s'appellent des *méthodes*. Définir une classe revient à donner des noms à ses propriétés et du code pour ses méthodes. La figure 5-3 propose une analogie de l'objet à un jukebox. Imaginez les CD qu'il contient dans son carrousel comme ses propriétés et la méthode pour les écouter revient à presser les boutons de son panneau de commandes. Il contient aussi une fente pour insérer des pièces (la méthode qui sert à activer l'objet) et le lecteur de disque laser (la méthode pour rechercher la musique ou les propriétés des CD).

Lorsque vous créez des objets, le mieux est de tirer parti de l'*encapsulation*, c'est-à-dire l'écriture d'une classe de telle manière que seules ses méthodes permettent de manipuler ses propriétés. Autrement dit, d'interdire au code de l'extérieur l'accès direct à ses données. Les méthodes fournies s'appellent l'*interface* de l'objet.

Cette approche simplifie le débogage, car la correction d'un programme fautif n'a lieu qu'au sein d'une classe. De plus, lorsqu'il faut faire évoluer et mettre à niveau un programme, si vous avez respecté une encapsulation correcte et conservé la même interface, vous pouvez simplement développer des classes de remplacement et les échanger contre les anciennes. Si ces nouvelles versions ne fonctionnent pas, vous les échangez à nouveau contre les anciennes pour régler le problème immédiatement, avant de vous lancer dans le débogage approfondi des nouvelles classes.

Dès que vous avez créé une classe, vous risquez de constater que vous avez besoin d'une autre classe très semblable mais non identique. La technique facile et rapide consiste alors à définir une nouvelle classe basée sur l'*héritage* de la première. Selon ce mécanisme, la nouvelle classe reçoit toutes les propriétés de celle dont elle hérite. La classe initiale est appelée *superclasse* et la nouvelle, sa *sous-classe* ou classe *dérivée*.

Figure 5-3. Le jukebox, excellent exemple d'objet avec son contenu

Pour reprendre l'analogie du jukebox, si vous en inventez un nouveau, capable de lire des vidéos en plus de la musique, vous pouvez hériter de toutes les propriétés et méthodes de l'original, la superclasse, et ajouter de nouvelles propriétés (vidéos) et de nouvelles méthodes (lecteur vidéo).

Un des avantages indéniables de ce système réside dans le fait que si vous améliorez la rapidité ou tout autre aspect de la superclasse, ses sous-classes profitent aussi de ces améliorations.

Déclarer une classe

Avant de pouvoir utiliser un objet, il faut en définir la classe, avec le mot clé `class`. La définition d'une classe contient le nom de la classe (sensible à la casse), ses propriétés et ses méthodes. L'exemple 5-10 définit une classe `Utilisateur` avec deux propriétés, `$nom` et `$motpasse`, indiqués par le m ot clé `public` (voir Portée des propriétés et méthodes en PHP 5, page 116). Il crée aussi une nouvelle instance de cette classe, appelée `$objet`.

Exemple 5-10. Déclaration d'une classe et examen d'un objet

```
<?php
  $objet = new Utilisateur;
  print_r($objet);

  class Utilisateur
  {
    public $nom, $motpasse;

    function enregistrer_utilisateur()
    {
          echo "Le code d'enregistrement de l'utilisateur vient ici";
    }
  }
?>
```

Cet exemple utilise aussi une fonction d'une valeur inestimable nommée `print_r`. Elle demande à PHP d'afficher des informations relatives à une variable sous une forme lisible par l'homme. C'est d'ailleurs ce que signifie le `_r` : *in human-readable format*. Pour le nouvel objet `$objet`, elle affiche :

```
Utilisateur Object
(
  [nom] =>
  [motpasse] =>
)
```

Cependant, comme un navigateur compresse les espaces, le résultat dans un navigateur est plus difficile à lire :

```
Utilisateur Object ( [nom] => [motpasse] => )
```


Quoi qu'il en soit, le résultat indique que `$objet` est un objet (`Object`) `Utilisateur` qui possède les propriétés `$nom` et `$motpasse`.

Créer un objet

Pour créer un objet d'une classe donnée, utilisez le mot clé `new`, comme suit : `objet = new Classe`. Voici deux manières de le faire :

```
$objet = new Utilisateur;
$temp  = new Utilisateur('nom', 'mot de passe');
```

La première ligne affecte un objet à la classe `Utilisateur`, tandis que la seconde le fait mais en passant des paramètres à l'appel.

Une classe peut exiger ou interdire les arguments ; elle peut aussi permettre les arguments mais sans les exiger.

Accéder aux objets

Ajoutons quelques lignes à l'exemple 5-10 et voyons les résultats. L'exemple 5-11 étend le programme précédent pour définir les propriétés de l'objet et appeler une méthode.

Exemple 5-11. Créer un objet et interagir avec lui

```
<?php
  $objet = new Utilisateur;
  print_r($objet); echo "<br>";

  $objet->nom = "Jules";
  $objet->motpasse = "monmotpasse";
  print_r($objet); echo "<br>";

  $objet->enregistrer_utilisateur();

  class Utilisateur
  {
    public $nom, $motpasse;

    function enregistrer_utilisateur()
    {
      echo "Le code d'enregistrement de l'utilisateur vient ici";
    }
  }
?>
```

Vous découvrez ici la syntaxe qui permet d'accéder à une propriété d'un objet : `$objet->propriété`. De la même manière, vous appelez une méthode comme ceci : `$objet->méthode()`.

Remarquez que, dans cette syntaxe, `propriété` et `méthode` ne portent pas de symbole `$` en préfixe. Si vous les préfixez d'un `$`, le code ne s'exécute plus car il tente de référencer la valeur à l'intérieur d'une variable. Ainsi, l'expression `$objet->$propriété` tente de lire la valeur affectée à une variable nommée `$propriété` (admettons que sa valeur soit la chaine `brun`), puis tente de référencer la propriété `$objet->brun`. Si `$propriété` est indéfini, alors une tentative de faire référence à `$objet->NULL` se produit et provoque une erreur.

Lorsque vous l'examinez à l'aide de la fonction d'affichage du code source du navigateur, la sortie de l'exemple 5-11 produit ce qui suit :

```
Utilisateur Object
(
    [nom] =>
    [motpasse] =>
)
<br>Utilisateur Object
(
    [nom] => Jules
    [motpasse] => monmotpasse
)
<br>Le code d'enregistrement de l'utilisateur vient ici
```

Une fois de plus, `print_r` démontre son utilité dans l'affichage du contenu de `$objet` avant et après l'affectation de valeurs aux propriétés. À partir d'ici, j'élude les instructions `print_r` dans les programmes mais vous savez qu'à tout moment, vous pouvez en insérer quelques-unes pour voir ce qui se passe exactement dans les exemples suivants.

Les résultats de l'exemple montrent également que la méthode `enregistrer_utilisateur` a été exécutée par l'appel à la méthode. L'affichage de la chaine rappelle que nous devons encore créer un peu de code.

L'emplacement des définitions des fonctions et des classes n'a pas d'importance, que ce soit avant ou après les instructions qui les utilisent. Cependant, elles sont généralement placées vers la fin du fichier.

Cloner des objets

Une fois un objet créé, lorsque vous le passez en paramètre, il l'est par référence. Selon l'analogie de la boite d'allumettes, cela revient à conserver plusieurs fils attachés à un objet stocké dans une boite d'allumettes, de sorte que vous pouvez suivre n'importe quel fil pour y accéder.

En d'autres termes, les affectations d'objets ne copient pas les objets dans leur totalité. L'exemple 5-12 illustre ce fait. Nous y définissons une classe `Utilisateur` très simple, sans méthode et avec la seule propriété `nom`.

Exemple 5-12. Copie d'un objet

```
<?php
  $objet1 = new Utilisateur();
  $objet1->nom = "Alice";
  $objet2 = $objet1;
  $objet2->nom = "Annie";

  echo "Nom d'objet1 = " . $objet1->nom . "<br>";
  echo "Nom d'objet2 = " . $objet2->nom;

  class Utilisateur
  {
    public $nom;
  }
?>
```

Nous avons créé l'objet `$objet1` et avons affecté la valeur `Alice` à sa propriété `nom`. Ensuite, nous créons `$objet2` par affectation de la valeur de `$objet1` et nous attribuons la valeur `Annie` à la propriété `nom` du seul `$objet2`, du moins c'est ce que nous croyons. Mais le code produit le résultat suivant :

```
Nom d'objet1 = Annie
Nom d'objet2 = Annie
```

Que s'est-il passé ? Les objets `$objet1` et `$objet2` font référence au même objet, par conséquent, la modification en `Annie` de la propriété `nom` de `$objet2` règle aussi la propriété de `$objet1`.

Pour éviter cette confusion, utilisez l'opérateur `clone`, qui crée une toute nouvelle instance de la classe et copie les valeurs des propriétés de l'instance initiale vers la nouvelle instance. L'exemple 5-13 illustre son utilisation.

Exemple 5-13. Clonage d'un objet

```
<?php
  $objet1 = new Utilisateur();
  $objet1->nom = "Alice";
  $objet2 = clone $objet1;
  $objet2->nom = "Annie";

  echo "Nom d'objet1 = " . $objet1->nom . "<br>";
  echo "Nom d'objet2 = " . $objet2->nom;

  class Utilisateur
  {
    public $nom;
  }
?>
```

Et voilà ! Les résultats de ce code sont ceux que nous espérions initialement :

```
Nom d'objet1 = Alice
Nom d'objet2 = Annie
```

Constructeurs

Lors de la création d'un objet, vous pouvez passer une liste d'arguments à la classe appelée. Ils sont transmis à une méthode particulière de la classe, appelée son *constructeur*, pour initialiser différentes propriétés.

Par le passé, vous deviez en principe donner à cette méthode le même nom que la classe, comme à l'exemple 5-14.

Exemple 5-14. Création d'une méthode de constructeur

```
<?php
  class Utilisateur
  {
    function Utilisateur($param1, $param2)
    {
      // Les instructions du constructeur viennent ici
      public $nomutilisateur = "Invité";
    }
  }
?>
```

Cependant, PHP 5 fournit une approche beaucoup plus logique pour nommer le constructeur, qui consiste à nommer cette méthode `__construct` (c'est-à-dire `construct` préfixé de deux caractères de soulignement), comme à l'exemple 5-15.

Exemple 5-15. Création d'une méthode de constructeur en PHP 5

```
<?php
  class Utilisateur
  {
    function __construct($param1, $param2)
    {
      // Les instructions du constructeur viennent ici
      public $nomutilisateur = "Invité";
    }
  }
?>
```

Destructeurs en PHP 5

Une autre nouveauté de PHP 5 réside dans la possibilité de définir des méthodes de destructeur. Cette possibilité s'avère utile lorsque le code a établi la dernière référence à un objet ou quand le programme touche à sa fin. L'exemple 5-16 illustre la création d'une méthode de destructeur.

Exemple 5-16. Création d'une méthode de destructeur en PHP 5

```
<?php
  class Utilisateur
  {
    function __destruct()
    {
      // Les instructions du destructeur viennent ici
    }
  }
?>
```

Écriture de méthodes

La déclaration d'une méthode ressemble fort à la déclaration d'une fonction, comme vous venez de le voir, mais quelques différences subsistent. Ainsi, les méthodes dont les noms débutent par un double caractère de soulignement (`__`) sont réservées et vous ne pouvez pas en créer de nouvelles sous cette forme.

Vous disposez automatiquement d'une variable spéciale dénommée `$this`, réservée pour accéder à l'objet courant et à ses propriétés. L'exemple 5-17 en illustre le fonctionnement, avec une nouvelle méthode de la définition de la classe `Utilisateur`, appelée `get_motpasse`. Remarquez au passage que `get` signifie « obtenir » ou « lire », et correspond à une convention que nous détaillerons plus loin dans le livre.

Exemple 5-17. Utilisation de la variable $this dans une méthode

```
<?php
  class Utilisateur
  {
    public $nom, $motpasse;

    function get_motpasse()
    {
      return $this->motpasse;
    }
  }
?>
```

La méthode `get_motpasse` exploite la variable `$this` pour accéder à l'objet courant et retourner la valeur de la propriété `$motpasse` de cet objet courant. Remarquez qu'une fois encore, le préfixe `$` de la propriété `$motpasse` est éludé lors de l'utilisation de l'opérateur `->`. Le fait de laisser le `$` en place constitue une erreur de programmation courante que vous risquez de rencontrer, surtout lorsque vous utilisez cet opérateur les premières fois.

Voici comment utiliser la méthode `get_motpasse` de la classe, définie dans l'exemple 5-17 :

```
$objet           = new Utilisateur;
$objet->motpasse = "secret";

echo $objet->get_motpasse();
```

Ce code affiche le mot de passe `secret`.

Méthodes statiques en PHP 5

Lorsque vous utilisez PHP 5, vous pouvez aussi définir une méthode comme statique (mot clé `static`), ce qui signifie qu'elle est appelée au niveau de la classe et non plus au niveau d'un objet de cette classe. Une méthode statique n'a aucun accès aux propriétés des objets individuels. L'exemple 5-18 montre la création d'une telle méthode et la manière d'y accéder.

Exemple 5-18. Création d'une méthode statique et accès à cette méthode

```
<?php
  Utilisateur::mdp_string();

  class Utilisateur
  {
    static function mdp_string()
    {
      echo "Entrez votre mot de passe";
    }
  }
?>
```

Remarquez la manière dont nous appelons la classe elle-même, suivie de la méthode statique, à l'aide d'un double deux-points (`::`), qui s'appelle l'opérateur de résolution de portée, au lieu du `->`. Les fonctions statiques s'avèrent utiles pour effectuer des tâches en relation avec la classe en elle-même, mais pas avec les instances particulières de la classe. L'exemple 5-21 montre un autre exemple de méthode statique.

Un message d'erreur apparaît lorsque vous essayez d'accéder à `$this->propriété` ou à toute autre propriété d'objet à partir d'une méthode statique.

Déclarer des propriétés

Il n'est pas nécessaire de déclarer explicitement les propriétés au sein des classes, car elles peuvent être définies implicitement lors de leur première utilisation. Pour illustrer ceci, dans l'exemple 5-19, la classe `Utilisateur` n'a aucune propriété ni méthode, pourtant ce code demeure licite.

Exemple 5-19. Définition implicite d'une propriété

```
<?php
  $objet1 = new Utilisateur();
  $objet1->nom = "Alice";

  echo $objet1->nom;

  class Utilisateur {}
?>
```

Ce programme affiche la chaine `Alice` sans aucun problème, parce que PHP déclare de manière implicite la variable `$objet1->nom` à votre place. Retenez toutefois que cette manière de programmer génère des bogues abominablement difficiles à retrouver, parce que `nom` est déclaré en dehors de la classe.

Pour rendre service à vous-même et à toute personne qui devra maintenir votre code, je vous conseille de prendre l'habitude de systématiquement déclarer vos propriétés de manière explicite. Vous verrez : dans certains cas, vous serez heureux de l'avoir fait.

Par ailleurs, lorsque vous déclarez une propriété dans une classe, vous pouvez lui affecter une valeur prédéfinie. Cette valeur doit être une constante ou une valeur littérale mais jamais le résultat d'une fonction ni d'une expression. L'exemple 5-20 montre quelques affectations valides et non permises.

Exemple 5-20. Exemples de déclarations valides et non valides

```
<?php
  class Test
  {
    public $nom      = "Éric Dubé";  // Valide
    public $age      = 42;           // Valide
    public $heure    = time();       // Non valide - appelle une fonction
    public $score    = $niveau * 2;  // Non valide - utilise une expression
  }
?>
```

Déclarer des constantes

Tout comme vous créez une constante globale à l'aide de la fonction `define`, vous pouvez définir des constantes au sein des classes. La pratique généralement acceptée indique d'utiliser des lettres en capitales pour les faire ressortir, comme dans l'exemple 5-21.

Exemple 5-21. Définition de constantes dans une classe

```
<?php
  Traduction::lecture();

  class Traduction
  {
```

```
    const ANGLAIS  = 0;
    const ESPAGNOL = 1;
    const FRANCAIS = 2;
    const ALLEMAND = 3;
    // ...

    static function lecture()
    {
      echo self::FRANCAIS;
    }
  }
?>
```

Une nouveauté : le mot clé `self`, suivi de l'opérateur double deux-points, `self::`, permet de faire directement référence à de telles constantes. Remarquez aussi que la première ligne de ce code appelle directement la classe, à l'aide de l'opérateur `::`, sans même créer d'instance de la classe, ceci parce qu'il s'agit d'une méthode statique, accessible au niveau de la classe. Comme vous vous y attendez sans doute, la valeur affichée à l'exécution du code est `2`.

Rappelez-vous qu'une fois définie, la valeur d'une constante ne peut plus changer.

Portée des propriétés et méthodes en PHP 5

PHP 5 fournit trois mots clés pour contrôler la portée des propriétés et des méthodes :

`public`
: C'est l'option par défaut lorsque vous déclarez une variable à l'aide des mots clés `var` ou `public`, ou lors de la déclaration implicite d'une variable à sa première utilisation. Les mots clés `var` et `public` sont interchangeables parce que, bien qu'obsolète, `var` est conservé en vue de la compatibilité avec les anciennes versions de PHP. Les méthodes sont considérées comme publiques par défaut.

`protected`
: Ces propriétés et méthodes (dites *membres*) ne peuvent être référencées que par les méthodes de la classe de l'objet et par celles de toutes ses sous-classes.

`private`
: Ces membres ne peuvent être référencés que par les méthodes de la même classe, pas par celles de ses sous-classes.

Voici un guide d'aide à la décision, quant au choix :

- Utilisez `public` quand du code à l'extérieur de la classe peut accéder à ce membre et quand les sous-classes peuvent aussi en hériter.
- Utilisez `protected` quand le code à l'extérieur de la classe ne peut pas accéder à ce membre et quand les sous-classes peuvent en hériter.

- Utilisez `private` quand le code à l'extérieur de la classe ne peut pas accéder à ce membre et quand les sous-classes ne peuvent pas non plus en hériter.

L'exemple 5-22 illustre l'utilisation de ces mots clés.

Exemple 5-22. Modificateurs de portée de propriétés et méthodes

```
<?php
  class Exemple
  {
    var $nom = "Michel";           // identique à public mais obsolète
    public $age = 23;              // propriété publique
    protected $compteutilisateurs; // propriété protégée, héritable

    private function admin()       // méthode strictenent privée
    {
      // Le code d'administration vient ici
    }
  }
?>
```

Propriétés et méthodes statiques

La plupart des données et des méthodes concernent les instances d'une classe. Ainsi, dans une classe `Utilisateur`, il est nécessaire de mémoriser le mot de passe spécifique à un utilisateur ou de vérifier quand cet utilisateur s'est inscrit. Ces faits et actions concernent distinctement chaque utilisateur et empruntent donc des propriétés et méthodes spécifiques à l'instance (l'objet).

À certaines occasions, il peut être nécessaire de conserver des données à propos de la classe elle-même. C'est le cas notamment, si vous devez conserver le nombre d'utilisateurs inscrits. Dans ce cas-là, vous définissez une variable qui s'applique à la totalité de la classe `Utilisateur`. PHP prévoit des propriétés et des méthodes statiques pour ce genre de situation.

Comme le démontrait brièvement l'exemple 5-18, la déclaration comme `static` des membres d'une classe les rend accessibles sans nécessiter d'instanciation préalable de la classe. Une instance d'une classe ne peut accéder directement à une propriété déclarée comme `static`, mais une méthode statique le peut.

L'exemple 5-23 définit une classe `Test` avec une propriété statique et une méthode publique (par défaut). Voyez les subtilités des accès à l'une et à l'autre.

Exemple 5-23. Définition d'une classe avec une propriété statique

```
<?php
  $temp = new Test();

  echo "Test A: " . Test::$propriete_statique . "<br>";
  echo "Test B: " . $temp->get_ps()           . "<br>";
```

```
  echo "Test C: " . $temp->propriete_statique . "<br>";

  class Test
  {
    static $propriete_statique = "Je suis statique";

    function get_ps()
    {
      return self::$propriete_statique;
    }
  }
?>
```

Lors de l'exécution, ce script affiche ce qui suit :

```
Test A: Je suis statique
Test B: Je suis statique

Notice: Undefined property: Test::$propriete_statique in ... on line 6
Test C:
```

Cet exemple montre que la propriété `$propriete_statique` est adressable directement à partir de la classe elle-même, grâce à l'opérateur `::`, dans Test A. Ensuite, Test B peut lire la valeur par un appel de la méthode (publique par défaut) `get_ps` de l'objet `$temp`, créé à partir de la classe `Test`. En revanche, le Test C échoue, parce qu'en tant qu'instance de la classe, l'objet `$temp` n'a pas le droit d'accéder à la propriété statique `$propriete_statique`.

Remarquez que la méthode `get_ps` accède à `$propriete_statique` par l'entremise du mot clé `self`. Il s'agit là de la manière d'accéder à une propriété ou une constante statique au sein d'une classe.

Héritage

À partir du moment où vous avez édifié une classe, vous pouvez en dériver une ou plusieurs sous-classes. Cela vous évite des tonnes de récriture de code : prenez une classe semblable à celle que vous devez construire, dérivez-en une sous-classe et modifiez juste les portions de code qui diffèrent. L'opérateur `extends` se charge de dériver une classe d'une superclasse. Retenez : `extends` signifie « dérive de ».

L'exemple 5-24 suivant déclare la classe `Abonne` (sans accent, comme toujours, pour un abonné au site web) comme une sous-classe d'`Utilisateur` à l'aide de l'opérateur `extends`.

Exemple 5-24. Héritage et dérivation d'une classe

```
<?php
  $objet          = new Abonne;
  $objet->nom     = "Éric";
```



```
  $objet->motpasse = "mdp";
  $objet->tel      = "1-012 345 6789";
  $objet->courriel = "eric@bloggs.com";
  $objet->afficher();

  class Utilisateur
  {
    public $nom, $motpasse;

    function enregistre_utilisateur()
    {
      echo "Code d'enregistrement utilisateur ici";
    }
  }

  class Abonne extends Utilisateur
  {
    public $tel, $courriel;

    function afficher()
    {
      echo "nom :     " . $this->nom      . "<br>";
      echo "MdP :     " . $this->motpasse . "<br>";
      echo "Tél. :    " . $this->tel      . "<br>";
      echo "Courriel: " . $this->courriel;
    }
  }
?>
```

La classe `Utilisateur` originale a deux propriétés, `$nom` et `$motpasse`, et une méthode (encore vide) pour enregistrer l'utilisateur dans la base de données. La classe `Abonne` étend cette classe par l'ajout de deux autres propriétés, `$tel` et `$courriel`, ainsi que d'une méthode pour afficher les propriétés de l'objet courant à l'aide de la variable `$this`, qui fait référence aux valeurs courantes de l'objet en cours d'accès. Les résultats d'affichage de ce code sont les suivants :

```
nom :     Éric
MdP :     mdp
Tél. :    1-012 345 6789
Courriel: eric@bloggs.com
```

Remarquez que les jolis espacements que nous avons définis dans le code source ont disparu car, est-il nécessaire de le rappeler, le navigateur compresse les espaces. Notez aussi que la manière dont s'affichent les caractères accentués dépend de l'encodage du fichier. En ASCII, les « é » peuvent apparaître sous forme d'un losange noir avec un point d'interrogation en blanc (). Pour voir les caractères accentués reproduits correctement, convertissez vos fichiers en UTF-8, par exemple.

Opérateur « parent »

Si vous définissez une méthode dans une sous-classe de même nom qu'une de celles de la classe mère, ses instructions surchargent et remplacent celle du parent. Or souvent, ce n'est pas le but et vous devez tout de même accéder à la méthode du parent. Dans ce cas-là, vous disposez de l'opérateur `parent`, illustré dans l'exemple 5-25.

Exemple 5-25. Surcharge d'une méthode et opérateur parent

```
<?php
  $objet = new Fils;
  $objet->test();
  $objet->test2();

  class Pere
  {
    function test()
    {
      echo "[Classe Pere] Je suis ton père<br>";
    }
  }

  class Fils extends Pere
  {
    function test()
    {
      echo "[Classe Fils] C'est moi, Luke<br>";
    }

    function test2()
    {
      parent::test();
    }
  }
?>
```

Ce script crée une classe dénommée `Pere`, puis une sous-classe nommée `Fils` qui hérite des propriétés et méthodes de la première, pour écraser la méthode `test`. Par conséquent, quand la deuxième ligne appelle la méthode `test`, c'est celle du `Fils` qui s'exécute. La seule manière d'exécuter la méthode `test` de la classe `Pere`, en l'appelant à partir de `$objet`, créé à partir de la classe `Fils`, c'est, comme dans la méthode `test2`, d'utiliser l'opérateur `parent`, puis les deux deux-points (`::`) et la méthode de `Pere`. Le code génère les résultats suivants :

```
[Classe Fils] C'est moi, Luke
[Classe Père] Je suis ton père
```

Lorsque vous voulez vous assurer que le code appelle une méthode de la classe courante, faites appel au mot clé `self`, comme suit :

```
self::methode();
```


Constructeurs de sous-classe

Lors de la dérivation d'une classe et de la déclaration de votre propre constructeur dans la sous-classe, soyez conscient du fait que PHP n'appelle pas automatiquement la méthode de constructeur de la superclasse. Si vous voulez garantir que tout le code d'initialisation s'exécute dans celle-ci, le constructeur de la sous-classe doit toujours appeler le constructeur parent, comme illustré dans l'exemple 5-26.

Exemple 5-26. Appel du constructeur du parent dans toute sous-classe

```
<?php
  $objet = new Tigre();
  echo "Les tigres ont...<br>";
  echo "Fourrure : " . $objet->fourrure . "<br>";
  echo "Rayures : " . $objet->rayures;

  class Felin
  {
    public $fourrure; // Les félins ont de la fourrure

    function __construct()
    {
      $this->fourrure = "TRUE";
    }
  }

  class Tigre extends Felin
  {
    public $rayures; // Les tigres ont des rayures

    function __construct()
    {
      parent::__construct();   // Appeler d'abord constructeur du parent
      $this->rayures = "TRUE"; // Préciser le reste
    }
  }
?>
```

Cet exemple tire parti de l'héritage d'une façon assez répandue et usuelle. La classe `Felin` crée la propriété `$fourrure`, que nous souhaitons réutiliser. Donc la classe `Tigre` hérite de `Felin`, de `$fourrure`, et crée une propriété supplémentaire, `$rayures`. Pour vérifier que les deux constructeurs ont bien été appelés, le programme affiche ce qui suit :

```
Les tigres ont...
Fourrure : TRUE
Rayures : TRUE
```

Méthodes « final »

Dans certains cas, il est nécessaire de figer une méthode dans une superclasse pour éviter que les sous-classes ne la surchargent. Pour cela existe le mot clé final, dont l'exemple 5-27 montre l'utilisation.

Exemple 5-27. Création d'une méthode final

```
<?php
  class Utilisateur
  {
    final function droitsdauteur()
    {
      echo "Cette classe a été créée par Éric Dubé";
    }
  }
?>
```

Maintenant que vous avez digéré le contenu de ce long chapitre, vous devriez avoir une meilleure idée d'ensemble de ce que PHP peut faire pour vous. Vous êtes désormais en mesure d'utiliser les fonctions avec une certaine aisance et, si vous le préférez, rédiger du code orienté objet. Au chapitre 6, nous terminons notre exploration théorique de PHP avec un sujet majeur : les rouages des tableaux en PHP.

Questions

1. Quel est le principal avantage de l'utilisation d'une fonction ?
2. Combien de valeurs une fonction peut-elle renvoyer ?
3. Quelle est la différence entre accéder à une variable par nom et par référence ?
4. Quelle est la signification de « portée » en PHP ?
5. Comment insère-t-on un fichier de code PHP dans un autre ?
6. En quoi un objet diffère-t-il d'une fonction ?
7. Comment crée-t-on un objet en PHP ?
8. Quelle syntaxe utilisez-vous pour créer une sous-classe d'une classe existante ?
9. Comment faites-vous pour appeler une portion de code d'initialisation au moment de la création d'un objet ?
10. Pourquoi est-il préférable de toujours déclarer de manière explicite les propriétés d'une classe ?

Retrouvez les réponses du chapitre 5 dans l'annexe A.

CHAPITRE 6

Tableaux en PHP

Le chapitre 3 offrait un aperçu des tableaux en PHP, ou *arrays*, juste suffisant pour évoquer leur puissance. Dans ce chapitre, vous allez découvrir bien d'autres choses à leur propos, dont certaines, si vous avez déjà fait l'expérience de langages fort typés comme C, vous surprendront par leur élégance et leur simplicité.

Les tableaux ont largement participé à la grande popularité de PHP. Non seulement ils éliminent la lourde servitude d'écrire du code pour traiter des structures de données complexes, mais ils fournissent également de nombreuses façons d'accéder aux données tout en demeurant incroyablement rapides.

Accès de base

Nous avons déjà examiné les tableaux comme des groupes de boites d'allumettes collées ensemble. Une autre manière d'imaginer un tableau consiste à le comparer à un collier de perles, où les perles représentent des variables, qui peuvent être numériques, des chaines et même d'autres tableaux. Et ils s'apparentent à des colliers de perles parce que chaque élément possède son propre emplacement et, à l'exception des premier et dernier, ont tous des éléments de chaque côté.

Certains tableaux utilisent comme références des indices numériques; d'autres acceptent des identifiants alphanumériques. Des fonctions intégrées permettent de les trier, de leur ajouter ou supprimer des sections, et de les parcourir, pour gérer chaque élément par l'entremise d'une boucle d'un genre spécial. Enfin, le fait de placer des tableaux dans d'autres tableaux permet d'obtenir des tableaux de deux, trois dimensions ou plus.

Tableaux indexés numériquement

Supposons que vous êtes chargé de créer un site web simple pour une société de distribution de fournitures de bureau et que vous travaillez en ce moment sur la section des papiers de bureau. Une manière de gérer les différents articles de cette catégorie consiste à les placer dans un tableau numérique. L'exemple 6-1 illustre la manière la plus simple de les gérer.

Exemple 6-1. Ajout d'éléments à un tableau

```
<?php
  $papier[] = "Copieur";
  $papier[] = "Jet d'encre";
  $papier[] = "Laser";
  $papier[] = "Photo";

  print_r($papier);
?>
```

Dans cet exemple, lors de chaque ajout d'une valeur au tableau `$papier`, le premier emplacement libre au sein du tableau sert à mémoriser la valeur et un pointeur interne à PHP est incrémenté pour pointer vers l'emplacement libre suivant, prêt pour de futures insertions. La fonction `print_r`, désormais familière, qui affiche le contenu d'une variable, d'un tableau ou d'un objet, permet de vérifier si le tableau est correctement rempli. Elle affiche ce qui suit :

```
Array
(
  [0] => Copieur
  [1] => Jet d'encre
  [2] => Laser
  [3] => Photo
)
```

Nous aurions pu écrire ce programme comme dans l'exemple 6-2, en précisant l'emplacement exact de chaque élément dans le tableau mais, comme vous le voyez, cette approche exige plus de frappes au clavier et complique la maintenance si vous souhaitez insérer ou supprimer des entrées dans le tableau. Donc, à moins que vous souhaitiez imposer un ordre différent, il vaut toujours mieux laisser PHP gérer la numérotation réelle des emplacements.

Exemple 6-2. Ajout d'éléments à des emplacements explicites d'un tableau

```
<?php
  $papier[0] = "Copieur";
  $papier[1] = "Jet d'encre";
  $papier[2] = "Laser";
  $papier[3] = "Photo";

  print_r($papier);
?>
```

Les résultats de sortie de ces deux exemples sont identiques mais, comme `print_r` ne s'utilise généralement pas dans un site web de production, l'exemple 6-3 montre comment afficher les différents types de papiers à l'aide d'une boucle `for`.

Exemple 6-3. Ajout d'éléments à un tableau et lecture de leurs valeurs

```
<?php
  $papier[] = "Copieur";
  $papier[] = "Jet d'encre";
  $papier[] = "Laser";
  $papier[] = "Photo";

  for ($j = 0 ; $j < 4 ; ++$j)
    echo "$j: $papier[$j]<br>";
?>
```

Le script affiche ce qui suit :

```
0: Copieur
1: Jet d'encre
2: Laser
3: Photo
```

Jusqu'ici, vous avez vu deux manières d'ajouter des éléments à un tableau et une possibilité pour y faire référence, mais PHP offre bien plus que cela. Nous y reviendrons sous peu. Avant d'aller plus loin, examinons un autre type de tableau.

Tableaux associatifs

Le fait de suivre les éléments d'un tableau à l'aide d'un indice fonctionne sans problème, mais exige un certain travail de mémoire pour se rappeler à quel indice se trouve tel ou tel élément. De plus, au niveau du suivi par d'autres programmeurs, le code n'est pas très digeste.

C'est là qu'interviennent les tableaux associatifs. Leur utilisation permet de référencer les éléments d'un tableau par un nom au lieu d'un nombre. L'exemple 6-4 reprend et étend le code précédent pour donner à chaque élément du tableau un nom d'identification et une valeur de chaine plus longue et explicite.

Exemple 6-4. Ajout d'éléments à un tableau associatif et lecture de leurs valeurs

```
<?php
  $papier['copieur']  = "Photocopieur et multiusage";
  $papier['jetencre'] = "Imprimante à jet d'encre";
  $papier['laser']    = "Imprimante laser";
  $papier['photo']    = "Papier d'impression photographique";

  echo $papier['laser'];
?>
```

Au lieu d'un nombre (qui ne convoie pas d'information utile, à part sur l'emplacement de l'élément dans le tableau), chaque élément possède cette fois un nom unique qui permet de faire référence à l'élément ailleurs, comme ici, dans une instruction `echo`, qui affiche simplement le libellé complet, `Imprimante laser`. Les noms (`copieur`, `jetencre` et ainsi de suite), sont désignés d'*index* ou de *clés*, tandis que les éléments qui leur sont attribués (comme `Imprimante laser`) s'appellent les *valeurs*.

Cette fonctionnalité très puissante de PHP s'utilise souvent lors de l'extraction d'informations à partir d'un fichier XML ou HTML. Par exemple, un analyseur HTML, comme ceux utilisés par un moteur de recherche, place tous les éléments d'une page web dans un tableau associatif dont les clés reflètent la structure de la page :

```
$html['title'] = "Ma page web";
$html['body']  = "... corps de la page ...";
```

Un tel programme décortiquerait tous les liens trouvés dans une page pour les placer dans un autre tableau, puis tous les titres et les sous-titres dans un autre encore. Le code qui utilise des tableaux associatifs au lieu des tableaux numériques est beaucoup plus facile à écrire et à déboguer.

Affectation à l'aide du mot clé array

Nous avons vu comment attribuer des valeurs dans des tableaux en ajoutant simplement les éléments un à un. Que vous spécifiiez des clés, des identifiants numériques ou laissiez PHP attribuer les indices numériques automatiquement, cette approche demande beaucoup de manipulations. Une méthode plus compacte et rapide pour attribuer des éléments à un tableau fait appel au mot clé `array`. L'exemple 6-5 montre comment utiliser cette méthode pour attribuer des éléments à un tableau, tant numérique qu'associatif.

Exemple 6-5. Ajouter des éléments à un tableau à l'aide du mot clé array

```
<?php
  $p1 = array("Copieur", "Jetencre", "Laser", "Photo");

  echo "Élément p1 : " . $p1[2] . "<br>";

  $p2 = array('copieur'   => "Photocopieur et multiusage",
              'jetencre'  => "Imprimante à jet d'encre",
              'laser'     => "Imprimante laser",
              'photo'     => "Papier d'impression photographique");

  echo "Élément p2 : " . $p2['jetencre'] . "<br>";
?>
```

La première partie de cet extrait ajoute les anciennes descriptions courtes de produits au tableau `$p1`. Il reçoit quatre éléments qui viennent occuper les emplacements `0` à `3`. Donc, l'instruction `echo` affiche ce qui suit :

```
Élément p1 : Laser
```

Ensuite, la seconde partie attribue des identifiants associatifs et leurs descriptions plus longues des produits au tableau `$p2`, suivant la forme *index => valeur*. L'utilisation de l'opérateur `=>` ressemble à l'opérateur d'affectation `=`, sauf qu'ici, vous attribuez une valeur à un *index* et non plus à une *variable*. L'index est alors lié de manière indissociable à cette valeur, à moins qu'il ne reçoive une nouvelle valeur. La commande `echo` affiche donc ce qui suit :

```
Élément p2 : Imprimante à jet d'encre
```


Vous pouvez vérifier que `$p1` et `$p2` correspondent à des types de tableaux différents, parce que les deux commandes suivantes, ajoutées au code, provoquent une erreur `Undefined index` ou `Undefined offset`, car l'identifiant de tableau est incorrect dans les deux cas :

```
echo $p1['jetencre']; // Undefined index - Index non défini
echo $p2[3];          // Undefined offset - Décalage non défini
```

Boucle foreach...as

Les concepteurs de PHP se sont ingéniés à rendre le langage facile à utiliser et, insatisfaits des structures de boucles déjà fournies, ils en ont ajouté une autre, spécialement destinée aux tableaux : la boucle `foreach...as`, qui signifie « pour chaque…en tant que ». Grâce à elle, vous pouvez parcourir chacun des éléments d'un tableau, autrement dit « itérer parmi » les éléments du tableau et effectuer des opérations sur ces éléments, l'un après l'autre.

Le processus débute avec le premier élément et s'achève au dernier, de sorte que vous ne devez même pas connaître le nombre d'éléments présents dans le tableau. L'exemple 6-6 illustre la réécriture de l'exemple 6-3, à l'aide de `foreach...as`.

Exemple 6-6. Parcours d'un tableau numérique à l'aide de foreach…as

```
<?php
  $papier = array("Copieur", "Jetencre", "Laser", "Photo");
  $j = 0;

  foreach($papier as $item)
  {
    echo "$j: $item<br>";
    ++$j;
  }
?>
```

Lorsque PHP rencontre une instruction `foreach`, il prend le premier élément du tableau et le dépose dans la variable qui suit le mot clé `as`. Ensuite, chaque fois que le flux de contrôle retourne dans le `foreach`, PHP place l'élément suivant du tableau dans la variable de l'`as`. Dans l'exemple, la variable `$item` reçoit tour à tour chacune des quatre valeurs du tableau `$papier`. Dès que tous les éléments du tableau ont été utilisés, l'exécution de la boucle se termine. Les résultats de ce code sont exactement les mêmes que ceux de l'exemple 6-3.

Voyons à présent comment fonctionne `foreach` avec un tableau associatif. Examinons l'exemple 6-7, qui réécrit la seconde partie de l'exemple 6-5.

Exemple 6-7. Traversée d'un tableau associatif à l'aide de foreach...as

```
<?php
  $papier = array('copieur'   => "Photocopieur et multiusage",
                  'jetencre'  => "Imprimante à jet d'encre",
                  'laser'     => "Imprimante laser",
                  'photo'     => "Papier d'impression photographique");

  foreach($papier as $item => $description)
    echo "$item : $description<br>";
?>
```

Rappelez-vous que les tableaux associatifs n'exigent pas d'indices numériques, donc la variable `$j` ne sert plus dans cet exemple. Au lieu de cela, chaque élément du tableau `$papier` est inséré dans la paire de clé et valeur des variables `$item` et `$description`, qui viennent ensuite s'afficher. Les résultats de l'affichage de ce code sont les suivants :

```
copieur : Photocopieur et multiusage
jetencre : Imprimante à jet d'encre
laser : Imprimante laser
photo : Papier d'impression photographique
```

En guise de syntaxe alternative à `foreach...as`, vous pouvez aussi utiliser la fonction `list` en conjonction avec la fonction `each`, comme illustré dans l'exemple 6-8.

Exemple 6-8. Traversée d'un tableau associatif à l'aide de list et each

```
<?php
  $papier = array('copieur'   => "Photocopieur et multiusage",
                  'jetencre'  => "Imprimante à jet d'encre",
                  'laser'     => "Imprimante laser",
                  'photo'     => "Papier d'impression photographique");

  while (list($item, $description) = each($papier))
    echo "$item : $description<br>";
?>
```

Le code de cet exemple définit une boucle `while` qui continue de boucler jusqu'à ce que `each` renvoie la valeur `FALSE`. La fonction `each` opère comme `foreach` : elle renvoie un tableau avec une paire clé-valeur à partir du tableau `$papier`, puis déplace le pointeur intégré à la paire suivante dans ce tableau. Lorsqu'il n'y a plus aucune paire à retourner, `each` renvoie `FALSE`.

La fonction `list` prend en argument un tableau, soit dans ce cas-ci une paire clé-valeur renvoyée par la fonction `each`, et affecte ces valeurs de tableau aux variables contenues entre ses parenthèses.

Vous pouvez voir le mode de fonctionnement de `list` plus en détail dans l'exemple 6-9, qui crée un tableau à partir de deux chaines, `Alice` et `Julie`, affectées à la fonction `list`, qui redistribue ces deux chaines en guise de valeurs aux variables `$a` et `$b`.

Exemple 6-9. Utilisation de la fonction list


```
<?php
  list($a, $b) = array('Alice', 'Julie');
  echo "a=$a b=$b";
?>
```

Le résultat de l'affichage est le suivant :

```
a=Alice b=Julie
```

Donc, vous pouvez trouver votre bonheur lorsque vous devez parcourir des tableaux : utilisez `foreach...as` pour construire une boucle qui extrait les valeurs et les distribue tour à tour à la variable placée après `as`, ou alors, utilisez la fonction `each` pour créer votre propre boucle, en combinaison avec `list`.

Tableaux multidimensionnels

Une fonctionnalité de conception simple dans la syntaxe des tableaux en PHP permet de créer des tableaux de plusieurs dimensions. En fait, ils peuvent être de n'importe quelle dimension, quoique les applications à plus de trois dimensions soient assez rares.

Cette possibilité consiste à inclure un tableau dans un autre et de poursuivre ainsi de proche en proche, quasi à l'infini. Voyons comment cela fonctionne avec le tableau associatif de l'exemple précédent et étendons-le (exemple 6-10).

Exemple 6-10. Création d'un tableau associatif multidimensionnel

```
<?php
  $produits = array(

    'papier' => array(

      'copieur'   => "Photocopieur et multiusage",
      'jetencre'  => "Imprimante à jet d'encre",
      'laser'     => "Imprimante laser",
      'photo'     => "Papier d'impression photographique"),

    'stylos' => array(

      'bille'     => "Stylos à bille",
      'fluo'      => "Surligneurs",
      'feutre'    => "Feutres et marqueurs"),

    'divers' => array(

      'rubans'    => "Rubans adhésifs",
      'colle'     => "Colles",
      'attache'   => "Attaches-trombonnes"
    )
  );

  echo "<pre>";
```

```
  foreach($produits as $section => $items)
    foreach($items as $cle => $valeur)
      echo "$section :\t$cle\t($valeur)<br>";

  echo "</pre>";
?>
```

Comme il commence à grossir un peu, j'ai renommé quelques éléments pour éclaircir le code. Par exemple, comme le tableau `$papier` précédent n'est plus qu'une sous-section d'un plus gros tableau, j'ai appelé le tableau principal `$produits`. Dans ce tableau apparaissent trois éléments, `papier`, `stylos` et `divers`, qui contiennent chacun un autre tableau de paires clé-valeur.

Si nécessaire, ces sous-tableaux pourraient eux-mêmes contenir d'autres tableaux. Ainsi, sous `bille`, il aurait pu y avoir des stylos de tailles, de types et de couleurs différents dans le magasin en ligne. Mais pour l'instant, je restreins le code à la profondeur deux.

Une fois les données affectées aux tableaux, j'utilise une paire de boucles `foreach...as` pour afficher les différentes valeurs. La boucle extérieure extrait les sections principales du niveau supérieur du tableau, tandis que la boucle intérieure extrait les paires clé-valeur des catégories au sein de chaque section.

À partir du moment où vous comprenez bien que chaque niveau du tableau fonctionne de la même manière, c'est-à-dire que c'est une paire clé-valeur, vous pouvez rédiger du code pour accéder à n'importe quel élément, à n'importe quel niveau.

L'instruction `echo` exploite le caractère d'échappement `\t` de PHP, qui affiche une tabulation. Bien que les tabulations ne soient normalement pas prises en compte par le navigateur web, je les conserve pour améliorer la présentation mais je les entoure de balises `<pre>` et `</pre>`, qui indiquent au navigateur de mettre le texte en forme avec une police à espacement constant et de *ne pas* ignorer les caractères d'espacement comme les tabulations et les retours à la ligne. Le résultat de ce code devrait ressembler au suivant :

```
papier :  copieur   (Photocopieur et multiusage)
papier :  jetencre  (Imprimante à jet d'encre)
papier :  laser     (Imprimante laser)
papier :  photo     (Papier d'impression photographique)
stylos :  bille     (Stylos à bille)
stylos :  fluo      (Surligneurs)
stylos :  feutre    (Feutres et marqueurs)
divers :  rubans    (Rubans adhésifs)
divers :  colle     (Colles)
divers :  attache   (Attaches-trombonnes)
```

Pour accéder directement à un élément de l'ensemble du tableau, utilisez des crochets :

```
echo $produits['divers']['colle'];
```

Le résultat donne la valeur `Colles`.

Vous pouvez aussi créer des tableaux numériques multidimensionnels accessibles par des indices au lieu d'identifiants alphanumériques. L'exemple 6-11 crée le plateau d'un jeu d'échec avec les pièces dans leurs emplacements de départ.

Exemple 6-11. Création d'un tableau numérique multidimensionnel

```
<?php
  $plateau = array(
    array('t', 'c', 'f', 'q', 'r', 'f', 'c', 't'),
    array('p', 'p', 'p', 'p', 'p', 'p', 'p', 'p'),
    array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
    array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
    array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
    array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
    array('P', 'P', 'P', 'P', 'P', 'P', 'P', 'P'),
    array('T', 'C', 'F', 'Q', 'R', 'F', 'C', 'T')
  );

  echo "<pre>";

  foreach($plateau as $rangee)
  {
    foreach ($rangee as $piece)
      echo "$piece ";

    echo "<br>";
  }

  echo "</pre>";
?>
```

Dans cet exemple, les lettres en bas de casse représentent les pièces noires et celles en capitales, les blanches. Les lettres signifient `t` = tour, `c` = cavalier, `f` = fou, `q` = reine, `r` = roi et `p` = pion. Encore une fois, deux boucles `foreach...as` parcourent le tableau et affichent le contenu. La boucle extérieure traite chaque rangée pour la placer dans la variable `$rangee`, qui est elle-même un tableau, puisque le tableau `$plateau` inclut un tableau dans chaque rangée. Cette boucle comporte un bloc de deux instructions, donc les accolades sont obligatoires pour les entourer.

La boucle intérieure traite chaque case dans une rangée, affichant le caractère (`$piece`) qui y est mémorisé, suivi d'une espace (pour donner à l'affichage l'aspect d'un carré). Le corps de cette boucle comporte une seule instruction, donc les accolades ne sont pas indispensables. Les balises `<pre>` et `</pre>` garantissent le bon affichage, comme suit :

```
t c f q r f c t
p p p p p p p p

p p p p p p p p
T C F Q R F C T
```

Et, à nouveau, pour accéder directement à une case au sein de ce tableau de tableaux, utilisez les crochets :

```
echo $plateau[7][3];
```

Cette instruction affiche la lettre capitale `Q`, à la huitième rangée en commençant par le haut, et le quatrième élément en commençant par la gauche. Est-il nécessaire de rappeler que les indices commencent à 0 et non 1 ?

Utiliser les fonctions de tableaux

Nous avons vu les fonctions `list` et `each` mais PHP propose de très nombreuses fonctions de manipulation des tableaux. Leur liste complète est disponible sur *http://php.net/manual/fr/ref.array.php*. Cependant, certaines de ces fonctions sont tellement fondamentales que cela vaut la peine de prendre le temps de les examiner en détail.

Est-ce un tableau : is_array?

Les tableaux et les variables partagent le même espace de nommage, ce qui signifie qu'il n'est pas permis de disposer d'une variable nommée `$eric` et d'un tableau portant le même nom, `$eric`. Si vous doutez devant une variable et si votre code doit vérifier que cette variable est un tableau, utilisez la fonction `is_array`, comme suit :

```
echo (is_array($eric)) ? "Est un tableau" : "N'est pas un tableau";
```

Remarquez que, si `$eric` n'a pas encore reçu de valeur, alors le message `Undefined variable` apparait.

Compteur : count

Si les fonctions `each` et `foreach...as` constituent d'excellents moyens pour parcourir le contenu d'un tableau, il est parfois nécessaire de connaitre exactement le nombre d'éléments présents dans un tableau, surtout si vous comptez en référencer directement les éléments. Pour compter tous les éléments au niveau le plus élevé d'un tableau, utilisez une commande du genre :

```
echo count($eric);
```

Si vous devez connaitre le nombre total d'éléments dans un tableau multidimensionnel, utilisez une instruction du genre :

```
echo count($eric, 1);
```

Le second paramètre est facultatif et définit le mode d'utilisation. Il contient `0` pour limiter le décompte au seul niveau supérieur, ou `1` pour imposer le comptage récursif de tous les sous-tableaux.

Tri : sort

Le tri est une opération si usuelle que PHP propose une fonction intégrée à cet égard. Dans sa forme la plus simple, utilisez-la comme suit :

```
sort($eric);
```

Au contraire de quelques autres fonctions, `sort` agit directement sur le tableau fourni, au lieu de retourner un nouveau tableau avec les éléments triés. Elle renvoie `TRUE` lorsqu'elle a réussi et `FALSE` en cas d'erreur. Elle prend en charge également un certain nombre d'indicateurs, mais les deux principaux à connaitre imposent le tri selon des valeurs numériques ou selon des valeurs de chaine (tri alphabétique), comme suit :

```
sort($eric, SORT_NUMERIC);
sort($eric, SORT_STRING);
```

Il est également possible d'inverser l'ordre de tri à l'aide de la fonction `rsort` (*reverse sort*), comme ceci :

```
rsort($eric, SORT_NUMERIC);
rsort($eric, SORT_STRING);
```

Mélanger : shuffle

Comme dans le cas du mélange d'un jeu de cartes, il est parfois nécessaire de disposer les éléments d'un tableau selon un ordre complètement aléatoire :

```
shuffle($cartes);
```

De même que `sort`, `shuffle` intervient directement sur le tableau fourni et renvoie `TRUE` en cas de réussite ou `FALSE` en cas d'erreur.

Éclater : explode

Cette fonction très pratique permet de prendre une chaine contenant une suite de caractères séparés par un caractère déterminé (ou une autre chaine de séparation), puis de disposer chacun de ces éléments dans un tableau. Un exemple classique consiste à éclater une phrase en un tableau contenant tous les mots, comme dans l'exemple 6-12.

Exemple 6-12. Éclater une chaine de mots séparés par une espace en un tableau

```
<?php
  $temp = explode(' ', "Ceci est une phrase de sept mots");
  print_r($temp);
?>
```

L'exemple affiche ce qui suit (sur une seule ligne lorsque vu dans le navigateur) :

```
Array
(
  [0] => Ceci
  [1] => est
  [2] => une
  [3] => phrase
```

```
    [4] => de
    [5] => sept
    [6] => mots
)
```

Le premier paramètre désigne le séparateur, qui ne se limite pas à l'espace ni à un seul caractère, comme le démontre la petite variante de l'exemple 6-13.

*Exemple 6-13. Éclater une chaine de mots séparés par *** en un tableau*

```
<?php
  $temp = explode('***', "Une***phrase***avec***astérisques");
  print_r($temp);
?>
```

Le code de l'exemple 6-13 affiche ce qui suit :

```
Array
(
  [0] => Une
  [1] => phrase
  [2] => avec
  [3] => astérisques
)
```

Extraire : extract

Il est parfois bien pratique de pouvoir convertir des paires de clés-valeurs d'un tableau en des variables PHP. C'est le cas, notamment, lorsque vous devez traiter les variables `$_GET` et `$_POST`, telles qu'elles sont envoyées à un script PHP par un formulaire.

Lors de la soumission d'un formulaire à travers le web, le serveur décompresse les variables dans un tableau global à l'usage du script en PHP. Si les variables ont été envoyées par la méthode Get, elles viennent se placer dans un tableau associatif dénommé `$_GET` ; si elles l'ont été par la méthode Post, elles viennent se placer dans un autre tableau associatif dénommé `$_POST`.

Vous pourriez bien entendu parcourir de tels tableaux associatifs selon les techniques illustrées précédemment dans les exemples. Cependant, dans certains cas, vous voulez simplement stocker les valeurs reçues sous forme de variables pour un usage ultérieur. Dans ces cas-là, vous pouvez demander à PHP d'exécuter cette opération automatiquement :

```
extract($_GET);
```

Selon ce mécanisme, si dans le tableau associatif figure une clé `q` qui correspond au paramètre de formulaire `q`, et une valeur associée `Bonjour`, alors PHP crée une variable `$q` et lui donne cette valeur.

Demeurez toujours très prudent avec cette approche, car si une des variables extraites entre en conflit avec l'une de celles que votre programme a déjà définies, celles-ci seront écrasées.

Pour éviter ce risque, faites appel aux nombreux paramètres supplémentaires disponibles pour cette fonction, comme suit :

```
extract($_GET, EXTR_PREFIX_ALL, 'fromget');
```

Avec ce paramètre `EXTR_PREFIX_ALL`, suivi d'un préfixe sous forme de chaine, toutes les nouvelles variables commencent par la chaine de préfixe, suivie d'un soulignement, puis le nom d'origine de la variable. Ainsi, dans l'exemple, la variable `$q` devient `$fromget_q`. Je vous encourage fortement à préférer cette version de la fonction lorsque vous traitez des tableaux `$_GET` et `$_POST`, ou tout autre tableau dont les clés sont contrôlables par l'utilisateur, parce que des visiteurs malintentionnés pourraient soumettre des clés conçues délibérément pour écraser des noms de variables communément utilisés et, ainsi, réussir à compromettre votre site web.

Compresser : compact

Si vous connaissez désormais la fonction `extract`, vous utiliserez parfois la fonction inverse, `compact`, pour créer un tableau à partir de variables et de leurs valeurs. L'exemple 6-14 illustre l'utilisation de cette fonction.

Exemple 6-14. Utilisation de la fonction compact

```
<?php
  $fname        = "Doctor";
  $sname        = "Who";
  $planet       = "Gallifrey";
  $system       = "Gridlock";
  $constellation = "Kasterborous";

  $contact = compact('fname', 'sname', 'planet', 'system', 'constellation');

  print_r($contact);
?>
```

L'exécution de ce script affiche ce qui suit :

```
Array
(
  [fname] => Doctor
  [sname] => Who
  [planet] => Gallifrey
  [system] => Gridlock
  [constellation] => Kasterborous
)
```

Remarquez que `compact` exige que les noms des variables soient fournis entre apostrophes et non préfixés du symbole `$`, car `compact` recherche une liste de noms de variables.

Une autre utilisation de cette fonction se situe dans le débogage, lorsque vous voulez rapidement voir le contenu de plusieurs variables, comme dans l'exemple 6-15.

Exemple 6-15. Utilisation de compact pour faciliter le débogage

```
<?php
  $j       = 23;
  $temp    = "Bonjour";
  $adresse = "1 Rue Vieille";
  $age     = 61;

  print_r(compact(explode(' ', 'j temp adresse age')));
?>
```

Cette technique fonctionne à l'aide de la fonction `explode` pour extraire tous les mots de la chaine fournie et les déposer dans un tableau. Celui-ci est transmis à la fonction `compact`, qui renvoie à son tour un tableau à `print_r`, qui en affiche le contenu.

Lorsque vous copiez-collez la ligne de code du `print_r`, il vous suffit de modifier les noms des variables cités pour obtenir rapidement l'affichage du contenu d'un groupe de variables. Cet exemple produit l'affichage suivant :

```
Array
(
  [j] => 23
  [temp] => Bonjour
  [addresse] => 1 Rue Vieille
  [age] => 61
)
```

Réinitialiser : reset

Lorsque la construction `foreach...as` ou la fonction `each` parcourt un tableau, elle conserve un pointeur PHP interne qui désigne l'élément du tableau qu'il lui faut renvoyer ensuite. Si votre code doit revenir au début d'un tableau, utilisez `reset`, qui renvoie aussi la valeur de cet élément. Voici deux exemples d'utilisation de cette fonction :

```
reset($eric);          // Éluder la valeur de retour
$item = reset($eric); // Conserver le premier élément du tableau dans $item
```

Aller à la fin : end

Comme pour `reset`, vous pouvez déplacer le pointeur interne de PHP vers le dernier élément d'un tableau à l'aide de la fonction `end`, qui renvoie aussi la valeur de ce dernier élément et s'utilise comme dans les exemples suivants :

```
end($eric);
$item = end($eric);
```

Ce chapitre conclut l'introduction aux bases de PHP. Vous êtes déjà en mesure de rédiger des programmes assez complexes à partir des compétences que vous avez acquises. Au chapitre suivant, nous verrons comment utiliser PHP pour des tâches usuelles et pratiques.

Questions

1. Quelle est la différence entre un tableau numérique et un tableau associatif ?
2. Quel est le principal avantage du mot clé `array` ?
3. Quelle est la différence entre `foreach` et `each` ?
4. Comment fait-on pour créer un tableau multidimensionnel ?
5. Comment pouvez-vous déterminer le nombre d'éléments dans un tableau ?
6. Quel est le but de la fonction `explode` ?
7. Comment faites-vous pour ramener le pointeur interne de PHP vers un élément d'un tableau vers le tout premier élément de ce tableau ?

Retrouvez les réponses du chapitre 6 dans l'annexe A.

CHAPITRE 7

PHP en pratique

Les chapitres précédents ont parcouru les éléments du langage PHP. Ce chapitre se fonde sur les connaissances que vous avez acquises pour vous montrer quelques tâches pratiques et usuelles tout aussi importantes. Il s'agit ici d'apprendre les meilleurs moyens de gérer les chaines de caractères pour obtenir un code clair et concis, capable de les afficher dans un navigateur exactement comme vous le voulez, notamment au niveau de la gestion des dates et heures. Vous apprendrez également à créer et modifier des fichiers, en particulier ceux téléchargés par les utilisateurs.

Utiliser printf

Vous avez déjà utilisé les fonctions `print` et `echo`, qui sortent simplement du texte pour l'afficher dans le navigateur. Une autre fonction, bien plus puissante, `printf`, contrôle le format des sorties à partir de caractères de mise en forme, de *formatage*, dans une chaine. Pour chaque caractère de formatage, `printf` attend un argument correspondant à afficher sous ce format. Ainsi, l'exemple suivant emprunte le spécificateur de conversion `%d` pour afficher la valeur `3` en décimal :

```
printf("Votre panier d'achat contient %d articles", 3);
```

Si vous remplacez le `%d` par `%b`, alors la valeur `3` s'affiche en binaire, soit `11`. Le tableau 7-1 énumère les spécificateurs de conversion pris en charge.

Tableau 7-1. Les spécificateurs de conversion de printf

Spécificateur	Action de conversion sur l'argument arg	Exemple (pour arg = 123)
%	Affiche le caractère % (aucun arg nécessaire)	%
b	Affiche arg sous forme d'un entier binaire	123.000000
c	Affiche le caractère ASCII pour l'arg	{
d	Affiche arg sous forme d'un entier décimal signé	123
e	Affiche arg selon la notation scientifique	1.23000e+2
f	Affiche arg en virgule flottante	1111011
o	Affiche arg sous forme d'entier en octal	173
s	Affiche arg sous forme de chaine (s comme string)	123
u	Affiche arg sous forme décimale non signée	123
x	Affiche arg en hexadécimal et en bas de casse	7b
X	Affiche arg en hexadécimal et en capitales	7B

La fonction printf accepte autant de spécificateurs que vous le souhaitez, à condition que vous fournissiez le nombre d'arguments correspondants et de préfixer chaque spécificateur du symbole %. Ainsi, la ligne de code suivante est valable et renvoie "Mon nom est Simon. J'ai 33 ans, ce qui équivaut à 21 en hexadécimal":

```
printf("Mon nom est %s. J'ai %d ans, ce qui équivaut à %X en hexadécimal",
  'Simon', 33, 21);
```

Si vous oubliez un des arguments, vous recevez un message d'erreur d'analyse qui indique qu'une parenthèse droite inattendue a été rencontrée.

Un autre exemple beaucoup plus pratique d'application de printf consiste à définir une couleur en HTML à l'aide de valeurs décimales. Prenons le cas d'une couleur définie par un triplet de valeurs, en pratique de 65 de rouge, de 127 de vert et de 245 de bleu, le tout en décimal. Vous ne voulez pas effectuer la conversion en hexadécimal vous-même, donc voici une solution bien pratique :

```
printf("<span style='color:#%X%X%X'>Bonjour</span>", 65, 127, 245);
```

Examinez attentivement la spécification de la couleur entre les apostrophes (''). D'abord apparait le dièse (#), appelé aussi «*hash*», attendu par la spécification de couleur. Ensuite apparaissent trois fois le spécificateur de format %X, un pour chacun de vos nombres. Le résultat de sortie de cette commande est le suivant :

```
<span style='color:#417FF5'>Bonjour</span>
```

En corolaire, vous verrez qu'il est bien pratique d'emprunter des variables ou des expressions en arguments de printf. Ainsi, si vous avez stocké les valeurs de rouge, vert et bleu dans trois variables $r, $v et $b, respectivement, il est alors facile de demander une couleur légèrement plus sombre, comme suit :

```
printf("<span style='color:#%X%X%X'>Bonjour</span>", $r-20, $v-20, $b-20);
```

Réglage de précision

Non seulement vous pouvez spécifier un type de conversion, mais vous pouvez également régler la précision des résultats affichés. Les montants en devises, par exemple, s'affichent habituellement avec seulement deux chiffres après le point décimal. Cependant, une valeur calculée peut contenir une précision plus importante. Ainsi, 123.42 / 12 donne 10.285. Pour faire en sorte que de telles valeurs soient mémorisées avec leur précision complète mais qu'elles ne s'affichent qu'avec deux décimales, insérez la chaine ".2" entre le % et le spécificateur de précision, comme suit :

```
printf("Le résultat est de : %.2f $", 123.42 / 12);
```

Et l'affichage donne :

```
Le résultat est de : 10.29 $
```

Les possibilités de contrôle vont bien au-delà de ceci car vous pouvez aussi décider de remplir les sorties avec des zéros ou des espaces. Il suffit de préfixer le spécificateur de certaines valeurs. L'exemple 7-1 illustre quatre combinaisons possibles.

Exemple 7-1. Réglages de précision

```
<?php
  echo "<pre>"; // Permet l'affichage des espaces

  // Calibrage à 15 espaces
  printf("Le résultat est de %15f $\n", 123.42 / 12);

  // Cal. à 15 espaces, remplissage avec des zéros
  printf("Le résultat est de %015f $\n", 123.42 / 12);

  // Cal. à 15 espaces, précision de 2 décimales
  printf("Le résultat est de %15.2f $\n", 123.42 / 12);

  // Cal. à 15 espaces, précision de 2 décimales, remplissage de zéros
  printf("Le résultat est de %015.2f $\n", 123.42 / 12);

  // Cal. à 15 espaces, précision de 2 décimales, remplissage de symboles #
  printf("Le résultat est de %'#15.2f $\n", 123.42 / 12);
?>
```

Le programme de cet exemple donne les résultats suivants :

```
Le résultat est de       10.285000 $
Le résultat est de 00000010.285000 $
Le résultat est de            10.29 $
Le résultat est de 000000000010.29 $
Le résultat est de ##########10.29 $
```

Le fonctionnement devient facile à comprendre dès que vous acquérez le réflexe d'analyser les spécificateurs de droite à gauche (tableau 7-2) et que vous éludez le signe de la devise, dans ce cas-ci $. Remarquez que :

- Le caractère le plus à droite est le spécificateur de conversion, soit f pour la virgule (point) flottante.
- Juste avant le spécificateur de conversion, s'il y a un point et un nombre associé, ils représentent la précision en nombre de décimales après le point.
- Qu'il y ait ou non de spécificateur de précision, s'il y a un nombre, celui-ci indique le nombre de caractères avec lequel la sortie sera calibrée. Dans l'exemple, ce calibrage est effectué à 15 caractères. Si la sortie a une longueur égale ou supérieure à ce calibrage, alors cet argument est ignoré.
- Le paramètre le plus à gauche permis après le symbole %, comme dans les deuxième et quatrième exemples, est 0, et est ignoré, sauf si une valeur de calibrage est définie, auquel cas, la sortie est calibrée avec un remplissage par des zéros au lieu d'espaces. Enfin, si le caractère de remplissage nécessaire au calibrage diffère de l'espace ou de zéro, utilisez celui de votre choix mais indiquez-le avec une apostrophe en préfixe, comme ceci : '#.
- Le plus à gauche apparait le symbole %, qui amorce la conversion.

Tableau 7-2. Composantes de spécificateur de conversion

Début de conversion	Caractère de remplissage	Caractère calibrés	Précision d'affichage	Spécificateur de conversion	Exemple
%		15		f	10.285000
%	0	15	.2	f	000000000010.29
%	'#	15	.4	f	########10.2850

Calibrage de chaine

Les chaines aussi bénéficient de possibilités de calibrage à des longueurs définies et avec des caractères de remplissage, mais il est également possible de les justifier à gauche ou à droite. L'exemple 7-2 illustre quelques-unes de ces possibilités.

Exemple 7-2. Calibrages de chaine

```
<?php
  echo "<pre>"; // Permet l'affichage des espaces

  $h = 'Rasmus';

  printf("[%s]\n",        $h); // Sortie normale de la chaine
  printf("[%12s]\n",      $h); // Justification à droite avec des espaces
  printf("[%-12s]\n",     $h); // Justification à gauche avec des espaces
  printf("[%012s]\n",     $h); // Remplissage de zéros
  printf("[%'#12s]\n\n",  $h); // Utilise le caractère de remplissage '#'

  $d = 'Rasmus Lerdorf';

  printf("[%12.8s]\n",    $d); // Justification droite, coupure à 8 car.
  printf("[%-12.12s]\n",  $d); // Justification gauche, coupure à 12 car.
  printf("[%-'@12.10s]\n", $d); // Justification gauche, car. '@', coupure à 10 car.
?>
```

Remarquez encore une fois que, pour améliorer la présentation dans une page web, la balise HTML `<pre>` permet de préserver les espaces et le caractère `\n` de nouvelle ligne lors de l'affichage. Les résultats de l'exécution de ce code donnent :

```
[Rasmus]
[      Rasmus]
[Rasmus      ]
[000000Rasmus]
[######Rasmus]

[    Rasmus L]
[Rasmus Lerdo]
[Rasmus Ler@@]
```


Lorsque vous précisez une valeur de calibrage, PHP l'ignore si la chaine a une longueur égale ou supérieure à cette valeur, *sauf* si vous indiquez une valeur de coupure pour réduire la longueur de la chaine du nombre de caractères à cette valeur de coupure, qui doit être inférieure à la valeur de calibrage.

Le tableau 7-3 détaille les composantes disponibles pour les spécificateurs de conversion de chaine.

Tableau 7-3. Composantes de spécificateur de conversion de chaine

Début de conversion	Justification gauche/droite	Car. de calibrage	Caractères calibrés	Précision d'affichage	Spécificateur de conversion	Exemple (avec "Ramus")
%					S	[Rasmus]
%		0	10		S	[Rasmus]
%		'#	8	.4	S	[####Rasm]

Utiliser sprintf

La fonction `sprintf` fonctionne selon le même principe de formatage de sortie que `printf` mais, au lieu de sortir les résultats dans le navigateur, elle permet d'affecter les résultats à une variable pour un usage ultérieur.

Ainsi, vous pouvez l'utiliser pour effectuer une conversion, illustrée dans l'exemple suivant, qui renvoie sous forme d'une chaine la valeur hexadécimale du groupe de couleurs RVB 65, 127, 245 dans `$hexstring` :

```
$hexstring = sprintf("%X%X%X", 65, 127, 245);
```

La mémorisation de la sortie dans une variable permet l'affichage par la suite :

```
$out = sprintf("Le résultat est : $%.2f", 123.42 / 12);
echo $out;
```

Fonctions de dates et heures

Pour tenir compte de la date et de l'heure, PHP utilise des horodatages (*timestamps*) standards d'Unix, formés simplement du nombre de secondes écoulées depuis le 1er janvier 1970 à 00:00. Pour connaitre le timestamp courant, utilisez la fonction `time` :

```
echo time();
```

Comme cette valeur est comptée en secondes, pour déterminer le timestamp de la même heure la semaine suivante, il suffit d'ajouter 7 jours fois 24 heures, fois 60 minutes et fois 60 secondes à la valeur connue :

```
echo time() + 7 * 24 *60 * 60;
```

Pour créer un timestamp pour une date donnée, utilisez la fonction `mktime`. Elle renvoie par exemple l'horodatage `946684800` pour la première seconde de la première minute de la première heure du premier jour de l'an 2000 :

```
echo mktime(0, 0, 0, 1, 1, 2000);
```

Les paramètres à lui passer dans l'ordre, de gauche à droite, sont :

- le nombre d'heures (de 0 à 23);
- le nombre de minutes (de 0 à 59);
- le nombre de secondes (de 0 à 59);
- le nombre de mois (de 1 à 12);
- le nombre de jours (de 1 à 31);
- l'année (de 1970 à 2038, ou de 1901 à 2038 avec PHP 5.1.0 ou plus, sur un système signé à 32 bits).

Vous vous demandez peut-être pourquoi ces dates de 1970 à 2038? Tout simplement parce que les premiers concepteurs d'Unix ont choisi l'année 1970 comme date de base, avant laquelle ils ont considéré qu'aucun programmeur ne devrait aller!

Heureusement, comme PHP (à partir de la version 5.1.0) prend en charge des systèmes avec des entiers signés de 32 bits pour les horodatages, les dates de 1901 à 2038 sont autorisées sur ceux-ci. Cela introduit toutefois un problème pire que l'original, puisque les concepteurs d'Unix ont aussi décidé que personne n'utiliserait Unix au bout de quelque 70 ans, donc ils ont cru qu'ils pouvaient se contenter de mémoriser les horodatages sous forme d'une valeur en 32 bits, qui arrivera en dépassement de capacité le 19 janvier 2038!

Cela entrainera ce que l'on appelle le bogue Y2K38 (ou de l'année 2038, un peu comme le bogue de l'an 2000, généré par l'usage habituel d'années à deux chiffres, qui a également dû être réglé). PHP a introduit la classe `DateTime` à partir de sa version 5.2 pour contourner ce problème, mais elle ne fonctionne que sur les architectures à 64 bits.

Pour afficher la date, utilisez la fonction `date`, qui prend en charge une pléthore d'options de formatage pour afficher la date comme vous le souhaitez. Sa syntaxe est la suivante :

```
date($format, $timestamp);
```

Le paramètre `$format` est une chaine contenant des spécificateurs de mise en forme, tels que détaillés au tableau 7-4, tandis que `$timestamp` est un timestamp Unix. Pour la liste complète des spécificateurs, consultez *http://php.net/manual/fr/function.date.php*. La commande suivante affiche la date et l'heure courantes selon le format anglais `"Thursday July 6th, 2017 - 1:38pm"` :

```
echo date("l F jS, Y - g:ia", time());
```

Pour afficher la date et l'heure selon un format plus conventionnel en français (`"6/7/2017 13:38"`), entrez la commande suivante :

```
echo date("j/n/Y h:i", time());
```


Tableau 7-4. Les principaux spécificateurs de format de la fonction date

Format	Description	Valeur retournée
Spécificateurs de jour		
d	Jour du mois, 2 chiffres, zéros en tête	01 à 31
D	Jour de la semaine, trois lettres	Mon à Sun
j	Jour du mois, pas de zéro en tête	1 à 31
l	(petit L) Jour de semaine, en toutes lettres (anglais)	Sunday à Saturday
N	Jour de la semaine, numérique, du lundi au dimanche	1 à 7
S	Suffixe anglais du jour du mois (utilisé avec j)	st, nd, rd ou th
w	Jour de la semaine, numérique, du dimanche au samedi	0 à 6
z	Jour de l'année	1 à 365
Spécificateur de semaine		
W	Numéro de semaine dans l'année	01 à 52
Spécificateur du mois		
F	Nom anglais du mois	January à December
m	Numéro du mois, zéros en tête	01 à 12
M	Nom abrégé anglais du mois	Jan à Dec
n	Numéro du mois, sans zéro en tête	1 à 12
t	Nombre de jours dans le mois	28 à 31
Spécificateurs d'année		
L	Année bissextile	1 = oui, 0 = non
y	Année à deux chiffres	00 à 99
Y	Année à quatre chiffres	0000 à 9999
Spécificateurs d'heure		
a	En bas de casse : avant-midi ou après-midi	am ou pm
A	En capitales : avant-midi ou après-midi	AM ou PM
g	Heure du jour, format 12 heures, sans zéro en tête	1 à 12
G	Heure du jour, format 24 heures, sans zéro en tête	0 à 23
h	Heure du jour, format 12 heures, zéros en tête	00 à 12
H	Heure du jour, format 24 heures, zéros en tête	00 à 23
i	Minutes, zéros en tête	00 à 59
s	Secondes, zéros en tête	00 à 59

La question se pose alors, de connaitre le jour de la semaine en français à partir d'un horodatage. Voici une petite solution toute simple, qui fait appel à un tableau numérique avec les jours de la semaine et le spécificateur `w` de `date`, pour connaitre l'indice, de 0 à 6 :

```
$dateheure = time();
$jds = array('dimanche', 'lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi', 'samedi');
$jourdesemaine = $jds[date("w", $dateheure)];
echo $jourdesemaine;
```

Le spécificateur `n` donne le numéro du mois, de 1 à 12. Vous pouvez ainsi déterminer le mois de la même manière mais n'oubliez pas de soustraire 1 à la valeur retournée par `date`, puisque les indices commencent à 0. Le même principe s'applique aux jours de la semaine.

Constantes de date

Un certain nombre de constantes sont définies pour `date`, afin d'accéder plus rapidement à des formats utilisés par des services spécifiques. Ainsi, `date(DATE_RSS)` renvoie la date et l'heure courantes dans un format reconnu pour un fil RSS. La liste suivante décrit quelques constantes les plus fréquemment utilisées :

`DATE_ATOM`
: C'est le format des fils de type Atom. Le format PHP équivalent est `"Y-m-d\TH:i:sP"` ce qui donne en sortie par exemple `"2018-08-16T12:00:00+00:00"`.

`DATE_COOKIE`
: Le format défini pour un *cookie* (témoin de connexion) à partir d'un serveur web ou en JavaScript. Le format PHP équivalent est `"l, d-M-y H:i:s T"`, ce qui donne par exemple `"Thursday, 16-Aug-18 12:00:00 UTC"`.

`DATE_RSS`
: Le format des fils RSS. Le format PHP équivalent est `"D, d M Y H:i:s O"`, ce qui donne par exemple `"Thu, 16 Aug 2018 12:00:00 UTC"`.

`DATE_W3C`
: Le format du *World Wide Web Consortium*, ou W3C. Le format PHP équivalent est `"Y-m-d\TH:i:sP"`, ce qui donne par exemple `" 2018-08-16T12:00:00+00:00"`.

La liste complète est disponible sur *http://php.net/manual/fr/class.datetime.php*.

Utiliser checkdate

Nous venons de voir comment afficher une date valide selon toutes sortes de formats. À l'inverse, comment vérifier qu'un utilisateur a entré une date valide dans le programme ? La réponse consiste à passer le mois, le jour et l'année à la fonction `checkdate`, qui renvoie `TRUE` si la date est valable ou `FALSE` sinon.

Ainsi, si l'utilisateur entre un 30 février, quelle que soit l'année, ce sera toujours une date non valide. L'exemple 7-3 montre comment tirer parti de cette fonction. Avec les données définies, la date est non valide.

Exemple 7-3. Vérification de la validité d'une date

```
<?php
  $mois  = 9;    // Septembre (n'a que 30 jours)
  $jour  = 31;   // 31
  $annee = 2018; // 2018
  if (checkdate($mois, $jour, $annee)) echo "Date valide";
  else echo "Date non valide";
?>
```


Gestion de fichiers

Aussi puissant soit-il, MySQL n'est pas le seul (ni nécessairement le meilleur) moyen de stocker toutes les données sur un serveur web. Il est quelquefois plus rapide et convivial d'accéder directement à des fichiers sur le disque dur. Parmi les cas où vous risquez de devoir faire ce genre de chose, citons la modification d'images, comme les avatars chargés par un utilisateur, et la gestion de fichiers journaux à traiter.

Tout d'abord, nous devons prendre un fait en compte à propos des dénominations de fichiers : lorsque vous rédigez du code destiné à une installation sur des systèmes PHP différents, il n'est pas possible de savoir si ces systèmes sont sensibles à la casse. Par exemple, les fichiers Windows et Mac OS X ne sont pas sensibles à la casse, tandis que ceux sous Linux et Unix le sont. Par conséquent, vous devez toujours partir du principe que le système est sensible à la casse et vous devez respecter des conventions, comme de n'utiliser que des noms de fichiers en bas de casse.

Vérifier si un fichier existe

Pour déterminer si un fichier existe, utilisez la fonction `file_exists`, qui renvoie `TRUE` ou `FALSE` et s'utilise comme suit :

```
if (file_exists("testfile.txt")) echo "Le fichier existe";
```

Créer un fichier

À ce stade, le fichier *testfile.txt* n'existe pas encore; donc nous allons le créer et y écrire quelques lignes. Entrez le programme de l'exemple 7-4 et enregistrez-le sous le nom *testfile.php*.

Exemple 7-4. Création d'un fichier texte simple

```
<?php // testfile.php
  $fh = fopen("testfile.txt", 'w') or die("Impossible de créer le fichier");

  $text = <<<_END
Ligne 1
Ligne 2
Ligne 3
_END;

  fwrite($fh, $text) or die("Impossible d'écrire dans le fichier");
  fclose($fh);
  echo "Écriture dans le fichier 'testfile.txt' réussie";
?>
```

Lorsque vous exécutez ceci dans un navigateur, si tout se passe bien, vous voyez apparaitre le message `Écriture dans le fichier 'testfile.txt' réussie`. Si vous obtenez un message d'erreur, alors votre disque dur est peut-être saturé ou, plus vraisemblablement, vous n'avez pas la permission de créer le fichier ou de le modifier, auquel cas vous devrez modifier les attributs du dossier de destination, selon votre système d'exploitation. Si tout s'est bien passé, le fichier *testfile.php* apparait dans le même dossier que celui où vous avez enregistré le programme *testfile.php*. Essayez d'ouvrir le fichier texte dans un éditeur de texte ou de programme. Le contenu devrait être le suivant :

```
Ligne 1
Ligne 2
Ligne 3
```

Cet exemple simple montre la séquence de toute gestion de fichier :

1. Commencez toujours par ouvrir le fichier. La fonction `fopen` s'en charge.
2. Ensuite, appelez les autres fonctions; ici, nous écrivons dans le fichier (`fwrite`), mais vous pouvez aussi lire le contenu d'un fichier existant (`fread` ou `fgets`) et réaliser d'autres choses.
3. Terminez par la fermeture du fichier (`fclose`). Même si le programme le fait automatiquement lorsqu'il s'achève, vous devez par principe nettoyer l'environnement par la clôture de tout fichier à la fin des écritures ou des lectures.

Chaque fichier ouvert nécessite une ressource de fichier pour que PHP puisse ensuite y accéder et le gérer. L'exemple précédent définit la variable `$fh` (que j'ai choisie pour représenter un *file handle*, un *descripteur de fichier*) avec la valeur renvoyée par la fonction `fopen`. Ensuite, pour accéder au fichier ouvert, chaque fonction de gestion de fichier, telle que `fwrite` ou `fclose`, doit recevoir ce descripteur de fichier `$fh` en tant que paramètre pour identifier le fichier auquel elle doit accéder. Ne vous préoccupez pas de ce que contient la variable `$fh`. C'est un entier utilisé par PHP pour faire référence à des informations internes à propos du fichier. Contentez-vous de passer la variable aux autres fonctions, mais n'oubliez pas de le faire.

En cas d'échec, `fopen` renvoie `FALSE`. L'exemple précédent montre un moyen simple pour intercepter l'échec et y répondre : il appelle la fonction `die` (mourir) pour arrêter le programme et afficher un message d'erreur à l'utilisateur. Une application web ne doit jamais s'arrêter d'une manière aussi brutale (vous devez plutôt créer une page web avec un message d'erreur), mais ceci suffit dans le cadre des tests.

Remarquez la présence du second argument de l'appel à la fonction `fopen`. Il contient le seul caractère `w`, qui indique à la fonction d'ouvrir le fichier en écriture (`w` pour *write*). La fonction crée le fichier s'il n'existe pas déjà. Soyez prudent lorsque vous utilisez ces fonctions car, si le fichier existe déjà, le paramètre de mode `w` dans `fopen` provoque la suppression du contenu du fichier s'il existe déjà (même si vous n'avez encore rien écrit de nouveau).

Plusieurs paramètres de mode existent, qui indiquent à `fopen` d'agir de manières différentes (tableau 7-5).

Tableau 7-5. Les modes d'ouverture pris en charge par fopen

Mode	Action	Description
`'r'`	Lire depuis le début du fichier	Ouverture du fichier en lecture seule; place le pointeur de fichier au début du fichier. Renvoie FALSE si le fichier n'existe pas déjà.
`'r+'`	Lire depuis le début du fichier et permettre l'écriture	Ouverture du fichier en lecture et écriture; place le pointeur du fichier au début. Renvoie FALSE si le fichier n'existe pas déjà.
`'w'`	Écrire depuis le début du fichier et tronquer le fichier	Ouverture en écriture seule; place le pointeur de fichier au début du fichier et tronque le fichier à une longueur zéro. Si le fichier n'existe pas, tente de le créer.
`'w+'`	Écrire depuis le début du fichier, tronquer le fichier et permettre la lecture	Ouverture en lecture et écriture; place le pointeur de fichier au début du fichier et tronque le fichier à une longueur zéro. Si le fichier n'existe pas, tente de le créer.
`'a'`	Ajouter à la fin du fichier	Ouverture en écriture seule ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, tente de le créer.
`'a+'`	Ajouter à la fin du fichier et permettre la lecture	Ouverture en lecture et écriture ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, tente de le créer.

Lire dans des fichiers

Le moyen le plus simple pour lire le contenu d'un fichier texte consiste à en prendre une ligne complète à l'aide de `fgets` (`f` pour fichier, `get` pour obtenir, `s` pour *string*, chaine), comme dans l'exemple 7-5.

Exemple 7-5. Lire un fichier avec fgets

```
<?php
  $fh = fopen("testfile.txt", 'r') or
    die("Le fichier n'existe pas ou vous n'avez pas la permission de l'ouvrir");

  $ligne = fgets($fh);
  fclose($fh);
  echo $ligne;
?>
```

Si vous avez créé le fichier comme indiqué à l'exemple 7-4, vous obtenez alors la première ligne :

```
Ligne 1
```

Pour récupérer plusieurs lignes ou des portions de ligne, utilisez la fonction `fread`, comme dans l'exemple 7-6.

Exemple 7-6. Lire un fichier avec fread

```
<?php
  $fh = fopen("testfile.txt", 'r') or
    die("Le fichier n'existe pas ou vous n'avez pas la permission de l'ouvrir");

  $text = fread($fh, 3);
  fclose($fh);
  echo $text;
?>
```

L'exemple demande trois caractères dans l'appel à `fread`, donc le programme affiche ce qui suit :

```
Lig
```

La fonction `fread` sert habituellement à lire des données binaires. Si vous l'utilisez sur des données de type texte qui s'étalent sur plusieurs lignes, n'oubliez pas de compter les caractères de nouvelle ligne.

Copier des fichiers

Examinons à présent la fonction PHP `copy` pour créer un clone de *testfile.txt*. Entrez l'exemple 7-7 et enregistrez-le sous le nom *copyfile.php*, puis appelez le programme dans le navigateur.

Exemple 7-7. Copie d'un fichier

```
<?php // copyfile.php
  copy('testfile.txt', 'testfile2.txt') or die("Impossible de copier le fichier");
  echo "Fichier copié avec succès vers 'testfile2.txt'";
?>
```

Si vous examinez le dossier, vous constatez cette fois qu'un nouveau fichier nommé *testfile2.txt* y figure. Au fait, si vous ne voulez pas que le programme sorte brutalement, lors d'un échec de tentative de copie de fichier, essayez la syntaxe alternative illustrée dans l'exemple 7-8.

Exemple 7-8. Syntaxe alternative pour copier un fichier

```
<?php // copyfile2.php
  if (!copy('testfile.txt', 'testfile2.txt'))
    echo "Impossible de copier le fichier";
  else echo "Fichier copié avec succès vers 'testfile2.txt'";
?>
```

Déplacer un fichier

Déplacer un fichier revient en fait à le renommer, avec la fonction `rename`, comme dans l'exemple 7-9.

Exemple 7-9. Déplacement d'un fichier

```
<?php // movefile.php
  if (!rename('testfile2.txt', 'testfile2.new'))
    echo "Impossible de renommer le fichier";
  else echo "Fichier renommé avec succès vers 'testfile2.new'";
?>
```

Vous pouvez également utiliser la fonction `rename` sur des dossiers. Pour éviter tout message d'avertissement, si le fichier original n'existe pas, appelez d'abord la fonction `file_exists` pour vérifier.

Supprimer un fichier

La suppression d'un fichier revient à utiliser la fonction `unlink` pour le supprimer du système de fichiers, comme illustré dans l'exemple 7-10.

Exemple 7-10. Suppression d'un fichier

```
<?php // deletefile.php
  if (!unlink('testfile2.new')) echo "Impossible de supprimer le fichier";
  else echo "Fichier 'testfile2.new' supprimé avec succès";
?>
```

Chaque fois que vous accédez directement à des fichiers sur votre disque dur, vous devez vérifier que vous ne corrompez pas la structure de ces fichiers. Ainsi, si vous êtes sur le point de supprimer un fichier en fonction d'informations d'emplacement de ce fichier fournies par l'utilisateur, vous devez absolument vous assurer que ce fichier peut être supprimé en toute sécurité et que l'utilisateur est autorisé à le supprimer.

De même que lors d'une tentative de déplacement d'un fichier qui n'existe pas, un message apparait si vous essayez de supprimer un fichier inexistant. Pour éviter ce message, vérifiez son existence à l'aide de `file_exists` avant d'appeler `unlink`.

Mettre un fichier à jour

Il est souvent nécessaire d'ajouter quelques données supplémentaires à un fichier déjà enregistré et plusieurs méthodes permettent d'y parvenir. Vous pouvez utiliser une des deux méthodes d'ajout à la fin du fichier (tableau 7-5) ou vous pouvez simplement ouvrir le fichier en lecture et écriture avec l'un des autres modes qui autorise l'écriture, pour déplacer ensuite le pointeur de fichier à l'emplacement adéquat dans le fichier où vous voulez écrire ou lire.

Le *pointeur de fichier* est l'emplacement dans un fichier où le prochain accès aura lieu, que ce soit en lecture ou écriture. Il diffère du *descripteur de fichier* (celui stocké dans la variable `$fh` de l'exemple 7-4), qui contient des informations sur le fichier en cours d'accès.

Voyez-le en action en entrant le code de l'exemple 7-11 et en l'enregistrant sous le nom de fichier *update.php*. Puis appelez-le dans votre navigateur.

Exemple 7-11. Modification d'un fichier

```
<?php // update.php
  $fh   = fopen("testfile.txt", 'r+') or die("Impossible d'ouvrir le fichier");
  $text = fgets($fh);

  fseek($fh, 0, SEEK_END);
  fwrite($fh, "$text") or die("Impossible d'écrire dans le fichier");
  fclose($fh);
    echo "Fichier 'testfile.txt' modifié avec succès";
  ?>
```

Ce script ouvre *testfile.txt* à la fois en lecture et écriture à l'aide du mode `'r+'`, qui place le pointeur juste au tout début. Il fait ensuite appel à la fonction `fgets` pour lire une seule ligne du fichier (jusqu'au premier caractère de nouvelle ligne). Ensuite, la fonction `fseek` (*seek* pour chercher) permet de déplacer le pointeur de fichier tout à la fin du fichier, où la ligne extraite du début (mémorisée dans `$text`) est ajoutée à la suite du fichier. Le script ferme le fichier avant d'afficher un message de réussite. Le fichier résultant contient à présent :

```
Ligne 1
Ligne 2
Ligne 3
Ligne 1
```

La première ligne a bien été copiée à la fin du fichier.

Telle qu'utilisée ici, en plus du descripteur de fichier `$fh`, la fonction `fseek` reçoit deux autres paramètres, `0` et `SEEK_END`. `SEEK_END` indique à la fonction de déplacer le pointeur de fichier tout à la fin, et `0` stipule le nombre d'emplacements dont le pointeur doit reculer à partir de ce point. Dans le cas de l'exemple 7-11, la valeur `0` impose que le pointeur demeure tout à la fin du fichier.

Deux autres options existent pour la fonction `fseek` : `SEEK_SET` et `SEEK_CUR`. L'option `SEEK_SET` indique à la fonction de placer le pointeur de fichier à l'emplacement exact donné par le paramètre précédent. Ainsi, la ligne suivante déplace le pointeur à l'emplacement 18 :

```
fseek($fh, 18, SEEK_SET);
```

Tandis que `SEEK_CUR` permet d'imposer au pointeur de fichier d'aller à l'emplacement actuel *plus* la valeur du décalage indiqué. Par conséquent, si le pointeur de fichier est actuellement à l'emplacement 18, l'appel suivant le déplace à l'emplacement 23 :

```
fseek($fh, 5, SEEK_CUR);
```

Bien que ce ne soit recommandé que si vous avez de très bonnes raisons de le faire, il est même possible d'utiliser des fichiers texte de ce genre (mais dont les lignes ont alors de préférence une longueur fixe) en guise de base de données de *fichiers à plat*. Le programme exploite alors `fseek` pour avancer et reculer dans un tel fichier pour retrouver ou ajouter des enregistrements. Pour supprimer un enregistrement, vous pouvez par exemple l'écraser avec des caractères `0`, et ainsi de suite.

Verrouiller un fichier contre les accès multiples

Les programmes web sont souvent appelés par de nombreux utilisateurs à la fois. Si plusieurs personnes tentent d'écrire simultanément dans un fichier, celui-ci subit une corruption qui peut en interdire l'utilisation ultérieure. Et si une personne écrit dans le fichier pendant qu'une autre est en train d'en lire des données, le fichier est sauf mais la personne en cours de lecture risque d'obtenir des résultats imprévisibles. Au niveau des fichiers, pour gérer des utilisateurs simultanés, il est indispensable de faire appel à la fonction `flock` (*lock* pour verrouiller) de verrouillage de fichier. Cette fonction place

en file d'attente toutes les autres requêtes d'accès au fichier tant que votre programme n'a pas libéré le verrou. Donc, chaque fois que vos programmes doivent accéder en écriture à des fichiers potentiellement accessibles en même temps par des utilisateurs concurrents, vous devez ajouter ce qu'il faut pour verrouiller le fichier vis-à-vis d'eux. L'exemple 7-12 montre comment faire, à partir d'une version corrigée de l'exemple 7-11.

Exemple 7-12. Modification d'un fichier avec verrouillage du fichier

```
<?php
  $fh = fopen("testfile.txt", 'r+') or die("Impossible d'ouvrir le fichier");
  $text = fgets($fh);

  if (flock($fh, LOCK_EX))
  {
    fseek($fh, 0, SEEK_END);
    fwrite($fh, "$text") or die("Impossible d'écrire dans le fichier");
    flock($fh, LOCK_UN);
  }

  fclose($fh);
  echo "Fichier 'testfile.txt' modifié avec succès";
?>
```

Une astuce de verrouillage de fichier permet de maintenir le meilleur temps de réponse pour vos visiteurs web : verrouillez le fichier juste avant d'y apporter une modification, appliquez la modification et libérez immédiatement le verrou dans la foulée. Le fait de verrouiller un fichier plus longtemps ralentit inutilement l'application. C'est la raison pour laquelle les appels à `flock` dans l'exemple 7-12 se font immédiatement avant et immédiatement après l'appel à `fwrite`.

Le premier appel à `flock` impose un verrou exclusif sur le fichier référencé par `$fh` à l'aide du paramètre `LOCK_EX` :

```
flock($fh, LOCK_EX);
```

À ce stade, aucun autre processus ne peut écrire ni lire dans le fichier, tant que le verrou n'est pas libéré avec le paramètre `LOCK_UN`, comme suit :

```
flock($fh, LOCK_UN);
```

Dès que le verrou est libéré, les autres processus en cours reçoivent à nouveau la possibilité d'accéder au fichier. C'est la raison qui explique aussi que vous devez autant que possible refaire le `fseek` au point où vous voulez accéder dans un fichier, chaque fois que vous voulez lire ou écrire des données, parce qu'un autre processus risque d'avoir modifié le fichier depuis votre dernier accès.

Ceci dit, avez-vous remarqué que l'appel au verrouillage exclusif fait partie d'une instruction `if` ? Ceci est dû au fait que `flock` n'est pas pris en charge par tous les systèmes, donc il est préférable de vérifier que vous avez réussi à sécuriser le verrou, juste au cas où vous ne l'auriez pas obtenu.

Autre chose dont vous devez être conscient, flock est connu pour être *consultatif*, c'est-à-dire qu'il ne verrouille un fichier que vis-à-vis d'autres processus qui font aussi appel à cette fonction. Si vous avez du code qui entre dans le fichier sans utiliser le verrouillage du fichier par flock, il peut toujours contourner le verrouillage et faire subir des dégâts à votre fichier.

Enfin, la mise en œuvre du verrouillage de fichier, puis le fait de fermer le fichier dans une autre portion de code, sans le déverrouiller, peut mener à un bogue extrêmement difficile à retrouver.

La fonction flock ne fonctionne pas sur NFS ni sur de nombreux systèmes de fichiers en réseau. De plus, lors de l'utilisation d'un serveur multithread tel qu'ISAPI, vous ne pouvez vous fier à flock pour protéger des fichiers contre d'autres scripts PHP qui fonctionnent dans des threads parallèles sur la même instance de serveur. En outre, flock n'est pris en charge sur aucun système utilisant l'ancien système de fichiers FAT, comme les anciennes versions de Windows.

Lire la totalité d'un fichier

Une fonction pratique permet de lire la totalité d'un fichier sans devoir gérer de descripteur de fichier : file_get_contents. Son usage est très aisé, comme l'illustre l'exemple 7-13.

Exemple 7-13. Utilisation de file_get_contents

```
<?php
  echo "<pre>"; // Permet l'affichage des retours à la ligne
  echo file_get_contents("testfile.txt");
  echo "</pre>"; // Ferme la balise pre
?>
```

En fait, cette fonction est bien plus puissante que cela, puisque vous pouvez aussi l'utiliser pour aller chercher un fichier sur un serveur à travers l'internet, comme dans l'exemple 7-14, qui demande le HTML de la page d'accueil des Éditions Reynald Goulet, puis l'affiche comme si l'utilisateur avait surfé lui-même sur la page. Le résultat est comparable à la figure 7-1.

Exemple 7-14. Récupération de la page d'accueil des Éditions Reynald Goulet

```
<?php
  echo file_get_contents("http://goulet.ca");
?>
```


Figure 7-1. La page d'accueil des Éditions Reynald Goulet capturée par file_get_contents

Téléverser des fichiers

Le téléversement (*upload*) ou téléchargement de fichiers *vers* un serveur web est un sujet quelquefois déconcertant pour nombre de gens, mais il ne peut pas être plus facile. Tout ce que vous devez faire pour déposer un fichier à distance à partir d'un formulaire se résume à choisir un type d'encodage spécial appelé multipart/form-data, et le navigateur se charge du reste. Pour voir cela à l'œuvre, entrez le programme de l'exemple 7-15 et enregistrez-le sous le nom *upload.php*. Lorsque vous l'exécutez, un formulaire s'affiche dans le navigateur qui permet de déposer sur le serveur le fichier de votre choix.

Exemple 7-15. Téléversement d'une image avec upload.php

```
<?php // upload.php
  echo <<<_END
    <html><head><title>Formulaire de téléversement en PHP</title></head><body>
    <form method='post' action='upload.php' enctype='multipart/form-data'>
    Sélectionnez le fichier : <input type='file' name='nomfichier' size='10'>
    <input type='submit' value='Déposer'>
    </form>
_END;

  if ($_FILES)
  {
    $nom = $_FILES['nomfichier']['name'];
    move_uploaded_file($_FILES['nomfichier']['tmp_name'], $nom);
```

```
    echo "Image téléchargée : '$nom'<br><img src='$nom'>";
  }
  echo "</body></html>";
?>
```

Examinons ce programme section par section. La première ligne de l'instruction `echo` sur plusieurs lignes commence un document HTML, affiche le titre et amorce le corps (*body*) du document.

Ensuite apparait le formulaire qui choisit la méthode Post de soumission du formulaire, définit la cible des données publiées comme le programme *upload.php* (le programme lui-même), et indique au navigateur que les données publiées doivent être encodées avec le type MIME de contenu multipart/form-data.

Le formulaire préparé et défini, les lignes suivantes affichent l'invite `Sélectionnez le fichier :`, puis demandent deux entrées. La première demande un fichier ; elle utilise le type d'entrée `file`, un nom de fichier `nomfichier`, et un champ d'entrée de 10 caractères. La seconde entrée se résume à un bouton de soumission qui porte l'étiquette `Déposer` (qui se substitue au texte par défaut `Envoyer`). Et le formulaire est fermé.

Ce court programme illustre une technique fréquemment utilisée en programmation web, où un seul et même programme est appelé deux fois : la première, lorsque l'utilisateur visite la page pour la première fois, et ensuite, quand l'utilisateur clique sur le bouton de soumission.

Le code PHP nécessaire pour déposer les données à distance est assez simple car tous les fichiers téléchargés sont placés dans le tableau système associatif `$_FILES`. Par conséquent, une rapide vérification, pour voir si `$_FILES` contient quoi que ce soit, suffit pour déterminer si l'utilisateur a envoyé un fichier. Ceci est réalisé à l'aide de l'instruction `if ($_FILES)`.

La première fois que l'utilisateur visite la page et avant d'avoir déposé un fichier, `$_FILES` est vide, donc ce bloc de code est ignoré. Lorsque l'utilisateur verse un fichier, le programme s'exécute à nouveau et découvre un élément dans le tableau `$_FILES`.

Dès que le programme constate la présence d'un fichier déposé, son nom réel, lu à partir de l'ordinateur qui a envoyé le fichier, est récupéré et placé dans la variable `$nom`. À ce stade, nous disposons de tout le nécessaire pour déplacer le fichier de l'emplacement temporaire où PHP a déposé le fichier, vers un dossier plus permanent. Pour ce faire, nous utilisons la fonction `move_uploaded_file`, à laquelle nous passons le nom du fichier original, pour l'enregistrer dans le dossier courant.

Enfin, la balise `IMG` permet d'afficher le fichier et nous obtenons le résultat de la figure 7-2.

Lorsque vous exécutez ce programme, si vous recevez des messages d'avertissement tels que `Permission denied` pour l'appel à la fonction `move_uploaded_file`, alors vous ne disposez pas des permissions nécessaires dans le dossier où le programme s'exécute.

Figure 7-2. Téléversement d'une image par données de formulaire

Utiliser $_FILES

Cinq informations sont mémorisées dans le tableau `$_FILES` lors du chargement d'un fichier, illustrées dans le tableau 7-6 (où fichier est le nom du champ de choix du fichier fourni dans le formulaire de soumission).

Tableau 7-6. Le contenu du tableau $_FILES

Élément du tableau	Contenu
`$_FILES['fichier']['name']`	Le nom du fichier téléversé (par ex. smiley.jpg)
`$_FILES['fichier']['type']`	Le type de contenu du fichier (par ex. image/jpeg)
`$_FILES['fichier']['size']`	La taille en octets du fichier
`$_FILES['fichier']['tmp_name']`	Le nom du fichier temporaire stocké sur le serveur
`$_FILES['fichier']['error']`	Le code d'erreur résultant de l'envoi du fichier

Les types de contenus étaient connus autrefois sous l'appellation *types MIME* (*Multipurpose Internet Mail Extension*), mais comme leur usage s'est ensuite étendu à tout l'internet, ils s'appellent désormais *Internet media types* (types de media de l'internet). Le tableau 7-7 énumère quelques-uns de ces types parmi les plus utilisés, qui peuvent apparaitre dans `$_FILES['fichier']['type']`.

Tableau 7-7. Quelques Internet media types usuels

`application/pdf`	`image/gif`	`multipart/form-data`	`text/xml`
`application/zip`	`image/jpeg`	`text/css`	`video/mpeg`
`audio/mpeg`	`image/png`	`text/html`	`video/mp4`
`audio/x-wav`	`image/tiff`	`text/plain`	`video/quicktime`

Validation

Je l'ai déjà évoqué, je le répète et je le répéterai certainement encore, la validation des données des formulaires est de la plus haute importance, à cause du risque que des utilisateurs tentent de pirater votre serveur par leur entremise.

En plus de données formées de manière malintentionnée, vous devez aussi vérifier des détails comme le fait qu'un fichier a bien été reçu et, si c'est le cas, que celui-ci est bien du type attendu.

L'exemple 7-16 prend ces éléments en compte pour récrire *upload.php* d'une manière plus sécuritaire et produire une nouvelle version, *upload2.php*.

Exemple 7-16. Une version plus sécuritaire d'upload.php

```
<?php //upload2.php
  echo <<<_END
    <html><head><title>Formulaire de téléversement en PHP</title></head><body>
    <form method='post' action='upload2.php' enctype='multipart/form-data'>
    Sélectionnez un fichier JPG, GIF, PNG ou TIF :
    <input type='file' name='nomfichier' size='10'>
    <input type='submit' value='Déposer'></form>
_END;

  if ($_FILES)
  {
    $nom = $_FILES['nomfichier']['name'];

    switch($_FILES['nomfichier']['type'])
    {
      case 'image/jpeg': $ext = 'jpg'; break;
      case 'image/gif':  $ext = 'gif'; break;
      case 'image/png':  $ext = 'png'; break;
      case 'image/tiff': $ext = 'tif'; break;
      default:           $ext = '';    break;
    }
    if ($ext)
    {
      $n = "image.$ext";
      move_uploaded_file($_FILES['nomfichier']['tmp_name'], $n);
      echo "Image téléchargée : '$nom' de type '$n':<br>";
      echo "<img src='$n'>";
    }
    else echo "'$nom' n'est pas accepté comme fichier image";
  }
  else echo "Aucune image chargée";

  echo "</body></html>";
?>
```


La section de code hors-HTML a pris de l'ampleur, de quelques lignes dans l'exemple 7-15 à plus de vingt lignes, et débute à `if` (`$_FILES`).

Comme dans la version précédente, cette instruction `if` vérifie si des données ont été publiées, mais un `else` correspondant fait à présent son apparition dans le bas du programme pour afficher à l'écran qu'aucun fichier n'a été déposé.

Au sein de l'instruction `if`, la variable `$nom` reçoit la valeur du nom de fichier, tel qu'il a été reçu de la part de l'ordinateur du client distant, comme précédemment, mais cette fois, nous ne nous fions pas à la validité des données envoyées par l'utilisateur. Une instruction `switch` permet de comparer le type du contenu chargé à quatre types d'image que le programme prend en charge. S'il y a correspondance, la variable `$ext` reçoit l'extension en trois lettres de ce type de fichier. Si aucune correspondance n'est établie, alors le fichier déposé n'est d'aucun type accepté et la variable `$ext` est réglée à une chaine vide, `""`.

La partie suivante du code vérifie ensuite si la variable `$ext` contient une chaine et, si c'est le cas, crée un nouveau nom de fichier appelé `$n` avec le nom de base image et l'extension mémorisée dans `$ext`. Cela signifie que le programme possède le plein contrôle du nom du fichier à créer, qui ne peut être qu'un des suivants : *image.jpg*, *image.gif*, *image.png* ou *image.tif*.

Désormais certain que le programme n'a pas été corrompu, le reste du code PHP reprend, à peu de chose près, la suite du code de la version précédente. Il déplace l'image temporairement stockée vers son nouvel emplacement, puis l'affiche, ainsi que les ancien et nouveau noms de l'image.

Vous ne devez pas vous inquiéter de supprimer le fichier temporaire que PHP crée durant le processus de chargement, car si le fichier n'a pas été déplacé ni renommé, il est automatiquement supprimé à la sortie du programme.

À la fin de l'instruction `if`, un `else` correspondant n'est exécuté que lorsqu'un fichier non pris en charge a été chargé, pour afficher un message approprié.

Lorsque vous rédigez vos propres routines de chargement de fichier, je vous conseille instamment d'emprunter une telle approche et de prédéfinir des noms et emplacements pour les fichiers chargés. De cette manière, aucune tentative d'ajout de chemin de fichier ni d'autres données malintentionnées dans les variables que vous utilisez ne peut « passer ». Si le risque subsiste que plusieurs utilisateurs déposent des fichiers de même nom, la parade consiste, par exemple, à préfixer le nom de fichier avec celui de l'utilisateur, ou de les déposer individuellement dans des dossiers spécifiques à chaque utilisateur.

Si vous devez utiliser à tout prix le nom d'un fichier fourni, assainissez-le en n'autorisant que des caractères alphanumériques et le point, ce que vous pouvez réaliser à l'aide de la commande suivante, qui utilise une expression régulière (voir chapitre 17) pour effectuer une recherche et un emplacement sur le contenu de la variable `$nom` :

```
$nom = preg_replace("/[^A-Za-z0-9.]/", "", $nom);
```

Ceci ne laisse plus que les caractères A à Z, a à z, 0 à 9 et les points dans la chaine `$nom`, et élimine tout le reste.

Encore mieux, pour garantir que votre programme fonctionne sur tous les systèmes, qu'ils soient sensibles ou non à la casse, vous avez intérêt à préférer la commande suivante, qui convertit en même temps tous les caractères de capitales en bas de casse :

```
$nom = strtolower(preg_replace("/[^A-Za-z0-9.]/", "", $nom));
```

Il se peut que vous rencontriez parfois le type de média `image/pjpeg`, qui correspond à du JPEG progressif. Vous pouvez l'ajouter sans crainte au code en tant qu'alias d'`image/jpeg`, comme suit :

```
case 'image/pjpeg':
case 'image/jpeg': $ext = 'jpg'; break;
```

Appels système

Quelquefois, PHP ne dispose pas de la fonction dont vous avez besoin pour effectuer une tâche précise, tandis que le système d'exploitation la fournit. Dans de tels cas, lancez l'appel système `exec` pour réaliser cette tâche.

Ainsi, pour obtenir un aperçu rapide du contenu du dossier courant, il est possible d'utiliser un programme comme celui de l'exemple 7-17. Si vous utilisez un système Windows, la commande système à appeler s'appelle `dir`, tandis que sous Linux, Unix ou Mac, placez en commentaire la première ligne et ôtez le commentaire devant la deuxième ligne pour utiliser la commande système `ls`. Pour tester l'exemple, entrez-le et enregistrez le fichier sous le nom *exec.php*, puis appelez celui-ci dans le navigateur.

Exemple 7-17. Exécution d'une commande système

```
<?php // exec.php
  $cmd = "dir";   // Windows
  // $cmd = "ls"; // Linux, Unix & Mac

  exec(escapeshellcmd($cmd), $output, $status);

  if ($status) echo "Échec de la commande exec";
  else
  {
    echo "<pre>";
    foreach($output as $line) echo htmlspecialchars("$line\n");
    echo "</pre>";
  }
?>
```


L'appel à la fonction `htmlspecialchars` permet ici de convertir tout caractère spécial renvoyé par le système en un caractère que HTML peut interpréter et afficher correctement, en clarifiant la sortie. Selon le système que vous utilisez, le résultat de l'exécution de ce programme prend un aspect comparable au suivant (avec la commande `dir` de Windows) :

```
Le volume dans le lecteur C n'a pas de nom.


26/02/2015  11:21    <REP>          .
26/02/2015  11:21    <REP>          ..
06/02/2015  15:46    <REP>          dashboard
06/02/2015  15:46    <REP>          forbidden
06/02/2015  15:46    <REP>          img
30/03/2013  12:29               202 index.html
30/03/2013  12:29               267 index.php
06/02/2015  15:46    <REP>          restricted
09/02/2015  11:49               122 test.htm
07/02/2015  08:24    <REP>          xampp
```

La commande `exec` nécessite trois paramètres :

- la commande elle-même (rangée dans `$cmd` dans l'exemple précédent) ;
- un tableau dans lequel le système dépose la sortie de la commande (soit `$output` dans l'exemple);
- une variable pour contenir l'état (`$status` dans l'exemple) de l'appel.

Les deux derniers paramètres sont facultatifs mais, si vous les omettez, vous ne connaitrez ni le résultat de sortie créé par la commande système, ni l'état de réussite ou d'échec de l'exécution de la commande.

Remarquez aussi l'utilisation de la fonction `escapeshellcmd`. Prenez l'habitude de toujours l'utiliser lorsque vous exécutez un appel à `exec`, parce qu'elle assainit la chaine de commande pour éviter l'exécution de commandes arbitraires et dangereuses si cette commande est l'effet d'une entrée par l'utilisateur.

Les fonctions d'appel au système sont généralement désactivées sur les hôtes mutualisés (partagés) sur le web, parce qu'elles posent de réels problèmes de sécurité. Autrement dit, essayez toujours de régler vos problèmes à base de code PHP-même et, seulement si vous ne pouvez faire autrement, envisagez alors de faire appel au système. Notez que les commandes système sont relativement lentes et, surtout, que vous devez mettre en œuvre deux implémentations, si votre application est susceptible de s'exécuter tant sous Windows que sous Linux/Unix/Mac.

XHTML ou HTML5?

Du fait que les documents XHTML doivent être parfaitement et strictement formulés, il est possible de les analyser à l'aide d'analyseurs XML standards, au contraire des documents HTML, qui exigent un analyseur HTML spécifique, plus indulgent.

C'est pour cette raison que XHTML n'a jamais réussi à percer et, lorsqu'il s'est agi de définir une nouvelle norme, le W3C a préféré soutenir HTML5 plutôt que la norme XHTML2, plus récente.

HTML5 renferme certaines des fonctionnalités à la fois de HTML4 et de XHTML, mais s'avère d'utilisation bien plus facile, de validation moins stricte et, heureusement, il suffit désormais d'un seul type de document à placer en tête d'une page en HTML5 (au lieu de toute une variété de types de documents transitoires, stricts ou de *frameset*, nécessaires auparavant), notamment :

```
<!DOCTYPE html>
```

Même le simple `<html>` suffit pour indiquer au navigateur que la page web est conçue pour HTML5 et, comme toutes les dernières versions des navigateurs les plus utilisés prennent en charge l'essentiel de la norme HTML5 depuis environ 2011, ce type de document est le seul dont vous avez besoin, à moins, bien entendu, que vous ne décidiez de vous intéresser à d'anciens navigateurs.

Quels que soient leurs intentions et leurs buts, lorsqu'ils rédigent des documents en HTML, les développeurs web peuvent désormais éluder en toute sécurité les types et syntaxes des anciens documents XHTML (comme l'utilisation de la balise `<br />` au lieu de la simplissime `<br>`). Cependant, si vous vous retrouvez confronté à la nécessité de prendre en charge un très ancien navigateur ou à développer une application inhabituelle qui se fonde sur XHTML, vous trouverez de plus amples informations sur comment faire à *http://xhtml.com*.

Questions

1. Quel spécificateur de conversion devez-vous utiliser dans `printf` pour afficher un nombre en (point) virgule flottante ?
2. Quelle instruction `printf` utiliseriez-vous pour prendre la chaine d'entrée `"Joyeux anniversaire"` et sortir la chaine `"**Joyeux"` ?
3. Pour envoyer le résultat de sortie de `printf` à une variable au lieu du navigateur, quelle variante de fonction utilisez-vous ?
4. Comment créez-vous un horodatage (*timestamp*) Unix correspondant au 2 mai 2016 à 7 h 11 du matin ?
5. Quel mode d'accès à un fichier utilisez-vous dans `fopen` pour ouvrir un fichier en écriture et en lecture, avec le fichier tronqué à son début ?
6. Quelle est la commande PHP qui permet de supprimer un fichier *fichier.txt* existant ?
7. Quelle fonction PHP permet de lire la totalité d'un fichier en une seule fois, même au-delà du web ?
8. Quelle variable superglobale de PHP contient les détails d'un fichier téléversé ?
9. Quelle fonction de PHP permet d'exécuter des commandes système ?
10. Quel est, parmi les suivants, le style de balise préférentiel en HTML5 : `<hr>` ou `<hr />` ?

Retrouvez les réponses du chapitre 7 dans l'annexe A.

CHAPITRE 8

Introduction à MySQL

Avec plus de dix millions d'installations dans le monde, MySQL est probablement le système de gestion de bases de données le plus populaire au niveau des serveurs Web. Développé au milieu des années 1990, sa technologie a aujourd'hui atteint une maturité qui justifie qu'il anime la plupart des destinations les plus visitées de l'internet.

Une des raisons de son succès réside sans aucun doute dans le fait qu'à l'instar de PHP, son utilisation est gratuite. Ce qui ne l'empêche pas d'être extrêmement puissant et exceptionnellement rapide. De plus, il fonctionne même sur des matériels les plus basiques et il est assez peu gourmand en termes de ressources système.

MySQL n'en demeure pas moins extrêmement évolutif, ce qui signifie qu'il peut croitre avec votre site web (pour les bancs d'essais de performances, consultez *http://www.mysql.fr/why-mysql/benchmarks)*.

Les bases de MySQL

Une *base de données* est une collection structurée d'enregistrements, autrement dit de données, stockée sur un système informatique et organisée de manière à faciliter la recherche d'informations et à accélérer l'obtention de ces informations.

Le *SQL* de MySQL signifie *langage de requête structuré* (*structured query language*). Ce langage est composé de mots d'anglais courant et est largement employé dans d'autres systèmes de gestion de bases de données, comme Oracle et Microsoft SQL Server. Il a été conçu pour exprimer des requêtes simples à partir d'une base de données, à l'aide de commandes telles que la suivante :

```
SELECT titre FROM publications WHERE auteur = 'Charles Dickens';
```

Une base de données MySQL est composée d'une ou plusieurs *tables*, qui contiennent chacune des *enregistrements* (*records*) ou *lignes* (*rows*). Ces enregistrements comportent plusieurs *colonnes* ou *champs* (*fields*) qui contiennent les données proprement dites. Le tableau 8-1 montre le contenu d'un exemple de base de données de cinq publications, avec l'auteur, le titre, le type et l'année de publication.

Tableau 8-1. Exemple de base de données simple

Auteur	Titre	Type	Année
Mark Twain	Les aventures de Tom Sawyer	Roman	1876
Jane Austen	Orgueil et préjugés	Roman	1811
Charles Darwin	De l'origine des espèces	Scientifique	1856
Charles Dickens	Le Magasin d'antiquités	Roman	1841
William Shakespeare	Roméo et Juliette	Tragédie	1594

Chaque ligne de la table correspond à un enregistrement dans une table MySQL, et chaque élément dans une ligne équivaut à un champ de table MySQL.

Pour identifier cette base de données de manière univoque (unique), je l'appellerai *publications* dans les exemples qui suivent. Vous aurez compris que toutes ces publications sont considérées comme des classiques de la littérature anglaise, j'appellerai donc cette table *classiques* au sein de la base de données qui en conserve les détails.

Résumé des termes liés aux bases de données

À partir d'ici, vous devez retenir les termes suivants :

Base de données
: Le conteneur global d'une collection de données MySQL.

Table
: Un conteneur de niveau inférieur au sein d'une base de données qui contient les données en tant que telles.

Ligne
: Un seul enregistrement dans une table, qui peut comporter plusieurs champs.

Colonne
: Le nom d'un champ dans une ligne.

Remarquez qu'il ne s'agit pas ici de reproduire une terminologie stricte ni précise, telle que celle enseignée dans les ouvrages académiques, mais de fournir des termes simples, utilisables au quotidien, pour vous permettre de vous accaparer les concepts fondamentaux et de démarrer rapidement dans l'utilisation d'une base de données.

Accéder à MySQL en ligne de commande

Pour interagir avec MySQL, trois possibilités existent : la ligne de commande, une interface web du genre de phpMyAdmin et par l'entremise d'un langage de programmation tel que PHP. Nous nous consacrerons à la troisième au chapitre 10, mais examinons d'abord les deux premières.

Démarrer l'interface en ligne de commande

Les sections suivantes décrivent les instructions nécessaires sous Windows, OS X et Linux.

Utilisateurs de Windows

Si vous avez installé XAMMP comme indiqué au chapitre 2, vous pouvez accéder à l'exécutable de MySQL à partir du dossier suivant :

```
C:\xampp\mysql\bin
```

Si vous avez installé XAMPP ailleurs que dans le dossier suggéré, utilisez plutôt ce dossier.

Par défaut, l'utilisateur initial de MySQL s'appelle *root* et n'a pas de mot de passe défini. Comme il s'agit d'un serveur de développement auquel vous seul êtes censé y accéder, nous ne nous inquiétons pas d'en créer un à ce niveau.

Donc, entrons dans l'interface en ligne de commande de MySQL : cliquez sur le menu *Démarrer*, puis dans la zone de texte *Rechercher les programmes et fichiers*, entrez `CMD` et appuyez sur *Entrée*. Une fenêtre s'ouvre, appelée *Invite de commande*. À partir de là, entrez la commande suivante (avec les adaptations nécessaires, comme indiqué ci-dessus) :

```
C:\xampp\mysql\bin\mysql -u root
```

Cette commande indique à MySQL de vous connecter en tant qu'utilisateur root, sans mot de passe.

Si à ce stade, vous recevez le message d'erreur `ERROR 2003 (HY000): Can't connect to MySQL server on 'localhost'`, cela signifie que le serveur MySQL n'est pas actif à ce moment précis. Pour l'activer, ouvrez le tableau de bord (*Control panel*) de *XAMPP*, cliquez sur *Start* en face de MySQL, puis réessayez.

Si tout se passe comme prévu, à ce stade, vous avez ouvert une session dans MySQL et vous pouvez entrer des commandes. Pour être certain que tout fonctionne correctement, entrez la commande suivante; les résultats devraient ressembler à ceux de la figure 8-1.

```
SHOW databases;
```

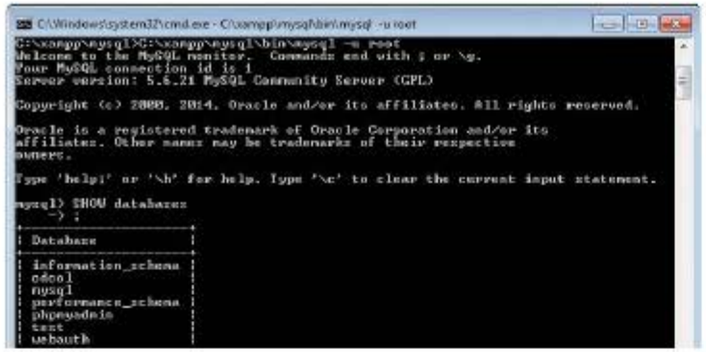


Figure 8-1. Accéder à la console de MySQL dans une Invite de commandes de Windows

Vous êtes prêt à poursuivre à la section suivante, Exploiter l'interface en ligne de commande, page 171.

Utilisateurs d'OS X

Pour suivre ce chapitre, vous devez avoir XAMPP installé sur votre ordinateur, comme indiqué au chapitre 2. Le serveur web et le serveur MySQL doivent aussi avoir démarré : cliquez sur *Start* en face d'Apache et de MySQL dans le tableau de bord (*Control panel*) de XAMPP, s'ils ne démarrent pas automatiquement avec l'ordinateur.

Pour entrer dans l'interface en ligne de commande de MySQL, démarrez le programme *Terminal* (disponible dans *Finder* → *Utilitaires*), puis appelez le programme *MySQL*, qui doit être installé dans le dossier */Applications/xampp/bin*.

Par défaut, l'utilisateur initial de MySQL s'appelle *root* et il a le mot de passe *root* également. Donc pour démarrer le programme, entrez la commande suivante :

```
/Applications/xampp/bin/mysql -u root
```

Cette commande indique à MySQL de vous connecter en tant qu'utilisateur *root* et de ne pas demander de mot de passe. Pour vérifier que tout fonctionne correctement, entrez la commande suivante ; les résultats devraient ressembler à ceux de la figure 8-2.

```
SHOW databases;
```

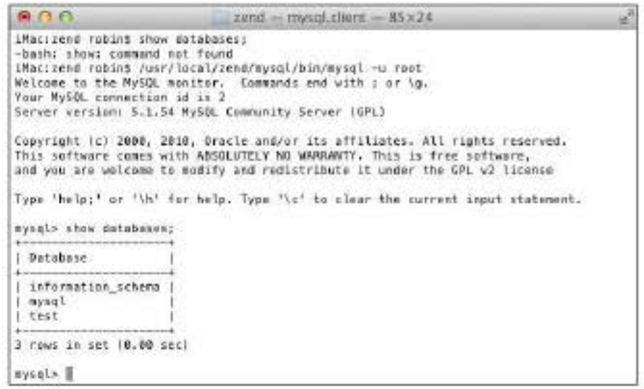



Figure 8-2. Accéder à la console de MySQL dans un Terminal OS X

Si vous recevez un message d'erreur indiquant `Can't connect to local MySQL server through socket`, cela signifie que vous devez d'abord démarrer le serveur MySQL, comme indiqué au chapitre 2.

Vous êtes prêt à poursuivre à la section suivante, Exploiter l'interface en ligne de commande, page 171.

Utilisateurs de Linux

Sur un ordinateur équipé d'un système d'exploitation semblable à Unix, comme Linux, vous avez peut-être déjà PHP et MySQL installés et en cours de fonctionnement, donc vous pouvez suivre les exemples de la section suivante. Si ce n'est pas le cas, suivez les indications de la procédure illustrée au chapitre 2 pour installer XAMPP. D'abord, entrez la commande suivante pour vous connecter à MySQL :

```
mysql -u root -p
```

Cette commande demande à MySQL d'ouvrir une session en tant qu'utilisateur *root* et de vous demander le mot de passe correspondant. Si vous avez un mot de passe, entrez-le, sinon appuyez simplement sur *Entrée*.

Dès que vous avez ouvert une session, entrez la commande suivante pour tester le programme ; la figure 8-3 illustre ce que vous devriez à peu près recevoir comme réponse :

```
SHOW databases;
```

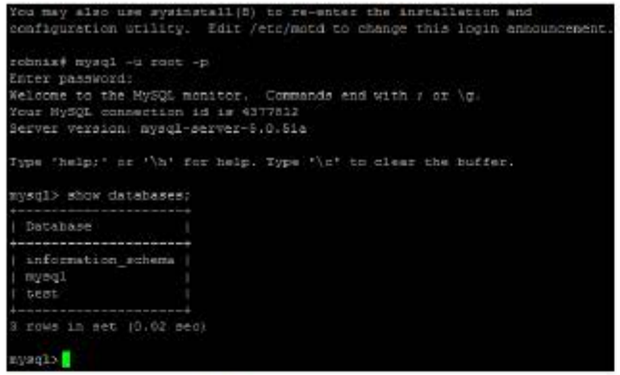


Figure 8-3. Accéder à la console de MySQL sous Linux

Si cette procédure échoue, alors consultez le chapitre 2 pour vérifier que MySQL est correctement installé et démarré. Sinon, vous êtes prêt à poursuivre à la section suivante, Exploiter l'interface en ligne de commande, page 171.

MySQL sur un serveur distant

Si vous accédez à MySQL sur un serveur distant, utilisez Telnet (ou mieux, pour des raisons de sécurité, SSH) pour ouvrir une session sur la machine distante, probablement un type de serveur Linux, FreeBSD ou Unix. Une fois la session ouverte, vous verrez que les choses se passent probablement de manière différente, selon la façon dont l'administrateur du système a installé et réglé le serveur, surtout s'il s'agit d'un serveur d'hébergement mutualisé. Par conséquent, vérifiez que vous avez accès à MySQL et que vous disposez de votre nom d'utilisateur (*username*) et votre mot de passe (*password*), qui vous ont été communiqués. Ceux-ci sous la main, essayez d'entrer la commande suivante :

```
mysql -u nomutilisateur -p
```

Entrez votre mot de passe à l'invite. Essayez ensuite la commande suivante, qui doit vous donner un résultat comparable à celui de la figure 8-3 :

```
SHOW databases;
```

Il se peut que d'autres bases de données aient été créées et que la base de données *test* ne soit pas encore présente.

Gardez à l'esprit que les administrateurs système ont le contrôle total sur tout, et que vous risquez de faire face à des installations inattendues. Ainsi, il est parfois demandé aux utilisateurs de bases de données distantes de préfixer tous les noms de leurs bases de données d'une certaine manière, avec une chaine d'identification unique pour ne pas entrer en conflit avec les bases de données créées par d'autres utilisateurs.

Par conséquent, si vous faites face au moindre problème, communiquez avec votre administrateur système (*sysadmin*), qui sera à même de vous aider à résoudre les soucis. Expliquez à votre sysadmin que vous avez besoin d'un nom d'utilisateur et d'un mot de passe. Demandez aussi la possibilité de créer une nouvelle base de données ou, éventuellement, d'en obtenir une déjà créée pour vous, prête à l'emploi. Vous pourrez alors y créer toutes les tables dont vous aurez besoin.

Exploiter l'interface en ligne de commande

À partir de maintenant, le fait que vous utilisiez Windows, OS X ou Linux pour accéder à MySQL n'a plus d'importance car toutes les commandes que vous entrez et les éventuelles erreurs que vous rencontrerez sont identiques dans la console en ligne de commande de MySQL.

Le point-virgule

Commençons par les notions fondamentales. Vous avez remarqué le point-virgule à la fin de la commande `SHOW databases;` ? Le point-virgule permet à MySQL de distinguer et de terminer les commandes. Quand vous oubliez de l'entrer, MySQL affiche une invite et attend la suite de la commande. Le point-virgule fait partie de la syntaxe pour vous permettre d'entrer des commandes sur plusieurs lignes, ce qui s'avère assez convivial parce que certaines commandes peuvent être longues. Vous pouvez aussi entrer plusieurs commandes sur une même ligne et les séparer par des points-virgules. L'interpréteur les traite toutes par lot lorsque vous appuyez sur *Entrée* et les exécute dans l'ordre.

Le fait de voir apparaitre une nouvelle invite au lieu des résultats d'une commande est, disons, très fréquent. Cela signifie que vous avez oublié de ponctuer la commande précédente par son point-virgule final. Si le cas se produit, appuyez simplement sur ; puis sur *Entrée* et vous obtiendrez ce que vous avez demandé.

MySQL peut vous présenter six types d'invites différentes (tableau 8-2), permettant de vous repérer parmi ce qu'il vous demande dans une entrée sur plusieurs lignes.

Tableau 8-2. Les six invites possibles de MySQL

Invite de MySQL	Signification
mysql>	Prêt et en attente d'une commande
->	En attente de la ligne suivante d'une commande
'>	En attente de la ligne suivante d'une chaine commencée par une apostrophe verticale
">	En attente de la ligne suivante d'une chaine commencée par un guillemet vertical
`>	En attente de la ligne suivante d'une chaine commencée par une apostrophe inclinée à droite
/*>	En attente de la ligne suivante d'une ligne de commentaire commencée par /*

Annuler une commande

Vous avez entré un début de commande puis vous décidez de ne pas l'exécuter. Quoi que vous ayez entré, *n'appuyez pas sur Ctrl+C* parce que cela ferme le programme ! Entrez plutôt `\c` et appuyez sur *Entrée*. L'exemple 8-1 montre comment l'utiliser.

Exemple 8-1. Annulation d'une ligne d'entrée

```
charabia inutile pour mysql \c
```

Lorsque vous entrez cette ligne, MySQL ignore tout ce que vous avez tapé et affiche une nouvelle invite. Sans le `\c`, il afficherait un message d'erreur. Soyez toutefois prudent car, si vous avez ouvert une chaine ou un commentaire, vous devez d'abord les terminer avec d'utiliser le `\c`, sinon MySQL pense que le `\c` fait partie de la chaine ou du commentaire. L'exemple 8-2 montre comment procéder.

Exemple 8-2. Annulation d'une ligne d'entrée dans une chaine

```
Ceci est un "charabia inutile pour mysql" \c
```

Remarquez aussi qu'un `\c` juste après un point-virgule n'annule rien, puisqu'il s'agit d'une nouvelle instruction.

Commandes de MySQL

Nous avons déjà utilisé la commande `SHOW`, qui peut énumérer les tables, les bases de données et bien d'autres éléments. Le tableau 8-3 décrit les commandes les plus utilisées.

Tableau 8-3. Commandes usuelles de MySQL

Commande	Action
ALTER	Modifier une base de données ou une table
BACKUP	Créer une sauvegarde d'une table
\c	Annuler l'entrée
CREATE	Créer une base de données
DELETE	Supprimer une ligne d'une table
DESCRIBE	Décrire les colonnes d'une table
DROP	Supprimer une base de données ou une table
EXIT (CTRL-C)	Quitter MySQL
GRANT	Modifier les privilèges d'un utilisateur
HELP (\h, \?)	Afficher l'aide
INSERT	Insérer des données
LOCK	Verrouiller une ou plusieurs tables
QUIT (\q)	Identique à EXIT
RENAME	Renommer une table
SHOW	Énumérer des détails à propos d'un objet
SOURCE	Exécuter un fichier
STATUS (\s)	Afficher l'état courant
TRUNCATE	Vider une table
UNLOCK	Déverrouiller une ou plusieurs tables
UPDATE	Modifier un enregistrement existant
USE	Utiliser une base de données

Nous allons voir en détail la plupart de ces commandes, au fur et à mesure mais, avant tout, vous devez retenir deux choses importantes à propos des commandes de MySQL :

- Les commandes et mots clés de MySQL ne sont pas sensibles à la casse. `CREATE`, `create` et `CrEaTe` signifient tous la même chose. Cependant, par souci de clarté, le style recommandé pour ceux-ci est de les écrire en capitales.
- Les noms des tables sont sensibles à la casse sous Linux et OS X, mais non sensibles à la casse sous Windows. Donc, pour des raisons de portabilité, décidez d'un type de casse et respectez-le. Le style recommandé pour les tables est de les écrire en bas de casse.

Créer une base de données

Si vous travaillez sur un serveur distant d'hébergement mutualisé, où vous ne disposez que d'un compte d'utilisateur et d'une seule base de données déjà créée pour votre usage, allez directement à la section Créer une table, page 175. Sinon, poursuivez sur votre lancée et entrez la commande suivante pour créer une base de données nommée *publications* :

```
CREATE DATABASE publications CHARSET=utf8;
```

La clause `CHARSET=utf8` est très importante pour gérer les caractères accentués dans la base de données.

Si la commande réussit, vous recevez un message qui ne signifie pas grand-chose, `Query OK, 1 row affected (0.00 sec)`, mais qui prendra bientôt tout son sens. Maintenant que la base de données est créée, la commande suivante permet de l'utiliser :

```
USE publications;
```

Le message `Database changed` apparait, pour indiquer qu'à partir de ce stade, vous êtes prêt à procéder aux manipulations suivantes sur cette base de données.

Créer des utilisateurs

Vous venez de voir combien il est facile d'utiliser MySQL. Vous avez créé une première base de données et êtes prêt à l'utiliser. Cependant, avant d'aller plus loin, il est temps de créer des utilisateurs spécifiques, car il est préférable de ne pas accorder à vos scripts d'accès `root` à MySQL. Vous risqueriez de forts maux de tête s'ils étaient piratés.

Pour créer un utilisateur, utilisez la commande `GRANT`. Elle suit la syntaxe suivante (n'entrez pas cette commande car elle ne peut fonctionner telle quelle) :

```
GRANT PRIVILEGES ON database.object TO 'nomutilisateur'@'nomhôte'
  IDENTIFIED BY 'motdepasse';
```

Ceci signifie accorder (GRANT) des permissions (PRIVILEGES) sur (ON) l'objet de base de données (database.object) déterminé à (TO) l'utilisateur nomutilisateur sur l'ordinateur de nom nomhôte, identifié par (IDENTIFIED BY) le mot de masse motdepasse. L'élément database.object fait référence à la base de données nommée database et à un ou plusieurs objets qu'elle contient, par exemple une table (tableau 8-4).

Tableau 8-4. Exemples de paramètres pour la commande GRANT

Arguments	Signification
.	Toutes les bases de données et tous leurs objets
database.*	La seule base de données nommée *database* et tous ses objets
database.object	La seule base de données nommée *database* et son objet nommé *object*

Nous avons maintenant tous les éléments pour créer un utilisateur disposant de l'accès à la seule base de données *publications* et à tous ses objets. Entrez la commande suivante (remplacez le nom d'utilisateur *jules* et le mot de passe *monmotpasse* par ceux de votre choix) :

```
GRANT ALL ON publications.* TO 'jules'@'localhost'
IDENTIFIED BY 'monmotpasse';
```

Autrement dit, nous accordons tous les privilèges d'accès (ALL) à l'utilisateur *jules@localhost*, sur la base de données *publications*, à l'aide du mot de passe *monmotpasse*. Pour vérifier que cette commande a bien fonctionné, sortez de la console de MySQL à l'aide de quit, puis rentrez dans MySQL comme précédemment, mais cette fois, au lieu d'entrer -u root -p, utilisez le nouvel utilisateur, avec -u jules -p ou celui que vous avez choisi. Le tableau 8-5 indique les commandes en fonction du système d'exploitation (SE en abrégé). Corrigez le chemin d'accès au programme client *mysql*, si vous l'avez installé dans un dossier différent du système.

Tableau 8-5. Démarrage du client MySQL et ouverture de session en tant que jules@localhost

SE	Exemple de commande
Windows	C:\xampp\mysql\bin\mysql -u jules -p
Mac OS X	/Applications/xampp/bin/mysql -u jules -p
Linux	mysql -u jules -p

Le paramètre -u indique l'utilisateur de l'ouverture de session et le -p indique à *mysql* de demander un mot de passe. À l'invite, entrez le mot de passe précisé dans la commande GRANT précédente et votre session s'ouvre. Si vous le voulez, vous pouvez indiquer le mot de passe directement, sans espace, après le paramètre -p de la commande pour éviter l'invite d'entrée du mot de passe. Ce genre de chose est cependant considéré comme une mauvaise pratique parce que si d'autres utilisateurs ont ouvert une session sur le système, ils disposent de moyens pour voir la commande que vous entrez, et donc, de connaitre votre mot de passe.

Un utilisateur ne peut jamais accorder à un autre que les privilèges dont il dispose lui-même. Il lui faut aussi disposer du privilège qui autorise l'envoi de commandes GRANT pour accorder des privilèges à d'autres utilisateurs ou créer des utilisateurs. Il existe toute une gamme de privilèges que vous pouvez accorder indépendamment les uns des autres, si vous ne voulez pas les accorder tous (ALL). Pour de plus amples informations sur le sujet, consultez le site suivant, qui détaille aussi la commande REVOKE, pour révoquer, supprimer des privilèges déjà accordés : *http://tinyurl.com/mysqlgrant*. Conservez également à l'esprit le fait que, si vous créez un utilisateur mais ne spécifiez pas la clause IDENTIFIED BY, cet utilisateur n'aura pas de mot de passe, ce qui conduit à des situations d'insécurité à éviter.

Créer une table

À ce stade, vous avez ouvert une session MySQL avec tous (ALL) les privilèges accordés sur la base de données *publications* (ou la base de données qui a été créée pour vous) et vous êtes prêt à créer une première table. Pour vérifier d'abord que vous utilisez la bonne base de données, entrez la commande suivante (remplacez *publications* par le nom de la base de données s'il diffère) :

```
USE publications;
```

Entrez ensuite la commande de l'exemple 8-3, une ligne à la fois.

Exemple 8-3. Création de la table classiques

```
CREATE TABLE classiques (
 auteur VARCHAR(128),
 titre VARCHAR(128),
 type VARCHAR(16),
 annee CHAR(4)) ENGINE MyISAM CHARSET utf8;
```

Il est également possible d'entrer cette commande sur une seule ligne, comme suit :

```
CREATE TABLE classiques (auteur VARCHAR(128), titre
VARCHAR(128), type VARCHAR(16), annee CHAR(4)) ENGINE
MyISAM CHARSET utf8;
```

Cependant, les commandes MySQL peuvent être longues et complexes, donc je vous recommande de n'entrer tout sur la même ligne que lorsque vous vous sentirez suffisamment à l'aise pour le faire.

MySQL renvoie la réponse Query OK, 0 rows affected, suivi du temps nécessaire pour exécuter la commande. Si vous obtenez plutôt un message d'erreur, vérifiez soigneusement la syntaxe. Chaque parenthèse, chaque virgule importe et il est très facile de commettre des erreurs de frappe. Au fait, ENGINE MyISAM indique à MySQL le type de moteur de base de données à utiliser pour cette table. Ici aussi, même si le jeu de caractères d'encodage est déjà indiqué par défaut au niveau de la base de données, nous ajoutons CHARSET utf8 pour imposer à la table la gestion correcte des caractères accentués.

Pour vérifier que la table a été créée, entrez la commande suivante :

```
DESCRIBE classiques;
```

Si tout s'est bien déroulé, vous voyez la séquence de commandes et de réponses illustrée dans l'exemple 8-4. Remarquez en particulier la mise en forme de la table, telle qu'elle est affichée.

Exemple 8-4. Une session MySQL : création et vérification d'une table

```
mysql> USE publications;
Database changed
mysql> CREATE TABLE classiques (
    ->  auteur VARCHAR(128),
    ->  titre VARCHAR(128),
    ->  type VARCHAR(16),
    ->  annee CHAR(4)) ENGINE MyISAM
    ->  CHARSET utf8;
Query OK, 0 rows affected (0.02 sec)

mysql> DESCRIBE classiques;
+---------+---------------+-------+-----+---------+-------+
| Field   | Type          | Null  | Key | Default | Extra |
+---------+---------------+-------+-----+---------+-------+
| auteur  | varchar(128)  | YES   |     | NULL    |       |
| titre   | varchar(128)  | YES   |     | NULL    |       |
| type    | varchar(16)   | YES   |     | NULL    |       |
| annee   | char(4)       | YES   |     | NULL    |       |
+---------+---------------+-------+-----+---------+-------+
4 rows in set (0.02 sec)
```

La commande `DESCRIBE` est d'une aide incontournable au débogage, lorsque vous devez vérifier que vous avez créé correctement une table MySQL. Elle vous permet aussi de vous rappeler les noms des champs ou colonnes, ainsi que le type de chaque donnée. Examinons les détails des entêtes du tableau.

`Field`
: Le nom de chaque champ ou colonne au sein de la table.

`Type`
: Le type de donnée MySQL stockée dans le champ.

`Null`
: Indique si le champ est autorisé à contenir une valeur `NULL`.

`Key`
: MySQL prend en charge des clés (*keys*) ou des index, qui permettent de rechercher très rapidement des données. L'entête `Key` désigne le type de clé appliqué à ce champ, le cas échéant.

`Default`
: La valeur par défaut affectée au champ lorsqu'aucune valeur n'a encore été précisée pour ce champ et lors de la création d'une nouvelle ligne.

`Extra`
: Des informations supplémentaires, comme le fait que le champ est réglé pour s'incrémenter automatiquement.

Types de données

L'exemple 8-3 indique que trois champs de la table ont reçu le type de donnée `VARCHAR`, tandis que le dernier a reçu le type `CHAR`. `CHAR` signifie caractère, tandis que `VARCHAR` désigne des chaines de caractères de longueur variable. Ces commandes reçoivent un nombre qui indique à MySQL le nombre maximum de caractères permis pour une chaine stockée dans le champ correspondant.

Ce type de donnée est très utile car MySQL peut planifier la taille des bases de données et cela lui permet d'effectuer plus facilement des recherches. L'inconvénient se situe dans le fait que si vous tentez d'affecter une chaine de longueur supérieure à la longueur autorisée, elle sera tronquée à la longueur maximale déclarée dans la définition de la table.

Le champ `annee` a, quant à lui, une longueur plus prévisible donc, au lieu de lui attribuer le type `VARCHAR`, nous lui attribuons le type `CHAR(4)`, plus efficace. Le paramètre 4 permet de stocker quatre octets de données et de prendre en charge toutes les années de -999 à 9999. Un octet comprend huit bits et peut contenir les valeurs 00000000 à 11111111, ce qui correspond à 0 jusqu'à 255 en décimal.

Vous pourriez bien entendu stocker uniquement les années à deux chiffres mais, si les données dépassent le siècle actuel, elles seraient tronquées et, comme pour le bogue de l'an 2000, provoqueraient le déport des dates débutant le 1er janvier 2000 vers le 1er janvier 1900, comme sur nombre d'installations informatiques parmi les plus vastes au monde.

J'ai préféré ne pas utiliser le type `YEAR` de MySQL dans la table classiques parce qu'il ne prend en charge que l'année 0000, ainsi que les années de 1901 à 2155. Ceci est dû au fait que MySQL stocke l'année dans un seul octet pour des raisons d'efficacité, ce qui entraine que seules 256 années sont disponibles, alors que les années de publication des ouvrages de la table classiques datent de bien avant 1901.

Les deux types `CHAR` et `VARCHAR` acceptent des chaines de texte et imposent une limite sur la taille du champ. La différence réside dans le fait que toute chaine d'un champ `CHAR` a exactement la taille précisée. Si vous y placez une chaine de longueur inférieure, elle est complétée d'espaces. Un champ `VARCHAR` ne complète pas le texte : il laisse le champ varier pour s'accommoder du texte inséré. En revanche, le `VARCHAR` exige un peu plus de travail pour suivre la taille de chaque valeur. Par conséquent, le `CHAR` est légèrement plus efficace si les longueurs sont très proches les unes des autres dans tous les enregistrements, tandis que le `VARCHAR` s'avère plus efficace quand les longueurs varient beaucoup et prennent une certaine importance. En outre, cette surcharge de travail entraine un petit ralentissement d'accès à des données en `VARCHAR` par rapport à des données en `CHAR`.

Type de donnée CHAR

Le tableau 8-6 énumère les variantes du type de donnée CHAR. Tous ces types prennent un paramètre pour préciser la longueur maximale ou exacte de la chaine permise dans le champ. Le tableau montre que chaque type possède une valeur maximale d'octets qu'il peut accueillir.

Tableau 8-6. Types de données CHAR de MySQL

Type de donnée	Octets utilisés	Exemples
CHAR(*n*)	Exactement *n* (<= 255)	CHAR(7) *"Bonjour" occupe 7 octets* CHAR(57) *"Au revoir" occupe 57 octets*
VARCHAR(*n*)	Jusque *n* (<= 65535)	VARCHAR(7) *"Bonjour" occupe 7 octets* VARCHAR(100) *"Bonne nuit" occupe 10 octets*

Type de donnée BINARY

Le type de donnée BINARY sert à stocker des chaines d'octets (*bytes*) complets qui ne s'associent à aucun jeu de caractères. Par exemple, le type de donnée BINARY permet de stocker une image GIF (tableau 8-7).

Tableau 8-7. Types de données BINARY de MySQL

Type de donnée	Octets utilisés	Exemples
BINARY(*n*) ou BYTE(*n*)	Exactement *n* (<= 255)	Comme CHAR mais contient des données binaires
VARBINARY(*n*)	Jusque *n* (<= 65535)	Comme VARCHAR mais contient des données binaires

Types de données TEXT et VARCHAR

Les différences entre les types TEXT et VARCHAR sont assez ténues :

- Avant sa version 5.0.3, MySQL supprimait les espaces de début et de fin des champs VARCHAR.
- Les champs TEXT ne peuvent pas avoir de valeur par défaut.
- MySQL n'indexe que les *n* premiers caractères d'une colonne de type TEXT (le *n* est précisé au moment de la création de l'index).

Il en résulte que le VARCHAR est préférable, car c'est un type de donnée plus rapide et meilleur si vous devez effectuer une recherche sur la totalité du contenu d'un tel champ. En revanche, si vous effectuez des recherches sur seulement un certain nombre de caractères du début de ce champ, vous pouvez alors utiliser le type de donnée TEXT. Le tableau 8-8 reprend les variantes de taille maximum du type TEXT : petite (TINYTEXT), normale (TEXT), moyenne (MEDIUMTEXT) et très grande (LONGTEXT).

Tableau 8-8. Types de données TEXT de MySQL

Type de donnée	Octets utilisés	Attributs
TINYTEXT(*n*)	Jusque *n* (<= 255)	Traité comme une chaine avec un jeu de caractères
TEXT(*n*)	Jusque *n* (<= 65535)	Traité comme une chaine avec un jeu de caractères
MEDIUMTEXT(*n*)	Jusque *n* (<= 1,67e+7)	Traité comme une chaine avec un jeu de caractères
LONGTEXT(*n*)	Jusque *n* (<= 4,29e+9)	Traité comme une chaine avec un jeu de caractères

Type de donnée BLOB

Le terme BLOB signifie *grand objet binaire* (*Binary Large OBject*) et, comme vous pouvez vous y attendre, il se prédestine à des données binaires dont la taille dépasse les 65 536 octets. L'autre principale différence entre les types de donnée BLOB et BINARY se situe au niveau de la valeur par défaut : le BLOB ne peut pas en avoir (tableau 8-9).

Tableau 8-9. Types de données BLOB de MySQL

Type de donnée	Octets utilisés	Attributs
TINYBLOB(*n*)	Jusque *n* (<= 255)	Traité comme des données binaires — sans jeu de caractères
BLOB(*n*)	Jusque *n* (<= 65535)	Traité comme des données binaires — sans jeu de caractères
MEDIUMBLOB(*n*)	Jusque *n* (<= 1,67e+7)	Traité comme des données binaires — sans jeu de caractères
LONGBLOB(*n*)	Jusque *n* (<= 4,29e+9)	Traité comme des données binaires — sans jeu de caractères

Types de données numériques

MySQL prend en charge de nombreux types de données numériques, du plus petit octet jusqu'au nombre à virgule flottante en double précision. Bien que la quantité maximale de mémoire qu'un champ numérique peut occuper se situe à huit octets, veillez toujours à adopter le type de donnée le moins gourmand possible, capable de recevoir la plus grande valeur de vos données que vous prévoyez, car cela permet de conserver des bases de données petites et plus rapidement accessibles.

Le tableau 8-10 énumère les types de données numériques pris en charge par MySQL et les plages de valeurs qu'ils peuvent contenir. Si vous ne connaissez pas ces termes, un *nombre signé* est un nombre qui s'inscrit dans une plage de valeurs à partir d'une valeur négative, 0, jusqu'à une valeur maximale positive. Tandis qu'un *nombre non signé* part de 0 jusqu'à une valeur maximale positive. Ces deux types de plages de valeurs contiennent autant de nombres différents possibles, mais vous pouvez vous les représenter, avec les nombres signés décalés à mi-chemin de part et d'autre du 0 : la moitié de ces nombres sont négatifs, l'autre moitié de ces nombres sont positifs. Notez que les valeurs en virgule (point) flottante sont obligatoirement signées, quelle qu'en soit la précision.

Tableau 8-10. Types de données numériques de MySQL

Type de donnée	Octets utilisés	Valeur minimale		Valeur maximale	
		Signée	Non signée	Signée	Non signée
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8.38e+6	0	8.38e+6	1.67e+7
INT / INTEGER	4	-2.15e+9	0	2.15e+9	4.29e+9
BIGINT	8	-9.22e+18	0	9.22e+18	1.84e+19
FLOAT	4	-3.40e+38	*S.O.*	3.4e+38	*S.O.*
DOUBLE / REAL	8	-1.80e+308	*S.O.*	1.80e+308	*S.O.*

Pour indiquer qu'un type de donnée numérique est non signé, utilisez le qualificateur `UNSIGNED`. Ainsi, l'exemple suivant crée une table nommée *nomtable* avec un champ dénommé *nonchamp*, du type de donnée entier non signé (`UNSIGND INTEGER`) :

```
CREATE TABLE nomtable (nonchamp INT UNSIGNED);
```

Lors de la création d'un champ numérique, vous pouvez aussi lui passer un paramètre numérique facultatif, comme suit :

```
CREATE TABLE nomtable (nonchamp INT(4));
```

Mais soyez extrêmement prudent avec cette formulation car, contrairement aux types de données `BINARY` et `CHAR`, ce paramètre n'indique pas le nombre d'octets nécessaires au stockage. Ceci peut sembler tout sauf intuitif, mais ce nombre représente en fait la longueur d'affichage en nombre de chiffres lors de la récupération du contenu du champ. Il est habituellement associé au qualificateur `ZEROFILL`, comme ceci :

```
CREATE TABLE nomtable (nonchamp INT(4) ZEROFILL);
```

Ceci indique que tout nombre de longueur inférieure à quatre chiffres sera complété par un ou plusieurs zéros, de manière à atteindre une longueur de champ de quatre caractères. Lorsque le champ comporte le nombre indiqué de caractères, aucun zéro n'est ajouté en complément.

DATE et TIME

Les derniers principaux types de données pris en charge par MySQL sont ceux relatifs aux dates et aux heures. Le tableau 8-11 les décrit en détail.

Tableau 8-11. Types de données DATE et TIME de MySQL

Type de donnée	Format de date/heure
DATETIME	'0000-00-00 00:00:00'
DATE	'0000-00-00'
TIMESTAMP	'0000-00-00 00:00:00'
TIME	'00:00:00'
YEAR	0000 (Années 0000 et de 1901 à 2155 seulement)

Les types `DATETIME` et `TIMESTAMP` s'affichent de la même manière. La principale différence entre eux se situe dans une très petite plage de dates pour un `TIMESTAMP` (des années 1970 à 2037), tandis que `DATETIME` est capable de contenir toute date imaginable, ce qui peut s'avérer important si vous vous intéressez à l'Antiquité ou à la science-fiction.

Le type `TIMESTAMP` demeure toutefois très utile parce que vous pouvez laisser MySQL en définir la valeur à votre place. Si vous ne précisez pas de valeur lors de l'ajout d'une ligne dans une table, la date courante y est automatiquement insérée. Vous pouvez aussi demander à MySQL de mettre à jour un `TIMESTAMP` chaque fois que vous modifiez une ligne.

Type de donnée AUTO_INCREMENT

Il est souvent nécessaire de faire en sorte que chaque ligne d'une base de données soit unique à coup sûr. Il est possible de garantir cela dans un programme en vérifiant soigneusement les données que vous entrez et de vous assurer qu'une valeur au moins diffère entre deux lignes et, ceci, pour toutes les lignes d'une table, mais cette approche s'avère propice aux erreurs et ne fonctionne correctement que dans certaines circonstances. Dans la table *classiques*, par exemple, un même auteur peut apparaitre dans plusieurs lignes. De même, l'année de publication n'offre aucune garantie d'absence de doublons et ainsi de suite. Il est difficile de garantir que vous n'ayez pas de lignes en double.

La solution générale consiste à utiliser une colonne supplémentaire à cette seule fin. Nous verrons dans quelques instants l'utilisation de l'ISBN, le numéro international normalisé du livre, mais auparavant, je voudrais présenter le type de donnée `AUTO_INCREMENT`.

Comme le suggère le nom de ce type, une colonne de ce type de donnée règle la valeur de son contenu à celle de l'entrée de colonne insérée précédemment, plus 1. L'exemple 8-5 illustre l'ajout d'une colonne dénommée `id` à la table *classiques*, avec auto-incrémentation.

Exemple 8-5. Ajout de la colonne auto-incrémentée id

```
ALTER TABLE classiques ADD id INT UNSIGNED NOT NULL AUTO_INCREMENT KEY;
```

Ceci permet aussi de présenter la commande `ALTER`, très semblable à `CREATE`. `ALTER` s'applique à une table existante et permet de modifier, d'ajouter ou de supprimer des colonnes. Notre exemple ajoute une colonne nommée *id*, avec les caractéristiques suivantes :

`INT UNSIGNED`
: Donne à la colonne un entier suffisamment grand pour stocker jusqu'à quatre milliards d'enregistrements distincts dans la table.

`NOT NULL`
: Garantit que chaque champ de cette colonne a une valeur. Nombre de programmeurs exploitent `NULL` dans un champ pour indiquer qu'il ne contient aucune

valeur. Mais cela entrainerait le risque de doublons, ce qui enfreindrait la raison principale de l'existence de cette colonne. Donc nous interdisons les valeurs nulles.

AUTO_INCREMENT
: Indique à MySQL de définir une valeur unique pour cette colonne dans toutes les lignes, comme décrit plus haut. Nous n'avons pas vraiment de contrôle sur la valeur que cette colonne prend à chaque enregistrement mais cela importe peu. L'important est de garantir qu'elle soit unique.

KEY
: Une colonne auto-incrémentée est aussi utile en tant que clé d'index, parce que vous aurez tendance à rechercher des enregistrements en fonction de cette colonne. Nous expliquerons cette notion à la section Les index, page 186.

Chaque entrée dans la colonne *id* aura désormais un numéro unique, le premier étant 1 et les suivants augmentant de 1, donc se suivant dans l'ordre de la séquence, à chaque nouvelle insertion de ligne.

Au lieu d'ajouter cette colonne de manière rétroactive, vous auriez pu l'inclure directement lors de l'envoi de la commande CREATE selon une forme légèrement différente. Dans ce cas, il aurait fallu récrire l'exemple 8-3 à la façon de l'exemple 8-6. Examinez en particulier la dernière ligne.

Exemple 8-6. Ajout de la colonne auto-incrémentée id à la création de la table

```
CREATE TABLE classiques (
 auteur VARCHAR(128),
 titre VARCHAR(128),
 type VARCHAR(16),
 annee CHAR(4),
 id INT UNSIGNED NOT NULL AUTO_INCREMENT KEY)
 ENGINE MyISAM CHARSET utf8;
```

Pour vérifier que la colonne a bien été ajoutée, utilisez la commande suivante pour afficher les colonnes et les types de données de la table :

```
DESCRIBE classiques;
```

Maintenant que nous avons vu à quoi pouvait servir une telle colonne, nous n'en avons plus besoin. Donc si vous l'avez créée avec la commande de l'exemple 8-5, supprimez-la à l'aide de l'exemple 8-7.

Exemple 8-7. Suppression de la colonne id

```
ALTER TABLE classiques DROP id;
```

Ajouter des données à une table

La commande INSERT permet d'insérer des lignes dans une table. Pour la voir en action, nous ajoutons des données à la table *classiques* issues du tableau 8-1, à l'aide d'une forme répétitive de la commande INSERT (exemple 8-8).

Exemple 8-8. Remplissage de la table classiques à l'aide de données

```
INSERT INTO classiques(auteur, titre, type, annee)
 VALUES('Mark Twain','Les aventures de Tom Sawyer','Roman','1876');
INSERT INTO classiques(auteur, titre, type, annee)
 VALUES('Jane Austen','Orgueil et préjugés','Roman','1811');
INSERT INTO classiques(auteur, titre, type, annee)
 VALUES('Charles Darwin','De l''origine des espèces','Scientifique','1856');
INSERT INTO classiques(auteur, titre, type, annee)
 VALUES('Charles Dickens','Le Magasin d''antiquités','Roman','1841');
INSERT INTO classiques(auteur, titre, type, annee)
 VALUES('William Shakespeare','Roméo et Juliette','Tragédie','1594');
```

Après chaque seconde ligne, le message Query OK s'affiche. Remarquez la présence des doubles apostrophes, par exemple dans la chaine De l''origine des espèces. En SQL, lorsqu'une chaine est entourée d'apostrophes verticales (') pour insérer une apostrophe au sein de la chaine, vous devez doubler l'apostrophe dans le texte (''). De même, quand une chaine est entourée de guillemets verticaux ("), si un guillemet apparait au sein de la chaine, vous devez le dédoubler (""), pour que MySQL ne le considère pas comme la fin de la chaine. Ce sont des séquences d'échappement (en PHP, nous avions vu \' et \" à cet effet).

Dès que vous avez entré toutes les lignes, entrez la commande suivante, qui affiche les données contenues dans la table (figure 8-4) :

```
SELECT * FROM classiques;
```

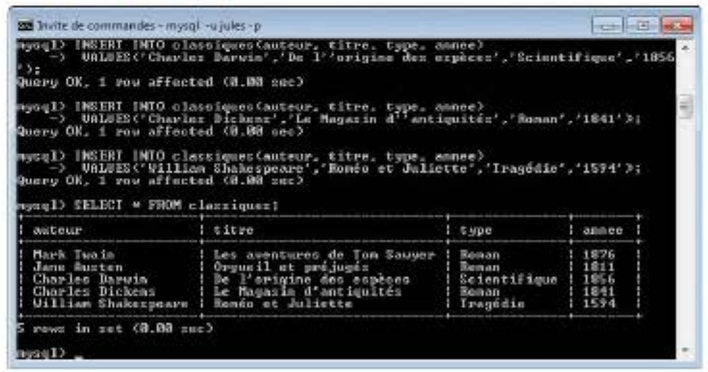


Figure 8-4. Remplissage de la table classiques et affichage de son contenu

Ne vous attardez pas sur la commande SELECT pour l'instant. Nous y reviendrons à la section Interroger une base de données MySQL, page 192. Pour l'instant, il suffit de considérer qu'elle permet d'afficher les données que vous venez d'entrer.

Revenons à la commande INSERT et à son utilisation. La première partie, INSERT INTO classiques, indique à MySQL où insérer les données qui suivent. Ensuite, apparaissent entre parenthèses les quatre noms de colonnes, *auteur*, *titre*, *type* et *annee*, séparés par des virgules. Ils indiquent à MySQL que ce sont les champs où il doit insérer les données.

La deuxième ligne de chaque commande INSERT contient le mot clé VALUES, suivi de quatre chaines entre parenthèses et séparées par des virgules. Ceci fournit à MySQL les quatre valeurs à insérer respectivement dans les quatre colonnes nommées précédemment. (Une fois encore, mon choix de découper les lignes est purement arbitraire.)

Chaque donnée est insérée dans la colonne correspondante, de proche en proche. Si vous énumérez accidentellement les données dans un ordre différent de celui des noms des champs, les données vont dans des colonnes incorrectes et le nombre de colonnes doit être égal au nombre de données fournies.

Renommer une table

Renommer une table, comme pour tout changement de structure ou toute modification des métadonnées d'une table, est réalisé à l'aide de la commande ALTER. Ainsi, pour modifier le nom de la table *classiques* en *pre1900*, écrivez :

```
ALTER TABLE classiques RENAME pre1900;
```

Si vous avez envoyé cette commande, pour revenir au nom de table précédent et pour que les exemples de la suite fonctionnent comme indiqué, entrez la commande suivante :

```
ALTER TABLE pre1900 RENAME classiques;
```

Modifier le type d'une colonne

Le changement de type de donnée d'une colonne fait également appel à la commande ALTER, en combinaison cette fois avec le mot clé MODIFY. Ainsi, pour changer le type de donnée de la colonne annee, de CHAR(4) en SMALLINT (qui n'exige que 2 octets de stockage au lieu de 4, donc qui réduit la taille de la base de données), entrez la commande suivante :

```
ALTER TABLE classiques MODIFY annee SMALLINT;
```

Lors de cette modification, si la conversion du type de donnée a un sens pour MySQL, il modifie automatiquement les données en conservant ce sens. Dans ce cas-ci, il remplace chaque chaine par un entier comparable, car il reconnait la chaine comme convertible en un entier.

Ajouter une colonne

Supposons que vous ayez créé une table et que vous l'ayez remplie de données. Vous vous rendez compte que vous avez besoin d'une colonne supplémentaire. Pas d'inquiétude, voici comment ajouter la colonne *pages*, qui servira à stocker le nombre de pages de chaque publication :

```
ALTER TABLE classiques ADD pages SMALLINT UNSIGNED;
```


Cette commande ajoute une colonne nommée *pages*, de type de donnée petit entier non signé, SMALLINT UNSIGNED, suffisant pour contenir une valeur qui peut atteindre 65 535, soit bien assez pour n'importe quel ouvrage publié!

Lorsque vous demandez ensuite à MySQL de décrire la table modifiée à l'aide de la commande DESCRIBE, comme suit, vous constatez que la modification a été effectuée avec succès (figure 8-5) :

```
DESCRIBE classiques;
```

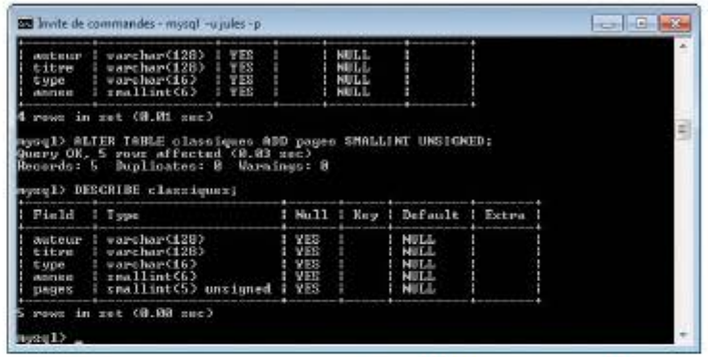


Figure 8-5. Ajout de la colonne pages et affichage de la structure de la table

Renommer une colonne

À la vue de la figure 8-5, vous considérez que la colonne *type* prête à confusion, parce que c'est le nom utilisé par MySQL pour identifier les types de données. Vous décidez donc de renommer cette colonne en *categorie*, comme suit :

```
ALTER TABLE classiques CHANGE type categorie VARCHAR(16);
```

Remarquez l'ajout de VARCHAR(16) à la fin de la commande. Le mot clé CHANGE impose de préciser le type de donnée, même si vous ne souhaitez pas le modifier, et VARCHAR(16) est le type de donnée avec lequel la colonne *type* a été créée à l'origine.

Supprimer une colonne

Finalement, après réflexion, la colonne *pages* ne vous semble pas particulièrement utile dans cette base de données et vous décidez de la supprimer. Le mot clé DROP sert à cela :

```
ALTER TABLE classiques DROP pages;
```

Retenez que le DROP est une opération irréversible, donc utilisez-le toujours avec précaution. Vous risquez de supprimer par inadvertance toute une table, voire même une base de données, si vous ne faites pas attention !

Supprimer une table

La suppression d'une table est tout aussi facile mais, comme le but n'est pas de vous faire entrer toutes les données de la table *classiques*, créons rapidement une table, vérifions son existence, puis supprimons-la à l'aide des commandes de l'exemple 8-9. La figure 8-6 illustre les résultats des quatre commandes de l'exemple.

Exemple 8-9. Création, affichage et suppression d'une table

```
CREATE TABLE supprimable(poubelle INT);
DESCRIBE supprimable;
DROP TABLE supprimable;
SHOW tables;
```

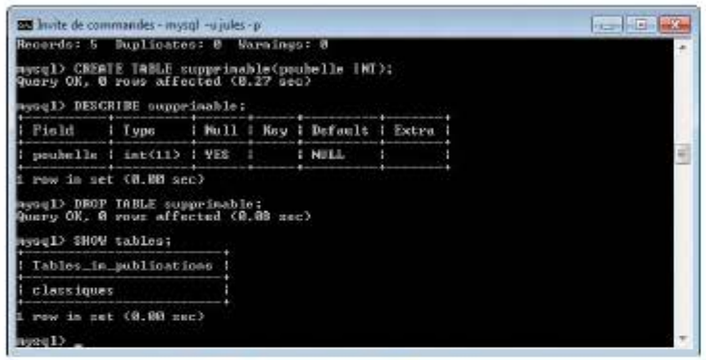


Figure 8-6. Création, affichage et suppression d'une table

Les index

Telles que les choses ont été définies jusqu'à présent, la table *classiques* fonctionne et permet des recherches sans difficulté à l'aide de MySQL. Mais la table peut grossir et accueillir quelques centaines de lignes de publications. À ce stade, les accès à la base de données deviennent de plus en plus lents, à mesure que viennent s'ajouter de nouvelles lignes, car MySQL doit inspecter chaque ligne de données lors de chaque requête. Cela revient à ouvrir tous les livres d'une bibliothèque, chaque fois que vous recherchez une information.

Bien entendu, vous ne mèneriez pas une recherche de la sorte dans une bibliothèque. La bibliothèque possède un système d'index sur fiches ou, plus vraisemblablement, dans une base de données propre. MySQL fonctionne de la même manière. Au prix d'une petite surcharge en mémoire et dans l'espace disque, la création d'un index sur une table permet à MySQL de mener des recherches à la vitesse de la lumière.

Créer un index

L'exécution de recherches très rapides passe par l'ajout préalable d'un *index*, que ce soit au moment de la création d'une table ou n'importe quand par la suite. Le choix d'un index ne relève pas d'une tâche simple. D'abord, parce qu'il existe plusieurs types d'index, comme les `INDEX` normaux, selon une clé primaire (`PRIMARY KEY`) ou encore en texte complet (`FULL TEXT`). Ensuite, parce qu'il faut choisir les colonnes qui nécessitent une indexation, une décision qui relève de la prédiction de la manière dont les recherches seront menées parmi les données de telle ou telle colonne. Un index peut s'avérer très complexe, dans la mesure où il combine plusieurs colonnes. Et même lorsque vous avez pris votre décision, il vous reste l'option de réduire la taille d'un index en limitant la portion de chaque colonne à indexer.

Lorsque nous essayons d'anticiper les recherches qu'un utilisateur pourrait effectuer dans la table *classiques*, il apparait évident que toutes les colonnes méritent une recherche. Cependant, si nous avions conservé la colonne *pages* créée à la section Ajouter une colonne, page 184, elle n'aurait pas eu besoin d'un index parce qu'il semble peu probable de devoir rechercher un livre par son nombre de pages. Pour l'instant, nous allons ajouter un index à chacune des colonnes, à l'aide des commandes de l'exemple 8-10.

Exemple 8-10. Ajout d'index à la table classiques

```
ALTER TABLE classiques ADD INDEX(auteur(20));
ALTER TABLE classiques ADD INDEX(titre(20));
ALTER TABLE classiques ADD INDEX(categorie(4));
ALTER TABLE classiques ADD INDEX(annee);
DESCRIBE classiques;
```

Les deux premières lignes créent un index sur chacune des deux colonnes *auteur* et *titre*, mais en limitent la largeur aux 20 premiers caractères. Ainsi, quand MySQL indexe le titre `Les aventures de Tom Sawyer`, il ne stocke dans l'index que les 20 premiers caractères, comme suit :

```
Les aventures de Tom
```

Ceci a pour but de réduire la taille de l'index et d'optimiser la vitesse d'accès à la base de données. J'ai choisi de réduire la largeur d'indexation à 20, parce qu'elle est très probablement suffisante pour garantir l'unicité de la plupart des chaines dans ces colonnes. Si MySQL devait trouver deux entrées identiques dans un index, il prendrait le temps d'examiner la table et de vérifier le contenu de la colonne indexée pour trouver les lignes de correspondance exacte.

Dans le cas de la colonne *categorie*, pour l'instant, seul le premier caractère est nécessaire pour identifier une chaine comme unique (R pour Roman, S pour Scientifique, T pour Tragédie), mais je préfère prendre quatre caractères pour anticiper de futures catégories qui ne seraient uniques qu'à partir de quatre lettres. Il est toujours possible de réindexer une colonne par la suite, lorsque la plage de valeurs possibles pour les catégories commence à s'étoffer. Et enfin, je n'impose aucune limite à l'index de la colonne *annee* car il s'agit d'un entier et non plus d'une chaine.

La figure 8-7 illustre l'envoi de ces commandes (et du DESCRIBE qui confirme le succès des opérations). Remarquez la présence de la clé MUL dans chaque colonne, qui signifie que plusieurs occurrences (ou doublons) d'une même valeur peuvent apparaître dans cette colonne. Or, ceci est précisément ce que nous voulons, car un auteur peut apparaître plusieurs fois, un même livre peut avoir plusieurs auteurs, et ainsi de suite.

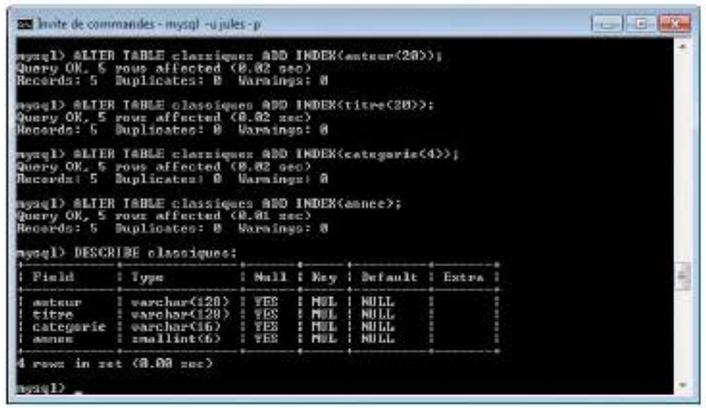


Figure 8-7. Ajout d'index à la table classiques

Utiliser CREATE INDEX

Outre ALTER TABLE, une autre possibilité existe pour ajouter un index : la commande CREATE INDEX. Elles sont équivalentes, sauf que CREATE INDEX ne permet pas de créer un index de type PRIMARY KEY (voir Clés primaires, page 189). La seconde ligne de l'exemple 8-11 illustre le format de cette commande.

Exemple 8-11. Les deux commandes suivantes sont équivalentes

```
ALTER TABLE classiques ADD INDEX(auteur(20));
CREATE INDEX auteur ON classiques (auteur(20));
```

Ajouter des index lors de la création d'une table

Il n'est pas nécessaire d'attendre après la création d'une table pour lui ajouter des index. En pratique, la création d'index après coup exige parfois beaucoup de temps, surtout lorsque la table à indexer contient déjà de très nombreux enregistrements. Par conséquent, examinons une commande qui crée la table *classiques* avec ses index en même temps.

L'exemple 8-12 est une reprise de l'exemple 8-3, mais cette fois avec la création des index en même temps que la table. Remarquez que, pour intégrer les modifications apportées par ce chapitre, cette nouvelle version utilise le nouveau nom de colonne

categorie à la place de *type* et qu'elle définit de suite le type de donnée d'*annee* à SMALLINT au lieu de CHAR(4). Si vous voulez tester cet exemple sans détruire la table *classiques* initiale, modifiez le nom *classiques* de la première ligne en quelque chose du genre *classiques1*, puis supprimez *classiques1* lorsque vous avez terminé.

Exemple 8-12. Création de la table classiques avec ses index

```
CREATE TABLE classiques (
 auteur VARCHAR(128),
 titre VARCHAR(128),
 categorie VARCHAR(16),
 annee SMALLINT,
 INDEX(auteur(20)),
 INDEX(titre(20)),
 INDEX(categorie(4)),
 INDEX(annee)) ENGINE MyISAM CHARSET utf8;
```

Clés primaires

À ce stade, nous avons une table *classiques*, dans laquelle nous avons fait en sorte que MySQL puisse y effectuer des recherches rapides grâce à des index. Mais il lui manque encore quelque chose. Il est possible de rechercher parmi toutes les publications, mais il n'existe pas encore de clé unique associée à chaque ouvrage pour garantir l'accès immédiat et sans équivoque à une ligne. L'importance de disposer d'une clé avec une valeur unique pour chaque enregistrement apparaitra plus clairement lorsque nous combinerons des données de plusieurs tables.

La section Type de donnée AUTO_INCREMENT de la page 181 a présenté brièvement la notion de clé primaire, lors de la création de la colonne *id* auto-incrémentée, qui aurait pu servir de clé primaire pour cette table. Cependant, je préfère réserver ce rôle à une colonne plus appropriée : le numéro ISBN.

Alors, allons-y et créons une nouvelle colonne pour cette clé. Sachant que le numéro ISBN nécessite 13 caractères, vous pourriez penser que la commande suivante suffirait :

```
ALTER TABLE classiques ADD isbn CHAR(13) PRIMARY KEY;
```

Hélas non. En fait, si vous l'essayez, vous obtenez l'erreur Duplicate entry for key 1, c'est-à-dire une erreur de doublon de clé pour cet index. La raison de ceci provient de la présence d'enregistrements dans la table or, cette commande tente d'ajouter une colonne et lui affecte la valeur NULL à chaque ligne, ce qui n'est pas permis dans le cas d'une clé primaire, puisque toutes les valeurs d'une colonne d'index de clé primaire doivent être uniques. En revanche, si aucune donnée ne figurait dans la table, comme à sa création, par exemple, alors cette commande fonctionnerait très bien.

Dans notre situation, nous devons tricher un peu et créer la nouvelle colonne sans aucun index, la remplir de données uniques pour chaque enregistrement, puis seulement ajouter l'index à l'aide des commandes de l'exemple 8-13. Par chance, chacune des années de publication est unique dans le jeu de données actuel, ce qui nous permet

d'identifier chaque ligne pour la mise à jour. Remarquez que cet exemple fait appel aux mots clés UPDATE et WHERE, que nous détaillerons à la section Interroger une base de données MySQL, à la page 192.

Remarquez que les numéros ISBN n'ont ici pas d'importance capitale et que ce sont ceux de versions anglaises. L'important réside dans l'unicité de ces valeurs.

Exemple 8-13. Remplissage de la colonne isbn de données et définition de la clé primaire

```
ALTER TABLE classiques ADD isbn CHAR(13);
UPDATE classiques SET isbn='9781598184891' WHERE annee='1876';
UPDATE classiques SET isbn='9780582506206' WHERE annee='1811';
UPDATE classiques SET isbn='9780517123201' WHERE annee='1856';
UPDATE classiques SET isbn='9780099533474' WHERE annee='1841';
UPDATE classiques SET isbn='9780192814968' WHERE annee='1594';
ALTER TABLE classiques ADD PRIMARY KEY(isbn);
DESCRIBE classiques;
```

Dès que vous avez entré ces commandes, comparez les résultats à ceux de la figure 8-8. Remarquez que les mots clés PRIMARY KEY remplacent le mot clé INDEX dans la syntaxe de la commande ALTER TABLE (comparez l'exemple 8-10 et l'exemple 8-13).

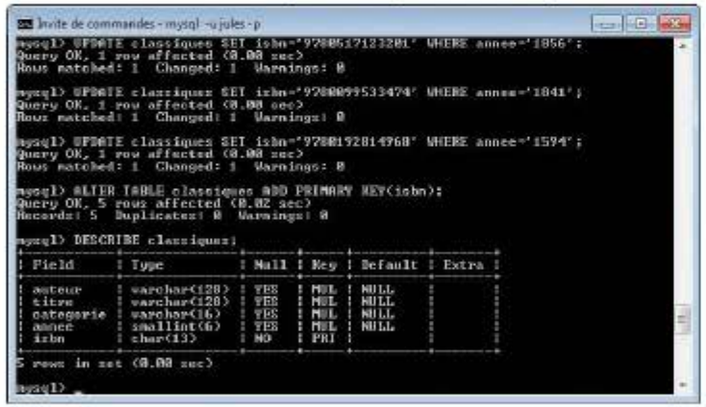


Figure 8-8. Ajout à posteriori d'une clé primaire à la table classiques

Pour créer la clé primaire au moment de la création de la table *classiques*, vous auriez pu utiliser les commandes de l'exemple 8-14. À nouveau, pour éviter de détruire la table *classiques* existante, renommez *classiques* à la première ligne en autre chose avant de reproduire cet exemple, puis supprimer cette table après le test.

Exemple 8-14. Création de la table classiques avec sa clé primaire

```
CREATE TABLE classiques (
 auteur VARCHAR(128),
 titre VARCHAR(128),
 categorie VARCHAR(16),
 annee SMALLINT,
 isbn CHAR(13),
 INDEX(auteur(20)),
 INDEX(titre(20)),
 INDEX(categorie(4)),
 INDEX(annee),
 PRIMARY KEY (isbn)) ENGINE MyISAM CHARSET utf8;
```


Créer un index FULLTEXT

Au contraire d'un index normal, l'index FULLTEXT de MySQL permet des recherches extrêmement rapides dans des colonnes entières de texte. Il stocke chaque chaine de donnée dans un index spécial, à partir duquel il est possible d'effectuer des recherches à l'aide d'un « langage naturel », un peu comme à l'aide d'un moteur de recherche.

Dire que MySQL stocke tous les mots dans un index FULLTEXT n'est pas tout à fait exact. En pratique, il possède une liste de plus de 500 mots anglais qu'il choisit d'ignorer parce qu'ils sont tellement communs qu'ils n'ont pas de très grande utilité dans le cadre des recherches textuelles. Cette liste, appelée liste de *mots vides* (*stopwords*), reprend des mots comme *the*, *as*, *is*, *of* et ainsi de suite (et en français, *le*, *la*, *de*, *est*, *à*, etc.). Cette liste permet à MySQL de réagir beaucoup plus vite et de conserver des bases de données de taille réduite. L'annexe C reprend la liste complète des mots vides.

Les éléments à connaitre à propos des index FULLTEXT sont les suivants :

- Les index FULLTEXT ne fonctionnent qu'avec des tables MyISAM (et, à partir de la version 5.6.0 de MySQL, avec les tables INNODB), le type de table utilisé par défaut par le moteur de stockage de MySQL (alors qu'il en supporte 10 différents types). Si vous devez convertir une table en MyISAM, utilisez la commande MySQL ALTER TABLE nomtable ENGINE = MyISAM;.
- Un index FULLTEXT n'est applicable que sur des colonnes de type CHAR, VARCHAR ou TEXT.
- La définition d'un index FULLTEXT est possible lors de la création d'une table, dans l'instruction CREATE TABLE, ou par la suite, à l'aide d'ALTER TABLE ou de CREATE INDEX.
- Pour de vastes jeux de données, il est *beaucoup* plus rapide de charger les données dans une table sans index FULLTEXT, puis de créer l'index par la suite, plutôt que de charger les données dans une table déjà équipée d'un index FULLTEXT.

Pour créer un index FULLTEXT, appliquez-le à un ou plusieurs champs, comme illustré à l'exemple 8-15, qui ajoute un index FULLTEXT à la paire de colonnes *auteur* et *titre* dans la table *classiques* (cet index vient en sus de ceux déjà créés et ne les influence pas).

Exemple 8-15. Ajout d'un index FULLTEXT à la table classiques

```
ALTER TABLE classiques ADD FULLTEXT(auteur,titre);
```

Vous pouvez ensuite effectuer des recherches en texte clair parmi cette paire de colonnes. Cette fonctionnalité revêt un authentique intérêt si vous pouvez accéder à la totalité du texte des ouvrages et l'ajouter à la base de données (surtout s'ils ne sont pas soumis aux droits d'auteur) car, avec un tel index, il est alors possible d'effectuer des recherches de mots, d'expressions et ainsi de suite. La section MATCH...AGAINST de la page 197 décrit les recherches de type FULLTEXT.

Lorsque MySQL semble fonctionner plus lentement que d'habitude quand vous accédez à une base de données, la cause se situe généralement au niveau des index. Soit vous n'avez pas d'index là où il en faudrait un, soit les index ne sont pas conçus de manière optimale. L'affinage des index d'une table permet souvent de régler ce genre de problème. Les notions relatives aux performances ne font pas partie de la portée de ce livre, mais le chapitre 9 vous donne quelques pistes pour que vous sachiez où chercher.

Interroger une base de données MySQL

Jusqu'ici, nous nous sommes concentrés sur la création d'une base de données, de tables, sur l'ajout de données et d'index pour accélérer les recherches. Il est maintenant temps d'examiner les manières d'effectuer des recherches avec les différentes commandes et leurs qualificateurs disponibles.

SELECT

À la figure 8-4, une commande SELECT a permis d'extraire des données d'une table. Dans cette section, nous avons utilisé la forme la plus simple de la commande pour sélectionner toutes les données et les afficher. En pratique, ne faites ce genre de chose que sur de très petites tables avec peu d'enregistrements, sinon les données s'affichent les unes à la suite des autres d'une manière illisible. Examinons la commande SELECT en détail.

La syntaxe de base est la suivante :

```
SELECT quelquechose FROM nomtable;
```

Le *quelquechose* peut être un * (astérisque), comme nous l'avons vu précédemment, ce qui signifie toutes les colonnes, ou les noms de seulement quelques colonnes. Ainsi, l'exemple 8-16 illustre la sélection des seules colonnes *auteur* et *titre*, puis *titre* et *isbn*. La figure 8-9 montre les résultats de ces commandes.

Exemple 8-16. Deux instructions SELECT sur la table classiques

```
SELECT auteur,titre FROM classiques;
SELECT titre,isbn FROM classiques;
```


Figure 8-9. Les résultats de deux instructions SELECT différentes

SELECT COUNT

Une alternative au paramètre *quelquechose* consiste à utiliser COUNT, qui peut s'utiliser de bien des manières. Ainsi, à l'exemple 8-17, il affiche le nombre de la table lorsqu'il reçoit * comme paramètre, ce qui signifie toutes les lignes. Comme vous l'aurez compris, le résultat renvoyé est 5, puisque la table contient cinq ouvrages.

Exemple 8-17. Compter les lignes

```
SELECT COUNT(*) FROM classiques;
```

SELECT DISTINCT

Le qualificateur DISTINCT (ou son synonyme, DISTINCTROW) permet d'éliminer plusieurs entrées pour n'en conserver qu'une, lorsque ces entrées contiennent les mêmes données. Par exemple, lorsque vous souhaitez une liste de tous les auteurs dans la table, si vous ne sélectionnez que la colonne *auteur* alors que la table contient plusieurs livres du même auteur, vous obtenez une longue liste avec le même auteur repris plusieurs fois. L'ajout du mot clé DISTINCT permet de n'afficher qu'une seule fois chaque auteur. Pour voir comment cela fonctionne, nous allons ajouter une ligne avec un autre ouvrage d'un auteur existant (exemple 8-18).

Exemple 8-18. Ajout de données avec un auteur en double

```
INSERT INTO classiques(auteur, titre, categorie, annee, isbn)
 VALUES('Charles Dickens','La Petite Dorrit','Roman','1857', '9780141439969');
```

À présent, Charles Dickens apparaît deux fois dans la table et nous pouvons comparer l'utilisation de SELECT avec et sans le qualificateur DISTINCT. L'exemple 8-19 et la figure 8-10 illustrent le simple SELECT qui énumère deux fois Charles Dickens et la commande avec le qualificateur DISTINCT qui ne l'affiche qu'une seule fois.

Exemple 8-19. SELECT avec et sans le qualificateur DISTINCT

```
SELECT auteur FROM classiques;
SELECT DISTINCT auteur FROM classiques;
```

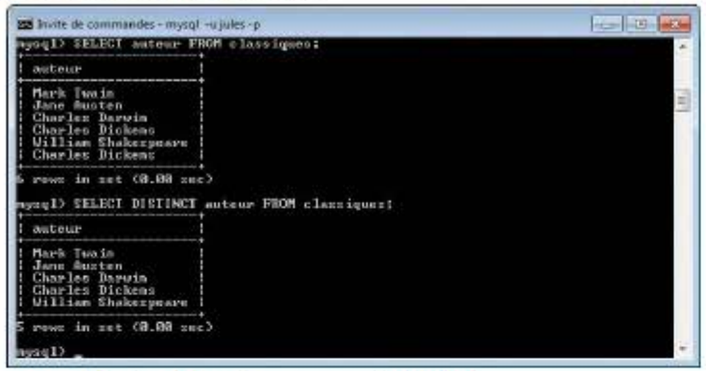


Figure 8-10. Sélection de données sans et avec DISTINCT

DELETE

Pour supprimer une ligne dans une table, utilisez la commande `DELETE`. Sa syntaxe ressemble à celle de la commande `SELECT` et permet de restreindre la sélection à la ligne exacte souhaitée ou à plusieurs lignes en fonction de critères définis dans les clauses `WHERE` et `LIMIT`.

Maintenant que vous avez vu les effets du qualificateur `DISTINCT`, si vous avez entré la commande de l'exemple 8-18, vous pouvez supprimer *La Petite Dorrit* de la table à l'aide de la commande de l'exemple 8-20.

Exemple 8-20. Suppression de la nouvelle entrée

```
DELETE FROM classiques WHERE titre='La Petite Dorrit';
```

Cette commande revient à dire de supprimer toutes les lignes dont la colonne *titre* contient la chaine `La Petite Dorrit`.

Le mot clé `WHERE` présente une grande puissance. Il importe de stipuler soigneusement et correctement cette clause, car une erreur d'entrée risque de supprimer des lignes non souhaitées (ou de n'avoir aucun effet si aucune ligne ne correspond au contenu de la clause `WHERE`). L'importance de cette clause, qui constitue le cœur même de SQL, est telle que nous allons lui consacrer un certain temps.

WHERE

Le mot clé `WHERE` permet d'affiner des requêtes de sélection pour ne cibler que les lignes où (*where*) une certaine expression est vraie. L'exemple 8-20 précédent ne renvoyait que les lignes dont la colonne titre était exactement égale à la chaine `La Petite Dorrit`, du fait de l'opérateur =. L'exemple 8-21 propose deux autres exemples d'utilisation de `WHERE` avec =.

Exemple 8-21. Utilisation du mot clé WHERE

```
SELECT auteur,titre FROM classiques WHERE auteur="Mark Twain";
SELECT auteur,titre FROM classiques WHERE isbn="9781598184891 ";
```

Étant donné l'état de la table, ces deux commandes renvoient le même résultat. Mais nous pourrions facilement ajouter d'autres livres de Mark Twain, auquel cas la première ligne afficherait tous les ouvrages de cet écrivain, tandis que la seconde ligne continuerait d'afficher seulement `Les aventures de Tom Sawyer`, parce que nous savons que le numéro ISBN est unique. En d'autres termes, les recherches basées sur une clé unique s'avèrent plus prévisibles. D'ailleurs, vous verrez plus loin la portée de cette évidence d'unicité de valeur dans le cadre des clés uniques et primaires.

Vous pouvez aussi utiliser un motif de correspondance dans vos recherches, grâce au qualificateur `LIKE`, qui signifie « comme », en remplacement de l'opérateur =. Ce qualificateur permet de rechercher des portions de chaines. Il s'utilise souvent en conjonction avec le caractère `%`, avant ou après du texte. Lorsque placé avant un mot, `%` signifie *quoi qu'il y ait avant* ce mot. Placé après un mot clé, il signifie *quoi qu'il y ait après* ce mot. L'exemple 8-22 exécute trois requêtes, l'une à partir du début d'une chaine, la suivante pour la fin de la chaine et la dernière pour retrouver le mot n'importe où dans la chaine. La figure 8-11 illustre les résultats obtenus par ces trois commandes.

Exemple 8-22. Utilisation du qualificateur LIKE

```
SELECT auteur,titre FROM classiques WHERE auteur LIKE "Charles%";
SELECT auteur,titre FROM classiques WHERE titre LIKE "%espèces";
SELECT auteur,titre FROM classiques WHERE titre LIKE "%et%";
```

La première commande affiche les ouvrages à la fois de Charles Darwin et de Charles Dickens car le qualificateur `LIKE` indique de renvoyer toute chaine commençant par Charles et suivi de n'importe quel texte. Ensuite, seule la chaine `De l'origine des espèces` est renvoyée, parce que c'est la seule ligne dont la colonne se termine par `espèces`. Enfin, les deux résultats `Orgueil et préjugés` et `Roméo et Juliette` apparaissent à l'affichage parce qu'ils contiennent tous deux le mot `et` n'importe où dans le *titre*.

Le `%` donne aussi un résultat de correspondance s'il n'y a rien à l'emplacement qu'il occupe ; autrement dit, il correspond aussi à une chaine vide.

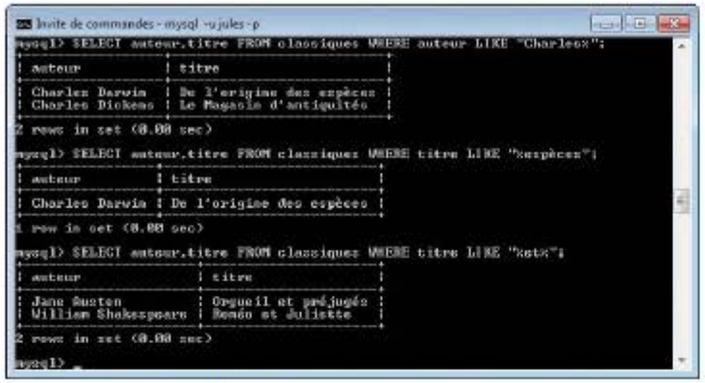


Figure 8-11. Utilisation de WHERE avec le qualificateur LIKE

LIMIT

Le qualificateur `LIMIT` permet de décider du nombre de lignes à renvoyer dans une requête et à partir de quel endroit dans la table il faut renvoyer ce nombre de lignes. Lorsqu'il ne reçoit qu'un paramètre, cela indique à MySQL de commencer au début de la liste des résultats et de ne renvoyer que le nombre indiqué de lignes à partir du début. Si vous passez deux paramètres à `LIMIT`, le premier indique le décalage à partir du début de la liste de résultats et le second, le nombre de lignes à renvoyer. Dans ce cas-ci, vous pouvez vous représenter le premier paramètre comme le fait de dire « ignorer ce nombre de résultats depuis le début ».

L'exemple 8-23 inclut trois commandes. La première renvoie les trois premières lignes de la table. La deuxième renvoie deux lignes à partir de l'emplacement 1 (donc ignore la première ligne). La dernière commande renvoie une seule ligne à partir de l'emplacement 3 (donc ignore les trois premières lignes). La figure 8-12 illustre les résultats de l'envoi de ces trois commandes.

Exemple 8-23. Limitation du nombre de résultats renvoyés

```
SELECT auteur,titre FROM classiques LIMIT 3;
SELECT auteur,titre FROM classiques LIMIT 1,2;
SELECT auteur,titre FROM classiques LIMIT 3,1;
```

Soyez très prudent avec le mot clé `LIMIT` car le décalage débute à 0 mais le nombre de lignes à renvoyer débute à 1. Donc `LIMIT 1,3` signifie renvoyer *trois* lignes à partir de la *deuxième* ligne.

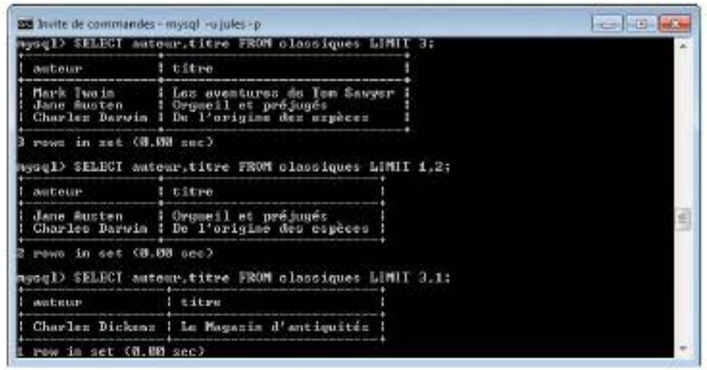


Figure 8-12. Réduction des lignes retournées à l'aide de LIMIT

MATCH...AGAINST

La construction `MATCH...AGAINST` ne sert que pour des colonnes qui possèdent un index `FULLTEXT` (voir la section Créer un index FULLTEXT à la page 191). Grâce à elle, vous pouvez réaliser des recherches en langage naturel au même titre que vous le feriez dans un moteur de recherche de l'internet. Contrairement à `WHERE...=` ou `WHERE...LIKE`, `MATCH...AGAINST` permet d'entrer plusieurs mots dans une requête de recherche et les compare à tous les mots des colonnes `FULLTEXT`. Les index `FULLTEXT` ne sont pas sensibles à la casse, donc la casse que vous utilisez dans votre requête ne revêt aucune importance.

En supposant que vous ayez ajouté un index `FULLTEXT` aux colonnes *auteur* et *titre*, vous pourriez entrer les trois requêtes de l'exemple 8-24. La première demande de renvoyer toute ligne qui, parmi ces colonnes, contient le mot `et`. Or, comme `et` est un mot vide (*stopword*), MySQL l'ignore et cette requête renvoie systématiquement un ensemble vide, que `et` y figure ou non. La deuxième requête demande d'afficher toutes les lignes qui contiennent en elles à la fois les mots *magasin* et *antiquités*, dans n'importe quel ordre. Enfin, la dernière requête applique le même genre de recherche pour les mots *tom* et *sawyer*. La figure 8-13 montre les résultats de ces requêtes.

Exemple 8-24. Utilisation de MATCH...AGAINST sur des index FULLTEXT

```
SELECT auteur,titre FROM classiques
 WHERE MATCH(auteur,titre) AGAINST('et');
SELECT auteur,titre FROM classiques
 WHERE MATCH(auteur,titre) AGAINST('magasin antiquités');
SELECT auteur,titre FROM classiques
 WHERE MATCH(auteur,titre) AGAINST('tom sawyer');
```

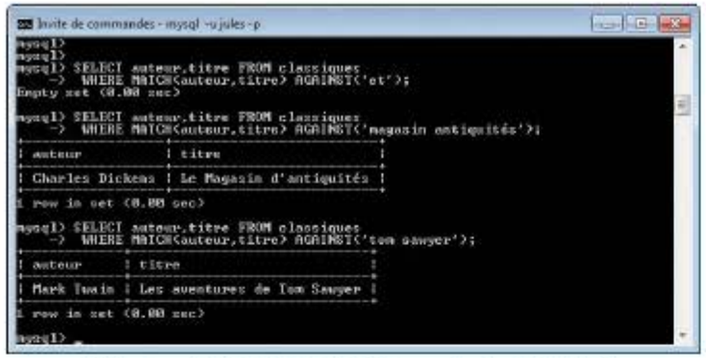


Figure 8-13. Utilisation de MATCH...AGAINST sur un index FULLTEXT

MATCH...AGAINST en mode booléen

Pour apporter encore plus de puissance à vos requêtes `MATCH...AGAINST`, utilisez le mode booléen, `IN BOOLEAN MODE`. Celui-ci modifie l'effet de la requête `FULLTEXT` normale de manière à rechercher toute combinaison des mots recherchés, au lieu de simplement imposer que tous les mots recherchés soient présents dans le texte. La seule présence d'un des mots recherchés dans la colonne provoque le retour de la ligne.

Le mode booléen permet aussi de préfixer d'un signe + ou – les mots recherchés pour indiquer s'ils doivent être inclus ou rejetés. Si le mode booléen normal dit qu'« un seul mot de la liste suffit pour retourner la ligne », le signe + impose que « ce mot doit être présent ; sinon, ne pas retourner la ligne ». Le signe – signifie que « tel mot ne doit pas être présent ; sa présence disqualifie la ligne de tout retour ».

L'exemple 8-25 illustre le mode booléen à travers deux requêtes. La première demande toutes les lignes contenant le mot *charles* et non le mot *espèces*. La seconde utilise des guillemets pour demander toutes les lignes contenant la phrase exacte *origine des*. La figure 8-14 montre les résultats de ces requêtes.

Exemple 8-25. Utilisation de MATCH...AGAINST en mode booléen

```
SELECT auteur,titre FROM classiques
 WHERE MATCH(auteur,titre)
 AGAINST('+charles -espèces' IN BOOLEAN MODE);
SELECT auteur,titre FROM classiques
 WHERE MATCH(auteur,titre)
 AGAINST('"origine des"' IN BOOLEAN MODE);
```

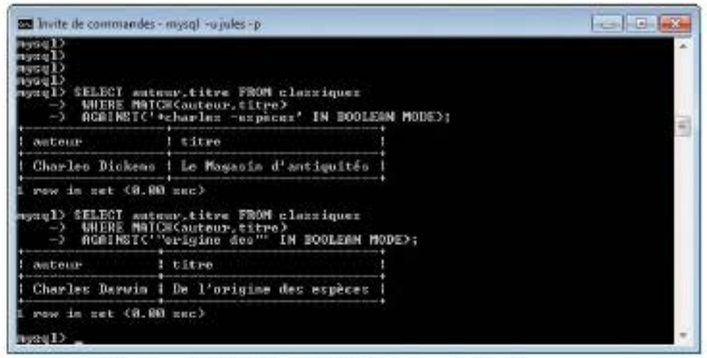



Figure 8-14. Utilisation de MATCH...AGAINST en mode booléen

Comme vous vous y attendiez, la première requête ne renvoie que `Le Magasin d'antiquités` de Charles Dickens, parce que toutes les lignes contenant *espèces* ont été exclues, donc le livre de Charles Darwin a été ignoré.

Une remarque intéressante concerne la seconde requête : le mot vide *de* fait partie de la chaîne de recherche, mais il demeure présent dans le résultat car les guillemets ont la priorité sur les mots vides.

UPDATE...SET

Cette construction permet de modifier le contenu d'un champ. Lorsque vous souhaitez modifier le contenu d'un ou de plusieurs champs, vous devez d'abord réduire la portée de la modification aux seuls champs à modifier, d'une manière comparable à la commande `SELECT`. L'exemple 8-26 présente deux formes d'utilisation de commande `UPDATE...SET` et la figure 8-15 montre les résultats.

Exemple 8-26. Utilisation d'UPDATE...SET

```
UPDATE classiques SET auteur='Mark Twain (Samuel Langhorne Clemens)'
 WHERE auteur='Mark Twain';
UPDATE classiques SET categorie='Roman classique'
 WHERE categorie='Roman';
```

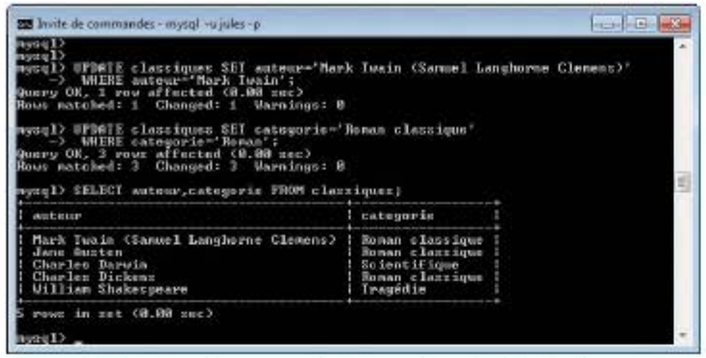


Figure 8-15. Modification du contenu de colonnes dans la table classiques

Dans la première commande, le nom réel de Mark Twain, Samuel Langhorne Clemens, vient s'ajouter entre parenthèses à la suite de son nom de plume, pour n'affecter qu'une seule ligne. La deuxième commande, en revanche, affecte trois lignes et remplace toutes les occurrences du mot *Roman* de la colonne *categorie* par *Roman classique*.

Lors de l'exécution d'une modification, vous disposez aussi des qualificateurs que vous avez déjà vus, comme `LIMIT`, mais aussi les mots clés `ORDER BY` et `GROUP BY` suivants.

ORDER BY

La clause `ORDER BY` trie les résultats renvoyés dans une ou plusieurs colonnes en ordre alphanumérique croissant (`ASC`, par défaut) ou décroissant (`DESC`). L'exemple 8-27 illustre deux requêtes de ce type, dont les résultats s'affichent à la figure 8-16.

Exemple 8-27. Utilisation d'ORDER BY

```
SELECT auteur,titre FROM classiques ORDER BY auteur;
SELECT auteur,titre FROM classiques ORDER BY titre DESC;
```

Ainsi, la première requête retourne les ouvrages par ordre alphabétique croissant (par défaut) d'auteurs, tandis que la seconde les retourne dans l'ordre décroissant des titres.

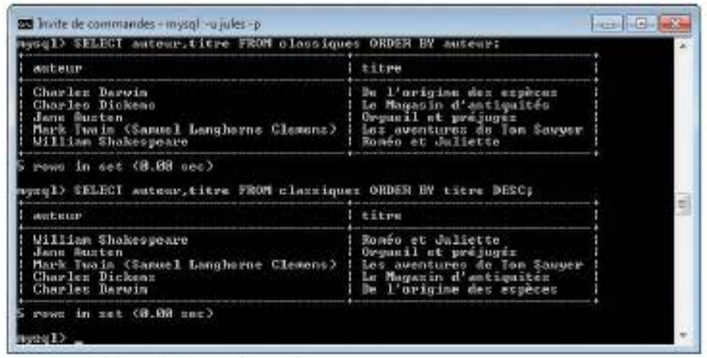


Figure 8-16. Tri des résultats de requêtes

Si vous voulez trier toutes les lignes par *auteur*, puis par *annee* décroissante de publication (pour afficher les plus récents en premier), utilisez la requête suivante :

```
SELECT auteur,titre,annee FROM classiques ORDER BY auteur,annee DESC;
```

Ceci indique que chaque qualificateur `ASC` ou `DESC` s'applique indépendamment sur une seule colonne. Le mot clé `DESC` ne s'applique qu'à la colonne qui le précède, soit annee. Comme la clause indique qu'*auteur* suit l'ordre de tri par défaut, les auteurs sont classés par ordre croissant. Vous auriez pu préciser de manière explicite l'ordre croissant pour la colonne *auteur*, avec les mêmes résultats au final :

```
SELECT auteur,titre,annee FROM classiques ORDER BY auteur ASC,annee DESC;
```

GROUP BY

Dans le même ordre d'idée qu'`ORDER BY`, vous pouvez regrouper des résultats renvoyés par des requêtes à l'aide de `GROUP BY`, dont le principal intérêt réside dans l'obtention d'informations relatives à un groupe de données. Par exemple, lorsque vous souhaitez connaître le nombre de publications existantes pour chacune des catégories de la table *classiques*, entrez la commande suivante :

```
SELECT categorie,COUNT(auteur) FROM classiques GROUP BY categorie;
```

Cette commande retourne les résultats suivants :

```
+------------------+---------------+
| categorie        | COUNT(auteur) |
+------------------+---------------+
| Roman classique  |             3 |
| Scientifique     |             1 |
| Tragédie         |             1 |
+------------------+---------------+
3 rows in set (0.00 sec)
```

Joindre deux tables

Il est assez normal de conserver plusieurs tables au sein d'une même base de données, contenant chacune des informations de types différents. Ainsi, examinons le cas d'une table *clients* qui doit contenir les noms de clients et faire référence aux publications achetées parmi celles présentes dans la table *classiques*. L'exemple 8-28 comporte des commandes pour créer cette table et la remplir de trois clients avec leurs achats. La figure 8-17 illustre les résultats obtenus.

Exemple 8-28. Création et remplissage de la table clients

```
CREATE TABLE clients (
 nom VARCHAR(128),
 isbn VARCHAR(13),
 PRIMARY KEY (isbn)) ENGINE MyISAM CHARSET utf8;
INSERT INTO clients(nom,isbn)
 VALUES('Jean Bergeron','9780099533474');
INSERT INTO clients(nom,isbn)
 VALUES('Marie Savard','9780582506206');
INSERT INTO clients(nom,isbn)
 VALUES('Jacques Vachon','9780517123201');
SELECT * FROM clients;
```

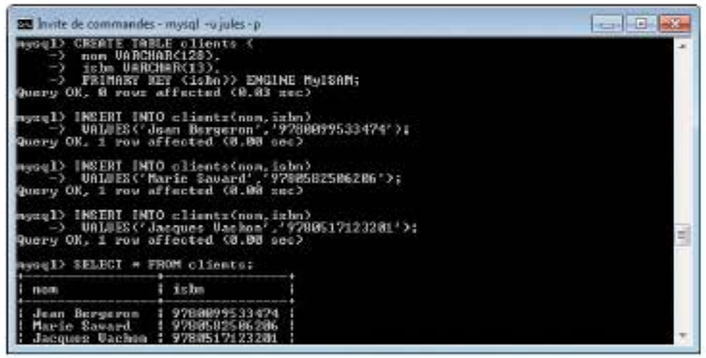


Figure 8-17. Création et remplissage de la table clients

Remarquez qu'il existe un raccourci pour insérer plusieurs rangées dans une table, comme le fait l'exemple 8-28: il est permis de remplacer les trois instructions `INSERT INTO` par une seule, avec une liste des données à insérer, entourées de parenthèses et séparées par des virgules, comme suit:

```
INSERT INTO clients(nom,isbn) VALUES
('Jean Bergeron','9780099533474'),
('Marie Savard','9780582506206'),
('Jacques Vachon','9780517123201');
```


Il est évident qu'une véritable table destinée à contenir des informations sur des clients reprendrait d'autres informations comme des adresses, des numéros de téléphone, des adresses de courrier électronique et ainsi de suite, mais ces informations ne seraient pas utiles pour les explications qui suivent. Lors de la création de cette table, vous avez peut-être remarqué qu'elle détient un point commun avec la table *classiques*: une colonne nommée *isbn*. Comme elle a la même signification dans les deux tables (le numéro ISBN fait référence à un livre et toujours au même livre), nous pouvons utiliser cette colonne pour lier, associer les deux tables dans une même requête, comme dans l'exemple 8-29.

Exemple 8-29. Jointure de deux tables dans un même SELECT

```
SELECT nom,auteur,titre from clients,classiques
 WHERE clients.isbn=classiques.isbn;
```

Les résultats de cette opération sont les suivants:

```
+------------------+------------------+---------------------------+
| nom              | auteur           | titre                     |
+------------------+------------------+---------------------------+
| Jean Bergeron    | Charles Dickens  | Le Magasin d'antiquités   |
| Marie Savard     | Jane Austen      | Orgueil et préjugés       |
| Jacques Vachon   | Charles Darwin   | De l'origine des espèces  |
+------------------+------------------+---------------------------+
3 rows in set (0.00 sec)
```

Voyez-vous comment cette requête a joliment lié les deux tables pour afficher les ouvrages de la table *classiques*, achetés par les clients de la table *clients*?

Jointure naturelle: NATURAL JOIN

La clause `NATURAL JOIN` permet de vous épargner des frappes au clavier et de clarifier quelque peu les requêtes. Ce type de jointure prend deux tables et joint automatiquement les colonnes qui portent le même nom. Pour obtenir les mêmes résultats qu'à l'exemple 8-29, vous pouvez écrire:

```
SELECT nom,auteur,titre from clients NATURAL JOIN classiques;
```

JOIN...ON

Si vous préférez préciser clairement la colonne de jointure des deux tables, utilisez la construction `JOIN...ON` comme suit, pour obtenir les mêmes résultats qu'à l'exemple 8-29:

```
SELECT nom,auteur,titre from clients
 JOIN classiques ON clients.isbn=classiques.isbn;
```

Utiliser AS

Pour vous éviter quelques frappes au clavier et améliorer la lisibilité d'une requête, vous pouvez aussi créer des alias à l'aide du mot clé `AS`. Dans la clause `FROM`, indiquez le nom d'une table, suivi d'`AS` (qui signifie « comme ») et d'un *alias* à utiliser. Le code qui suit donne encore les mêmes résultats que l'exemple 8-29. Les alias s'avèrent surtout

utiles dans les longues requêtes qui font intervenir de nombreuses tables et font référence à ces tables à de multiples reprises.

```
SELECT nom,auteur,titre from
  clients AS cli,classiques AS livres WHERE cli.isbn=livres.isbn;
```

Les résultats de cette opération sont les suivants :

```
+----------------+----------------+-------------------------+
| nom            | auteur         | titre                   |
+----------------+----------------+-------------------------+
| Jean Bergeron  | Charles Dickens | Le Magasin d'antiquités |
| Marie Savard   | Jane Austen    | Orgueil et préjugés     |
| Jacques Vachon | Charles Darwin | De l'origine des espèces |
+----------------+----------------+-------------------------+
3 rows in set (0.00 sec)
```

Utiliser les opérateurs logiques

Les opérateurs logiques AND (et), OR (ou) et NOT (pas) sont disponibles dans les requêtes WHERE pour encore restreindre les sélections. L'exemple 8-30 montre l'utilisation de chacun d'eux dans des requêtes distinctes, mais vous pouvez les combiner à souhait.

Exemple 8-30. Utilisation des opérateurs logiques

```
SELECT auteur,titre FROM classiques WHERE
 auteur LIKE "Charles%" AND auteur LIKE "%Darwin";
SELECT auteur,titre FROM classiques WHERE
 auteur LIKE "%Mark Twain%" OR auteur LIKE "%Samuel Langhorne Clemens%";
SELECT auteur,titre FROM classiques WHERE
 auteur LIKE "Charles%" AND auteur NOT LIKE "%Darwin";
```

J'ai rédigé la première requête ainsi, parce que Charles Darwin peut figurer dans certains cas avec son nom complet, Charles Robert Darwin. Ainsi, la requête renvoie les publications dont la colonne *auteur* commence par *Charles* et se termine par *Darwin*. La deuxième requête recherche les publications à partir du nom de plume de Mark Twain ou de son vrai nom, Samuel Langhorne Clemens. La troisième requête renvoie les publications écrites par les auteurs dont le prénom est Charles mais dont le nom de famille *n'est pas* Darwin.

Fonctions de MySQL

Vous vous demandez sans doute qui pourrait souhaiter utiliser les fonctions de MySQL alors que PHP propose toute une panoplie de fonctions utiles de son côté. La réponse est très simple : les fonctions de MySQL opèrent sur les données présentes dans la base de données, tandis que si vous utilisez PHP, vous devrez extraire des données brutes de MySQL, les manipuler et enfin effectuer la requête que vous souhaitez sur la base de données.

Le fait que des fonctions soient disponibles dans MySQL réduit de manière substantielle le temps nécessaire pour effectuer des requêtes complexes, mais réduit également leur complexité. Si vous souhaitez en savoir davantage sur les fonctions disponibles de chaines et de date ou heure, consultez les deux sites suivants (en anglais) :

http://tinyurl.com/mysqlstrings
http://tinyurl.com/mysqldates

Cependant, pour vous aider à débuter, l'annexe D décrit un sous-ensemble des fonctions les plus utiles.

Accéder à MySQL par l'entremise de phpMyAdmin

Si vous devez apprendre les principales commandes et connaitre leur fonctionnement pour utiliser MySQL, dès que vous les avez comprises, il peut s'avérer plus rapide et plus simple de faire appel à un programme tel que phpMyAdmin pour gérer vos bases de données et vos tables.

Pour cela, entrez l'adresse URL suivante pour appeler la page principale de XAMPP de la figure 8-18 :

```
http://localhost/xampp
```

Figure 8-18. Le tableau de bord de XAMPP

Cliquez ensuite sur le lien *phpMyAdmin* dans le bas du menu de gauche pour ouvrir le programme (figure 8-19).

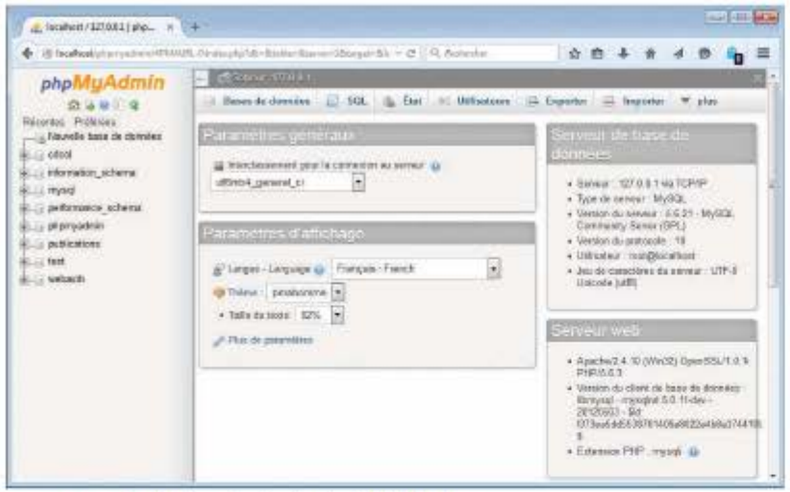


Figure 8-19. La fenêtre principale de phpMyAdmin

Dans le volet de gauche de l'écran principal de phpMyAdmin, cliquez sur une des bases de données sur laquelle vous souhaitez travailler. Ceci entraine l'ouverture de la base de données et l'affichage de ses tables. Vous pouvez aussi cliquer sur *Nouvelle base de données pour en créer une*.

À partir de cet écran principal, vous pouvez exécuter toutes les opérations principales, comme créer des bases de données, des tables, des index et bien plus encore. Pour lire la documentation de phpMyAdmin, consultez *https://docs.phpmyadmin.net/fr/latest/*.

Si vous avez suivi les explications de ce chapitre et essayé tous les exemples proposés, félicitations! Ce fut un long voyage à travers le monde de MySQL. Vous avez appris à créer une base de données, mais également à rédiger des requêtes complexes qui combinent plusieurs tables, à exploiter les opérateurs booléens et les différents qualificateurs de MySQL.

Au chapitre suivant, nous allons aborder la conception efficace d'une base de données, les techniques avancées en SQL, ainsi que les fonctions et les transactions de MySQL.

Questions

1. Quel est le but du point-virgule dans les requêtes MySQL ?
2. Quelle commande devez-vous utiliser pour voir toutes les bases de données et tables disponibles ?
3. Comment créer sur l'hôte local un nouvel utilisateur sous MySQL, nommé *nouvel-utilisateur*, avec le mot de passe *nouveaupasse* et avec l'accès à tout dans la base de données *nouvellebase* ?
4. Comment peut-on voir la structure d'une table ?
5. Quel est le but d'un index MySQL ?
6. Quels avantages un index `FULLTEXT` apporte-t-il ?
7. Qu'est-ce qu'un mot vide ?
8. `SELECT DISTINCT` et `GROUP BY` permettent l'affichage de seulement une ligne en sortie pour chaque valeur d'une colonne, même si plusieurs lignes contiennent cette valeur. Quelles sont les principales différences entre `SELECT DISTINCT` et `GROUP BY` ?
9. À l'aide de `SELECT...WHERE`, comment faites-vous pour ne renvoyer que les lignes qui contiennent quelque part le mot *Langhorne* dans la colonne auteur de la table *classiques* utilisée dans ce chapitre ?
10. Que faut-il définir dans deux tables pour permettre la jointure de celles-ci ?

Retrouvez les réponses du chapitre 8 dans l'annexe A.

CHAPITRE 9

Maitriser MySQL

Au chapitre 8, vous avez acquis de bonnes bases dans la pratique des bases de données relationnelles à l'aide du langage SQL. Vous avez appris à créer des bases de données et les tables qu'elles contiennent, mais aussi à insérer des données, les rechercher, les modifier et les supprimer.

Forts de ces connaissances, nous allons aborder la conception de bases de données qui favorise la vitesse maximale et l'efficacité. Par exemple, comment décider des données à disposer dans quelles tables ? En fait, au fil des ans, des règles ont été établies pour garantir (si vous les suivez) l'efficacité et la capacité à croitre des bases de données, à mesure que vous leur ajoutez de plus en plus de données.

Conception de base de données

Il est extrêmement important de bien concevoir une base de données avant même de la créer ; sinon, vous devrez certainement revenir en arrière pour en modifier la structure par l'éclatement de certaines tables en plusieurs, la fusion d'autres tables et le déplacement de certaines colonnes pour établir des relations raisonnables et réfléchies qui permettent à MySQL de les exploiter.

Le fait de vous assoir devant une feuille de papier, avec un crayon et une gomme, pour réfléchir à (et dessiner) une sélection de requêtes que vous et vos utilisateurs serez amenés à émettre, constitue déjà un excellent point de départ. Dans le cas d'une base de données destinée à une librairie en ligne, voici quelques questions qui peuvent s'avérer utiles pour y réfléchir :

- Combien y a-t-il d'auteurs, de livres et de clients dans la base de données ?
- Quel auteur a écrit tel livre ?
- Quels sont les livres écrits par tel auteur ?
- Quel est le livre le plus cher ?
- Quel livre affiche les meilleures ventes ?
- Quels livres ne se sont pas vendus cette année ?
- Quels sont les livres achetés par un client particulier ?
- Quels autres livres se sont vendus en même temps que certains autres livres ?

Bien entendu, le nombre de requêtes que vous pouvez constituer pour interroger une telle base de données est sans limite, mais ces quelques exemples vous apportent déjà un aperçu de la manière de présenter vos tables. Ainsi, il faudra probablement regrouper les livres et les numéros ISBN dans une seule table, parce qu'ils sont intimement liés (nous verrons plus loin quelques subtilités à ce propos). À l'inverse, les livres et les clients occuperont des tables distinctes, parce que leur interconnexion s'avère très lâche, soit l'inverse d'intime. Un client peut acheter un livre quelconque et même plusieurs exemplaires d'un livre, tandis qu'un livre peut être acheté par de nombreux clients différents ou ignoré par les clients potentiels.

Lorsque vous prévoyez effectuer de nombreuses recherches sur quelque chose, il peut s'avérer avantageux de lui attribuer sa propre table. Et lorsque les couplages entre les choses sont ténus, il vaut mieux les placer dans des tables séparées.

La prise de conscience et le respect de ces règles fondamentales permettent déjà de dégager une piste : il faudra au moins trois tables pour concrétiser ces requêtes :

Auteurs
: De nombreuses recherches porteront sur les auteurs, dont nombre parmi eux ont collaboré à certains titres d'ouvrages, plusieurs d'entre eux seront présentés dans des collections. Le fait d'énumérer toutes les informations sur chaque auteur, reliées à tel auteur, produit des résultats optimums dans les recherches, d'où l'existence d'une table *Auteurs*.

Livres
: De nombreux livres apparaissent dans des éditions différentes. Parfois, ils changent d'éditeur et parfois, ils portent le même titre qu'un autre livre sans aucun rapport. Donc les liens entre les livres et les auteurs s'avèrent suffisamment compliqués pour définir une table distincte.

Clients
: La nécessité d'une table réservée aux clients apparait comme la plus évidente, puisqu'ils ont toute liberté pour acquérir n'importe quel livre de n'importe quel auteur.

Clés primaires : les clés des bases de données relationnelles

Grâce à la puissance des *bases de données relationnelles*, nous pouvons définir des informations spécifiques pour chaque auteur, livre et client dans un seul emplacement déterminé. Bien évidemment, ce sont les liens entre eux qui nous intéressent, comme qui a écrit chaque livre et qui l'a acheté, mais nous pouvons constituer ces informations en établissant des liens entre les trois tables. Je vous montrerai les principes de base et vous verrez qu'il suffit d'un peu de pratique pour que les réflexes deviennent naturels.

Pour que la magie opère, il faut avant tout que chaque auteur reçoive un identifiant unique, de même pour chaque livre et chaque client. Le chapitre précédent vous a déjà donné le moyen d'aboutir à cela, avec la *clé primaire*. Dans le cas d'un livre, il semble logique d'utiliser le numéro ISBN, bien qu'il faille alors gérer la possibilité qu'un même livre paraisse sous plusieurs éditions, avec des ISBN différents. Pour les auteurs

et les clients, il suffira de leur attribuer des clés arbitraires, que la fonctionnalité `AUTO_INCREMENT` vue au chapitre précédent simplifie fortement.

En bref, chaque table doit être conçue autour d'un concept susceptible de faire l'objet de recherches fréquentes, dans ce cas-ci, un auteur, un livre ou un client, et ce concept doit porter une clé primaire. Ne choisissez pas comme clé quelque chose qui risque de contenir une même valeur pour des objets différents. L'ISBN constitue un des rares cas pour lesquels l'industrie a défini d'emblée une clé primaire à laquelle vous pouvez vous fier pour obtenir un identifiant unique de chaque produit. La plupart du temps, vous devez vous-même définir une clé (issue d'une décision) arbitraire à cette fin, à l'aide de l'`AUTO_INCREMENT`.

Normalisation

Le processus de répartition des données dans des tables et de création de clés primaires porte le nom de *normalisation*. Son principal but consiste à garantir que chaque information apparaisse une seule fois dans une même base de données. Le dédoublement des données est inefficace, parce qu'il fait gonfler inutilement la base de données et ralentit les accès aux données. Plus grave encore, la présence de doublons entraine le risque de ne modifier qu'une seule copie d'une donnée dédoublée et donc de provoquer des incohérences dans la base de données, qui ont pour conséquence des erreurs difficiles à corriger.

Par exemple, si vous reprenez les titres de livres à la fois dans la table *Auteurs* et dans la table *Livres*, alors que vous devez corriger une erreur typographique dans un titre, vous devez effectuer une recherche dans les deux tables pour être certain d'effectuer la même modification dans chaque occurrence de ce titre. Voilà pourquoi il vaut mieux conserver les titres de livres à un seul endroit et utiliser plutôt l'ISBN dans les autres, pour y faire référence.

Cependant, dans le processus d'éclatement d'une base de données en plusieurs tables, il importe de ne pas aller trop loin non plus, de ne pas créer plus de tables que nécessaire, car cela conduirait à une conception inefficace et à des ralentissements dans les accès aux données.

Heureusement, E. F. Codd, l'inventeur du modèle relationnel, a analysé le concept de normalisation et l'a réparti dans trois schémas qu'il a nommés *première*, *deuxième* et *troisième formes normales*. Si vous modifiez une base de données pour satisfaire chacune de ces formes dans l'ordre, vous garantirez à votre base de données un équilibre optimal en vue de l'accès le plus rapide et de l'occupation la plus faible en mémoire et sur disque dur.

Aux fins de démonstration du fonctionnement du processus de normalisation, nous allons partir d'une base de données plutôt monstrueuse illustrée au tableau 9-1, qui montre une seule table contenant toutes les informations des noms d'auteurs, de titres de livres et de clients fictifs. Considérez-la comme une première tentative de conception d'une table destinée à suivre les clients qui ont commandé des livres. Il est évident qu'il s'agit là d'une conception inefficace, car les données y figurent en double un peu partout (les doublons sont mis en évidence), mais elle constitue un point de départ.

Tableau 9-1. Conception extrêmement inefficace d'une table de base de données

Auteur 1	Auteur 2	Titre	ISBN	Prix CAD	Nom client	Adresse client	Date achat
Daniel Sklar	Adam Trachtenberg	Recettes de PHP	0596101015	44.99	Annabelle Bourque	1384 Rideau Street, Ottawa, ON K1N 5Y8	2009-03-03
Danny Goodman		HTML dynamique	0596527403	59.99	**Didier Rivière**	**597, Côte des Neiges, Montréal, QC H3S 1Z5**	**2008-12-19**
Hugh E Williams	Daniel Lane	PHP et MySQL	0596005436	44.95	Éric Turcotte	128, rue Hart, Trois-Rivières, QC G9A 4S5	2009-06-22
Daniel Sklar	Adam Trachtenberg	Recettes de PHP	0596101015	44.99	**Didier Rivière**	**597, Côte des Neiges, Montréal, QC H3S 1Z5**	**2008-12-19**
Rasmus Lerdorf	Kevin Tatroe et Peter MacIntyre	Programmation en PHP	0596006815	39.99	Daniel Mercier	24 Bd St-Pierre O, Caraquet, NB E1W 1A4	2009-01-16

Les trois sections suivantes examinent cette conception de base de données et expliquent comment l'améliorer par la suppression de plusieurs doublons de données et l'éclatement de cette seule table en plusieurs tables, contenant chacune des données correspondant à un seul concept.

Première forme normale

Pour qu'une base de données respecte la *première forme normale*, elle doit remplir trois exigences :

- Il ne faut aucune répétition de colonnes contenant le même genre de données.
- Toutes les colonnes ne doivent contenir qu'une seule valeur par colonne.
- Une clé primaire doit exister pour identifier de manière unique chaque ligne de données.

L'examen de ces exigences dans l'ordre permet de reconnaitre immédiatement que les deux colonnes *Auteur 1* et *Auteur 2* forment des répétitions d'un même type de donnée. Donc voilà déjà une première cible de colonne à extraire et à placer dans une table distincte, car les colonnes répétées *Auteur* enfreignent la règle 1.

Ensuite, pour le dernier livre cité, *Programmation en PHP*, trois auteurs apparaissent, que j'ai retranscrits en plaçant Kevin Tatroe et Peter MacIntyre dans la même colonne *Auteur 2*, ce qui enfreint la règle 2, et constitue une raison de plus pour transférer les détails d'*Auteur* dans une table distincte.

En revanche, la règle 3 est satisfaite car la clé primaire a déjà été créée dans l'ISBN.

Le tableau 9-2 montre le résultat de la suppression des colonnes des auteurs du tableau 9-1. Les résultats semblent déjà moins encombrés, même si des doublons apparaissent encore, mis en évidence.

Tableau 9-2. Résultat de la suppression des colonnes d'auteurs du tableau 9-1

Titre	ISBN	Prix CAD	Nom client	Adresse client	Date achat
Recettes de PHP	0596101015	44.99	Annabelle Bourque	1384 Rideau Street, Ottawa, ON K1N 5Y8	2009-03-03
HTML dynamique	0596527403	59.99	**Didier Rivière**	**597, Côte des Neiges, Montréal, QC H3S 1Z5**	**2008-12-19**
PHP et MySQL	0596005436	44.95	Éric Turcotte	128, rue Hart, Trois-Rivières, QC G9A 4S5	2009-06-22
Recettes de PHP	0596101015	44.99	**Didier Rivière**	**597, Côte des Neiges, Montréal, QC H3S 1Z5**	**2008-12-19**
Programmation en PHP	0596006815	39.99	Daniel Mercier	24 Bd St-Pierre O, Caraquet, NB E1W 1A4	2009-01-16

Le tableau 9-3 montre que la nouvelle table *Auteurs* est petite et simple. Elle énumère tous les ISBN des ouvrages associés à un auteur. Si un ouvrage a plus d'un auteur, les auteurs supplémentaires ont droit à leur propre ligne. Au premier abord, vous pourriez vous sentir mal à l'aise avec cette table parce qu'elle ne vous dit apparemment pas quel auteur a écrit quel livre. Mais ne vous inquiétez pas, MySQL peut vous le dire très rapidement. Il suffit de lui indiquer le livre pour lequel vous recherchez des informations et MySQL se sert des ISBN pour interroger la table *Auteurs* en quelques millisecondes.

Tableau 9-3. La nouvelle table Auteurs

ISBN	Auteur
0596101015	Daniel Sklar
0596101015	Adam Trachtenberg
0596527403	Danny Goodman
0596005436	Hugh E Williams
0596005436	Daniel Lane
0596006815	Rasmus Lerdorf
0596006815	Kevin Tatroe
0596006815	Peter MacIntyre

Comme évoqué précédemment, l'ISBN sera la clé primaire de la table *Livres* lorsque nous aborderons la création de cette table. J'insiste sur ce point car il est important de comprendre que l'ISBN *n'est pas* la clé primaire de la table *Auteurs*. Dans la réalité, la table *Auteurs* devrait posséder aussi une clé primaire, de sorte que chaque auteur possède aussi une clé pour l'identifier de manière unique.

Donc, dans la table *Auteurs*, la colonne *ISBN* devra recevoir probablement aussi une clé pour accélérer les recherches mais pas cette clé primaire. En pratique, elle ne peut constituer la clé primaire de cette table parce que ses valeurs ne sont pas uniques de ligne en ligne : le même numéro ISBN apparait plusieurs fois pour deux auteurs ou plus qui ont collaboré à l'écriture de livres.

Comme elle sert à associer des auteurs à des livres d'une autre table, cette colonne porte le nom de *clé étrangère*.

Les clés (autrement dit les index) visent plusieurs buts dans MySQL. La raison fondamentale de la définition d'une clé se situe dans l'accélération des recherches. Le chapitre 8 a montré quelques exemples où des clés intervenaient dans des clauses `WHERE` de requêtes. Mais une clé s'avère aussi utile pour identifier un élément de façon unique. Donc, une clé unique sert souvent de clé primaire dans une table, mais devient une clé étrangère dans une autre table pour associer des enregistrements de cette table à ceux de la première.

Deuxième forme normale

La première forme normale traite des données en double (ou *redondance* de données) entre plusieurs colonnes. La *deuxième forme normale* traite de la redondance parmi plusieurs lignes. Pour respecter la deuxième forme normale, les tables doivent d'abord être en première forme normale. Une fois cette étape franchie, pour atteindre la deuxième forme normale, nous identifions les colonnes dont des données se répètent dans plusieurs lignes, puis nous les extrayons pour les placer dans leurs propres tables.

Examinons à nouveau le tableau 9-2. Remarquez que Didier Rivière a acheté deux livres et que, de ce fait, ses données sont dupliquées. Cela nous indique que les colonnes correspondant aux clients doivent occuper leur propre table. Le tableau 9-4 monte le résultat du retrait des colonnes des clients du tableau 9-2.

Tableau 9-4. La nouvelle table Titres

ISBN	Titre	Prix
0596101015	Recettes de PHP	44.99
0596527403	HTML dynamique	59.99
0596005436	PHP et MySQL	44.95
0596006815	Programmation en PHP	39.99

Tout ce qui reste dans la table *Titres* du tableau 9-4 sont les colonnes *ISBN*, *Titre* et *Prix*, pour quatre livres uniques. Par conséquent, cette table est désormais efficace et autodocumentée, et elle respecte les exigences à la fois des première et deuxième formes normales. Ce faisant, nous avons réussi à réduire les informations à des données étroitement reliées aux titres des ouvrages. Cette table pourrait aussi reprendre les années de publication, les nombres de pages, les nombres de réimpressions et ainsi de suite, puisque ces détails sont étroitement liés. La seule règle à respecter à ce stade consiste à ne

pas permettre la présence de plusieurs valeurs d'un même livre, par exemple pour les rééditions, parce que nous aurions plusieurs fois le titre d'un même ouvrage dans des lignes différentes et nous enfreindrions la deuxième forme normale. La restauration d'une colonne *Auteur*, par exemple, enfreindrait cette normalisation.

Lorsque nous nous penchons sur les colonnes relatives aux clients, extraites dans le tableau 9-5, nous constatons qu'une normalisation est également nécessaire, car les informations de Didier Rivière sont encore en double. Et l'argument subsiste que la règle 2 de la première forme normale (toutes les colonnes ne doivent contenir qu'une seule donnée) n'est pas tout à fait respectée, car les adresses doivent subir un éclatement en colonnes distinctes pour l'*Adresse*, la *Ville*, la *Province* et le code postal *CP*.

Tableau 9-5. Les informations des clients extraites du tableau 9-2

ISBN	Nom client	Adresse client	Date achat
0596101015	Annabelle Bourque	1384 Rideau Street, Ottawa, ON K1N 5Y8	2009-03-03
0596527403	Didier Rivière	597, Côte des Neiges, Montréal, QC H3S 1Z5	2008-12-19
0596005436	Éric Turcotte	128, rue Hart, Trois-Rivières, QC G9A 4S5	2009-06-22
0596101015	Didier Rivière	597, Côte des Neiges, Montréal, QC H3S 1Z5	2008-12-19
0596006815	Daniel Mercier	24 Bd St-Pierre O, Caraquet, NB E1W 1A4	2009-01-16

À partir de ce constat, nous devons encore éclater cette table un peu plus pour garantir de n'entrer les informations qu'une seule fois. Comme l'ISBN n'est pas la clé primaire et ne peut servir pour identifier les clients (pas plus que les auteurs), la table nécessite une nouvelle clé.

Le tableau 9-6 illustre les résultats de la normalisation de la table *Clients* dans ses première et deuxième formes normales. Chaque client reçoit à présent un identifiant unique nommé *NumCli* (numéro client) qui constitue la clé primaire de la table, que nous créerons très probablement à l'aide de l'`AUTO_INCREMENT`. Toutes les portions des adresses des clients ont subi également un éclatement en des colonnes distinctes pour y faciliter la recherche et les modifications.

Tableau 9-6. La nouvelle table Clients

NumCli	Nom	Adresse	Ville	Prov	CP
1	Annabelle Bourque	1384 Rideau Street	Ottawa	ON	K1N 5Y8
2	Didier Rivière	597, Côte des Neiges	Montréal	QC	H3S 1Z5
3	Éric Turcotte	128, rue Hart	Trois-Rivières	QC	G9A 4S5
4	Daniel Mercier	24 Bd St-Pierre O	Caraquet	NB	E1W 1A4

Dans la foulée, pour normaliser la table, nous avons dû retirer du tableau 9-6 les informations sur les achats des clients, parce que sinon, nous aurions conservé plusieurs copies des informations sur les clients pour chaque livre acheté. Pour éviter cela, nous plaçons les données sur les achats dans une nouvelle table dénommée *Achats* (tableau 9-7).

Tableau 9-7. La nouvelle table Achats

NumCli	ISBN	Date
1	0596101015	2009-03-03
2	0596527403	2008-12-19
2	0596101015	2008-12-19
3	0596005436	2009-06-22
4	0596006815	2009-01-16

Cette fois, nous réutilisons la colonne *NumCli* du tableau 9-6 en guise de clé pour associer les tables *Clients* et *Achats*. Comme la colonne *ISBN* est reprise ici, il est également possible de lier cette table aux autres tables *Auteurs* et *Titres*.

La colonne *NumCli* est une clé qui peut s'avérer utile dans la table *Achats*, mais ce ne peut être une clé primaire. Un même client peut acheter plusieurs livres (et même plusieurs copies d'un même livre), donc la colonne *NumCli* n'est pas la clé primaire d'*Achats*. En fait, la table *Achats* ne possède pas de clé primaire et c'est très bien ainsi, car nous ne prévoyons pas de suivre chaque achat unique. Si un client achète deux copies du même livre le même jour, il nous suffira de créer deux lignes avec les mêmes informations. Pour faciliter les recherches, nous pouvons définir tant *NumCli* que l'*ISBN* comme des clés, mais pas comme des clés primaires.

Nous obtenons quatre tables, soit une de plus que les trois que nous avions initialement prévues. Nous sommes parvenus à cette décision selon le processus de normalisation, en suivant méthodiquement les règles des première et deuxième formes normales, qui ont abouti à considérer la nécessité d'une quatrième table dénommée *Achats*.

Nous disposons à ce stade des tables *Auteurs* (tableau 9-3), *Titres* (tableau 9-4), *Clients* (tableau 9-6) et *Achats* (tableau 9-7), et nous pouvons associer chaque table aux autres à l'aide des clés *NumCli* ou *ISBN*.

Par exemple, pour connaitre les livres que Didier Rivière a achetés, vous pouvez le rechercher dans le tableau 9-6 des *Clients*, où vous lisez son *NumCli* égal à 2. Avec ce numéro, allez dans le tableau 9-7 de la table *Achats* et examinez la colonne *ISBN*: il a acheté les titres correspondant aux ISBN 0596527403 et 0596101015 en date du 19 décembre 2008. Cette méthode de recherche présente une certaine lourdeur pour un être humain mais, pour MySQL, il n'y a rien de plus facile.

Pour déterminer les titres correspondant à ces ISBN, retournez au tableau 9-4 de la table des Titres, et vous constatez que les livres achetés sont *HTML dynamique* et *Recettes de PHP*. Pour connaitre les auteurs de ces livres, retournez au tableau 9-3 de la table *Auteurs*: l'ISBN 0596527403, *HTML dynamique*, a été écrit par Danny Goodman et l'ISBN 0596101015, *Recettes de PHP*, a été écrit par David Sklar et Adam Trachtenberg.

Troisième forme normale

Dès que la base de données respecte les première et deuxième formes normales, elle se trouve déjà dans une assez bonne forme et il n'est pas vraiment nécessaire de la modifier davantage. Cependant, si vous souhaitez demeurer strict avec la base de données, vous pouvez vérifier qu'elle respecte la *troisième forme normale*. Celle-ci exige que les données non directement dépendantes de la clé primaire mais dépendantes d'une autre valeur de la même table soient également extraites et placées dans une table distincte, en accord avec la dépendance entre ces tables.

Ainsi, dans le tableau 9-6 de la table *Clients*, nous pouvons considérer que les clés *Province*, *Ville* et *CP* ne sont pas directement liées au numéro de client, la clé primaire, parce que plusieurs personnes peuvent partager ces mêmes détails dans leurs adresses. Cependant, elles sont directement liées les unes aux autres car l'*Adresse* dépend de la *Ville*, et la *Ville* dépend de la *Province*.

Par conséquent, pour que le tableau 9-6 satisfasse à la troisième forme normale, il faudrait l'éclater en tables distinctes, illustrées par les tableaux 9-8 à 9-11 ci-dessous.

Tableau 9-8. Troisième forme normale de la table Clients

NumCli	Nom	Adresse	CP
1	Annabelle Bourque	1384 Rideau Street	K1N 5Y8
2	Didier Rivière	597, Côte des Neiges	H3S 1Z5
3	Éric Turcotte	128, rue Hart	G9A 4S5
4	Daniel Mercier	24 Bd St-Pierre O	E1W 1A4

Tableau 9-9. Troisième forme normale de la table CP des codes postaux

CP	NumVille
K1N 5Y8	1234
H3S 1Z5	5678
G9A 4S5	4321
E1W 1A4	8765

Tableau 9-10. Troisième forme normale de la table des Villes

NumVille	Nom	NumProv
1234	Ottawa	5
5678	Montréal	17
4321	Trois-Rivières	17
8765	Caraquet	21

Tableau 9-11. Troisième forme normale de la table des Provinces

NumProv	Nom	Abréviation
5	Ottawa	ON
17	Montréal	QC
21	Nouveau-Brunswick	NB

C'est bien joli, tout cela, mais comment faut-il utiliser ces tables au lieu de celle du tableau 9-6 ? Si vous suivez la logique de recherche de proche en proche, vous devez partir du code postal, *CP*, dans le tableau 9-8 et le rechercher dans le tableau 9-9 des codes postaux. Celui-ci donne un *NumVille* correspondant. Avec le *NumVille*, allez dans le tableau 9-10 pour connaitre le *Nom* de ville mais aussi le *NumProv*, la clé de la table des *Provinces*. Avec le *NumProv*, allez dans le tableau 9-11, la table des *Provinces*, pour obtenir le *Nom* de la province et son *Abréviation*.

Bien que cette troisième forme normale développée ainsi puisse représenter une complication extrême, elle présente des avantages. Par exemple, revenez au tableau 9-11, qui donne le nom de la province et son abréviation en deux lettres. Elle pourrait contenir des informations sur sa population et d'autres données démographiques, qui peuvent s'avérer utiles.

Le tableau 9-10 pourrait aussi contenir des données démographiques plus locales encore, qui pourraient s'avérer utiles pour vous ou vos clients. L'éclatement de ces données permet d'entretenir plus facilement la base de données s'il faut lui ajouter des colonnes.

La décision quant à l'utilisation de la troisième forme normale peut s'avérer complexe. L'évaluation dépend des données actuelles, mais aussi de prévisions d'évolution dans le temps, au cas où il faudrait ajouter des données. Si vous êtes absolument certain que vous n'aurez jamais besoin que de l'adresse complète des clients, vous pouvez décider d'ignorer cette dernière étape de normalisation.

Dans le cas extrême inverse, si vous développez une base de données pour une vaste organisation telle que les services postaux du pays, imaginez ce qu'il faudrait apporter comme modifications à cette base de données au cas où une ville serait rebaptisée. Dans une table telle que celle illustrée au tableau 9-6, vous seriez obligé d'effectuer une recherche globale avec remplacement dans chaque instance de cette ville. En revanche, avec une base de données normalisée selon la troisième forme normale, vous n'auriez à changer qu'une seule occurrence, comme dans le tableau 9-10, et cette modification apparaitrait partout dans la base de données et ses requêtes.

Dès lors, je vous suggère de toujours vous poser les deux questions suivantes pour décider d'appliquer ou non la troisième forme normale à une table :

- Est-il probable de devoir ajouter par la suite de nouvelles colonnes à cette table ?
- Un des champs de cette table est-il susceptible de recevoir une modification globale, de quelque ordre que ce soit ?

Si vous répondez par l'affirmative à au moins une de ces questions, alors envisagez l'application de cette dernière étape de normalisation.

Cas où la normalisation n'intervient plus

Maintenant que vous avez bien assimilé les règles de la normalisation, j'ai le regret de vous annoncer que, dans certains cas, vous pouvez les envoyer par la fenêtre et les oublier, en particulier pour des sites à très fort trafic. C'est exactement ça : ne normalisez

jamais complètement des tables destinées à des sites qui risquent de provoquer la saturation de MySQL.

La normalisation exige de répartir les données sur plusieurs tables, ce qui oblige à effectuer plusieurs appels à MySQL à chaque requête. Sur un site très achalandé, si vous avez des tables normalisées, les accès à la base de données ralentissent très fortement lorsque vous avez au-delà de plusieurs dizaines d'utilisateurs concurrents, parce qu'ils génèrent des centaines d'accès simultanés à la base de données. J'irai jusqu'à vous conseiller de dénormaliser autant que possible les données les plus fréquemment interrogées.

En pratique, lorsque vous disposez de données dédoublées dans vos tables, vous réduisez de manière flagrante le nombre de requêtes nécessaires en plus pour y accéder, car les données dont vous avez besoin sont déjà présentes là où vous en avez besoin, ce qui évite d'aller les rechercher dans d'autres tables. Il suffit donc d'ajouter une colonne dans une requête et ce champ est immédiatement disponible dans tous les résultats générés.

Bien entendu, le corolaire de cette approche réside dans les inconvénients cités précédemment, comme la nécessité d'espace de stockage supplémentaire et l'obligation d'effectuer des modifications dans toutes les copies de la même donnée lorsqu'une mise à jour s'avère indispensable.

Heureusement, les mises à jour multiples jouissent d'une automatisation. MySQL fournit une fonctionnalité nommée *déclencheur* (*trigger*), qui permet de répercuter automatiquement des modifications dans une base de données en réaction à celles que vous appliquez. (Les déclencheurs échappent toutefois à la portée de ce livre.) Une autre possibilité de propager des données redondantes consiste à construire un programme en PHP qui s'exécute d'une manière régulière et maintient la synchronisation des copies. Le programme lit les modifications dans une table « maitre » et met à jour les autres. Le chapitre suivant montre comment accéder à MySQL à partir de PHP.

Ceci étant dit, tant que vous n'avez pas acquis une expérience suffisante de MySQL, je vous recommande de normaliser complètement vos tables (au moins jusqu'aux première et deuxième formes normales), parce que cela éduque vos réflexes et vous permet de décider en parfaite connaissance de cause. Ce n'est que lorsque vous vous lancez dans le développement réel et que vous constatez que les journaux de MySQL commencent à s'affoler que vous pouvez commencer à envisager une dénormalisation.

Relations

MySQL est considéré comme un système de gestion de bases de données *relationnelles*, parce que ses tables ne stockent pas que des données mais aussi des relations parmi ces données. Trois catégories de relations existent.

Un-à-un

Une *relation un-à-un* ressemble à un mariage (traditionnel, du moins) : chaque élément possède une relation avec un et un seul élément d'un autre type. Cette situation se produit rarement. En effet, un auteur peut écrire plusieurs livres, un livre peut être écrit par plusieurs auteurs et même une adresse peut être partagée par plusieurs clients.

Le meilleur exemple de relation un-à-un de ce chapitre se situe du côté de l'association entre le nom d'une province et son abréviation en deux lettres.

Supposons toutefois qu'il n'y a qu'un client associé à une certaine adresse. Dans un tel cas, la relation Clients-Adresses de la figure 9-1 constitue une relation un-à-un : un seul client habite à chaque adresse et chaque adresse correspond à un seul client.

Tableau 9-8a (Clients)

NumCli	Nom
1	Annabelle Bourque
2	Didier Rivière
3	Éric Turcotte
4	Daniel Mercier

Tableau 9-8b (Adresses)

Adresse	CP
1384 Rideau Street	K1N 5Y8
597, Côte des Neiges	H3S 1Z5
128, rue Hart	G9A 4S5
24 Bd St-Pierre O	E1W 1A4

Figure 9-1. La table Clients, du tableau 9-8, éclatée en deux tables

Généralement, quand deux éléments possèdent une relation un-à-un, il est préférable de les insérer en tant que colonnes d'une même table. Pourtant, deux raisons motivent leur éclatement dans deux tables différentes :

- Pour vous préparer à l'éventualité que la relation change par la suite.
- La table comporte tellement de colonnes que vous pressentez des problèmes de performances ou de maintenance, conjurés par l'éclatement.

Bien entendu, si vous deviez édifier vos propres bases de données dans le monde réel, vous créeriez plutôt des relations un-à-plusieurs entre Clients et Adresses (*une* adresse pour *plusieurs* clients).

Un-à-plusieurs

Une *relation un-à-plusieurs* (ou plusieurs-à-un) s'établit quand une ligne d'une table s'associe à plusieurs lignes d'une autre table. Le tableau 9-8 impliquerait une relation un-à-plusieurs si plusieurs clients partageaient la même adresse. C'est la raison pour laquelle nous devrions l'éclater si c'était le cas.

Ainsi, si vous examinez le tableau 9-8a de la figure 9-1, vous pouvez voir qu'elle partage une relation un-à-plusieurs avec le tableau 9-7, parce que chaque client n'apparait qu'une seule fois dans le tableau 9-8a. En revanche, le tableau 9-7, de la table *Achats*, peut contenir (et contient) plusieurs achats pour certains clients. Par conséquent, *un* client possède une relation avec *plusieurs* achats.

La figure 9-2 présente ces deux tables côte-à-côte, les traits interrompus représentent les associations (ou jointures) entre les lignes de ces tables, qui partent d'une seule ligne de la table de gauche et peuvent relier plusieurs lignes de la table de droite. La relation un-à-plusieurs constitue le schéma préférentiel lors de la description d'une

relation plusieurs-à-un, auquel cas il vous suffit d'échanger les tables de gauche et de droite pour les représenter sous forme d'une relation un-à-plusieurs.

Tableau 9-8a (Clients)

NumCli	Nom
1	Annabelle Bourque
2	Didier Rivière
	(etc...)
3	Éric Turcotte
4	Daniel Mercier

Tableau 9-7 (Achats)

NumCli	ISBN	Date
1	0596101015	2009-03-03
2	0596527403	2008-12-19
2	0596101015	2008-12-19
3	0596005436	2009-06-22
4	0596006815	2009-01-16

Figure 9-2. Illustration de la relation un-à-plusieurs entre deux tables

Plusieurs-à-plusieurs

Dans une *relation plusieurs-à-plusieurs*, plusieurs lignes d'une table s'associent à plusieurs lignes d'une autre table. Pour créer ce type de relation, ajoutez une troisième table intermédiaire contenant les mêmes colonnes de clés que les deux autres tables. Cette troisième table ne contient rien que les clés assurant les liens vers les autres tables.

Le tableau 9-12 représente ce genre de table. Elle est extraite du tableau 9-7, la table *Achats*, mais sans la colonne de date d'achat. Elle contient une copie de l'ISBN de chaque titre vendu et le numéro de client de chaque acheteur.

Tableau 9-12. Table intermédiaire

Client	ISBN
1	0596101015
2	0596527403
2	0596101015
3	0596005436
4	0596006815

Une fois cette table constituée, vous pouvez suivre toutes les informations de la base de données par l'entremise d'une suite de relations. Vous pouvez prendre une adresse en guise de point de départ et trouver les auteurs de tous les livres achetés par le client habitant à telle adresse.

Ainsi, supposons que vous souhaitiez connaitre les achats effectués par des clients du code postal H3S 1Z5. Trouvez ce code postal dans le tableau 9-8b, et vous découvrez que le client numéro 2 a acheté au moins un article de la base de données. À ce stade, utilisez le tableau 9-8a pour obtenir son nom ou suivez le tableau 9-12 intermédiaire pour connaitre les livres achetés.

À partir de là, vous découvrez l'achat de deux titres et vous pouvez retourner au tableau 9-4 pour obtenir les titres et les prix de ces livres, ou au tableau 9-3 pour en déterminer les auteurs.

Si vous considérez que cela revient à combiner plusieurs relations un-à-plusieurs, vous avez parfaitement raison ! Pour l'illustrer, la figure 9-3 rapproche les trois tables et montre la démarche.

Colonnes du tableau 9-8b (Clients)

CP	NumCli
K1N 5Y8	1
H3S 1Z5	2
(etc.)	
G9A 4S5	3
E1W 1A4	4

Tableau 9-12 intermédiaire (Client/ISBN)

NumCli	ISBN
1	0596101015
2	0596101015
2	0596527403
3	0596005436
4	0596006815

Colonnes du tableau 9-4 (Titres)

ISBN	Titre
0596101015	Recettes de PHP
(etc.)	
0596527403	HTML dynamique
0596005436	PHP et MySQL
0596006815	Programmation en PHP

Figure 9-3. Création d'une relation plusieurs-à-plusieurs à l'aide d'une table intermédiaire

Suivez n'importe quel code postal de la table de gauche jusqu'au numéro de client associé. De là, suivez le lien vers la table du milieu, qui joint les tables de gauche et de droite par l'association des numéros de clients et des ISBN. Vous avez tout pour suivre un ISBN et accéder à la table de droite pour connaitre les titres des livres correspondants.

Cette table intermédiaire permet aussi de suivre le chemin inverse et partir des titres de livres pour aboutir aux codes postaux. La table *Titres* indique l'ISBN, qui permet de retrouver dans la table intermédiaire les numéros des clients qui ont acheté tel livre puis, avec ces numéros de clients, vous entrez dans la table *Clients* pour découvrir les codes postaux des clients concernés.

Bases de données et anonymat

Un des aspects intéressants de l'utilisation des relations réside dans la possibilité d'accumuler une pléthore d'informations à propos de certains éléments, par exemple un client, sans savoir réellement qui est ce client. Remarquez que, dans l'exemple précédent, nous sommes parvenus, au départ du code postal des clients, à déterminer les achats des clients, et inversement, sans faire attention aux noms des clients. Les bases de données sont souvent accusées de suivre les gens à la trace, mais elles permettent aussi de respecter la vie privée des gens, tout en obtenant des données statistiques utiles.

Transactions

Pour certaines applications, il relève d'une importance vitale qu'une séquence de requêtes s'exécute dans un ordre déterminé et correct, et que chaque requête élémentaire s'achève avec succès. Pour le démontrer, supposons que vous créiez une séquence de transfert de fonds d'un compte bancaire à un autre. Dans ce cas, il ne faut surtout pas qu'un des évènements suivants se produise :

- Vous ajoutez les fonds au second compte mais, lorsque vous tentez de soustraire ces fonds du premier compte, la mise à jour échoue, de sorte que les deux comptes contiennent les mêmes fonds.
- Vous soustrayez les fonds du premier compte mais, lors de la requête de demande d'ajout de ces fonds au second compte, la mise à jour échoue, de sorte que les fonds s'évanouissent dans la nature.

Ces deux cas extrêmes démontrent que non seulement l'ordre des requêtes importe dans ce type de transaction, mais il est essentiel que toutes les étapes de la transaction s'achèvent avec succès. Mais alors, comment faire pour que tout se passe bien, sachant que lorsqu'une partie de la transaction s'achève, elle ne peut plus être annulée, « dé-faite »? Devez-vous suivre, étape par étape, chaque partie de la transaction et les défaire toutes en vrac si l'une d'elles échoue ? Bien sûr que non, car MySQL fournit de puissantes fonctionnalités de gestion de transactions pour faire face à ce genre d'éventualité.

De plus, les transactions permettent des accès concurrents à une base de données émise par de nombreux utilisateurs et programmes simultanés. MySQL gère tout cela en coulisses et garantit la mise en file d'attente de toutes les transactions afin que les utilisateurs et les programmes attendent leur tour, sans interférer les uns avec les autres.

Moteurs de stockage de transaction

Pour exploiter les possibilités de transactions de MySQL, vous devez faire appel au moteur de stockage *InnoDB* de MySQL. Ce détail est très simple à mettre en place, car il ne représente qu'un paramètre à préciser lors de la création d'une table. Donc, allez-y, créez la table des comptes bancaires de l'exemple 9-1. (Rappelez-vous que vous devez accéder à la console de ligne de commande de MySQL et avoir sélectionné la base de données adéquate pour y créer cette table.)

Exemple 9-1. Création d'une table prête pour les transactions

```
CREATE TABLE comptes (
numero INT, solde FLOAT, PRIMARY KEY(numero)
) ENGINE InnoDB CHARSET utf8;
DESCRIBE comptes;
```

La dernière ligne de cet exemple affiche la structure de la nouvelle table pour vérifier qu'elle a bien été créée. Les résultats de cette commande sont les suivants :

```
+---------+---------+------+-----+---------+--------+
| Field   | Type    | Null | Key | Default | Extra  |
+---------+---------+------+-----+---------+--------+
| numero  | int(11) | NO   | PRI | 0       |        |
| solde   | float   | YES  |     | NULL    |        |
+---------+---------+------+-----+---------+--------+
2 rows in set (0.01 sec)
```

Créons ensuite les deux comptes qui vous permettront de vous entrainer à utiliser les transactions. Entrez les commandes de l'exemple 9-2.

Exemple 9-2. Ajout de comptes à la table

```
INSERT INTO comptes(numero, solde) VALUES(12345, 1025.50);
INSERT INTO comptes(numero, solde) VALUES(67890, 140.00);
SELECT * FROM comptes;
```

La dernière ligne affiche le contenu de la table et confirme l'insertion des deux lignes. Ses résultats sont les suivants :

```
+---------+--------+
| numero  | solde  |
+---------+--------+
|  12345  | 1025.5 |
|  67890  |    140 |
+---------+--------+
2 rows in set (0.00 sec)
```

Livrons-nous à quelques expériences de transactions avec cette table.

Pour commencer : BEGIN

Les transactions en MySQL commencent toujours par une instruction, soit `BEGIN`, soit `START TRANSACTION`. Entrez les commandes de l'exemple 9-3 pour envoyer une transaction à MySQL.

Exemple 9-3. Une transaction MySQL

```
BEGIN;
UPDATE comptes SET solde=solde+25.11 WHERE numero=12345;
COMMIT;
SELECT * FROM comptes;
```

Le résultat de cette transaction s'affiche grâce à la dernière ligne et donne ce qui suit :

```
+---------+---------+
| numero  | solde   |
+---------+---------+
|  12345  | 1050.61 |
|  67890  |     140 |
+---------+---------+
2 rows in set (0.00 sec)
```


Vous constatez que le compte numéro 12345 a vu son solde augmenté de 25,11 et qu'il atteint à présent 1 050,61. Vous découvrez aussi la commande `COMMIT` dans l'exemple 9-3, qui fait l'objet de la section suivante.

Pour valider : COMMIT

Lorsque vous avez entré une suite de requêtes et que vous considérez que la transaction est terminée, entrez la commande `COMMIT` pour valider toutes les modifications apportées à la base de données depuis le début de la transaction. Tant qu'il ne reçoit pas le `COMMIT`, MySQL considère les modifications apportées depuis le début de la transaction comme temporaires. Cette fonctionnalité vous permet de vous raviser et d'annuler toute la transaction, cette fois avec la commande `ROLLBACK`.

Pour tout annuler : ROLLBACK

La commande `ROLLBACK` indique à MySQL d'oublier toutes les requêtes déposées depuis le début d'une transaction et de clôturer cette transaction. Entrez les commandes de l'exemple 9-4 pour voir les effets d'une transaction de transfert de fonds entre les deux comptes.

Exemple 9-4. Une transaction de transfert de fonds

```
BEGIN;
UPDATE comptes SET solde=solde-250 WHERE numero=12345;
UPDATE comptes SET solde=solde+250 WHERE numero=67890;
SELECT * FROM comptes;
```

Lorsque vous entrez ces lignes, la dernière commande affiche le résultat :

```
+---------+--------+
| numero  | solde  |
+---------+--------+
|  12345  | 800.61 |
|  67890  |    390 |
+---------+--------+
2 rows in set (0.00 sec)
```

Le premier compte bancaire perd 250 dans l'opération, tandis que le second compte augmente de 250, ce qui revient à dire que vous avez transféré une valeur de 250 du premier au second compte. Supposons pourtant que quelque chose se soit mal passé et que vous souhaitiez annuler cette transaction. C'est le rôle de la commande `ROLLBACK` de l'exemple 9-5.

Exemple 9-5. Annulation d'une transaction à l'aide de ROLLBACK

```
ROLLBACK;
SELECT * FROM comptes;
```

Lorsque vous comparez les résultats suivants avec ceux des comptes avant et pendant la transaction, vous constatez que les comptes ont retrouvé leurs valeurs antérieures à la transaction, du fait de l'annulation de la totalité de la transaction par la commande `ROLLBACK`:

```
+---------+---------+
| numero  | solde   |
+---------+---------+
|  12345  | 1050.61 |
|  67890  |     140 |
+---------+---------+
2 rows in set (0.00 sec)
```

Utiliser EXPLAIN

MySQL propose un outil extraordinaire pour enquêter sur la manière dont il interprète les requêtes que vous lui donnez à exécuter. Grâce à `EXPLAIN` (qui signifie expliquer), vous obtenez un aperçu d'une requête pour déterminer s'il est possible de l'écrire autrement pour la rendre plus efficace. L'exemple 9-6 montre son utilisation sur la table *comptes* créée précédemment.

Exemple 9-6. Utilisation de la commande EXPLAIN

```
EXPLAIN SELECT * FROM comptes WHERE numero='12345';
```

Les résultats de cette commande prennent un aspect comparable aux suivants:

```
+--+-----------+--------+-----+-------------+-------+-------+-----+----+------+
|id|select_type|table   |type |possible_keys|key    |key_len|ref  |rows|Extra|
+--+-----------+--------+-----+-------------+-------+-------+-----+----+------+
| 1|SIMPLE     |comptes |const|PRIMARY      |PRIMARY|4      |const| 1  |NULL |
+--+-----------+--------+-----+-------------+-------+-------+-----+----+------+
1 row in set (0.00 sec)
```

MySQL vous retourne les informations suivantes:

`select_type`
: Le type de sélection est `SIMPLE`. Si vous aviez joint des tables, cette colonne indiquerait le type de jointure.

`table`
: La table sujette de la requête est la table comptes.

`type`
: Le type de requête est `const`. Les types possibles sont `ALL`, `index`, `range` (plage), `ref` (référence), `eq_ref` (référence équivalente), `const`, `system` et `NULL`.

`possible_keys`
: Une clé primaire, `PRIMARY`, est possible, ce qui signifie que la requête devrait s'avérer rapide.

`key`
: La clé en cours d'utilisation est `PRIMARY`, ce qui est très bien.

`key_len`
: La longueur de clé est de 4. C'est le nombre d'octets de l'index que MySQL utilisera.

`ref`
: La colonne `ref` affiche quelles colonnes ou constantes interviennent dans la clé. Dans ce cas-ci, la clé utilisée est constante.

`rows`
: Le nombre de lignes sur lesquelles la requête impose une recherche, soit `1`. Excellent.

Chaque fois que vous trouvez qu'une requête semble prendre plus de temps qu'il n'en faut, essayez la commande `EXPLAIN` sur celle-ci pour voir si vous pouvez l'optimiser. Vous obtenez la ou les clés utilisées (s'il y en a), leur longueur et ainsi de suite, de sorte que vous pouvez affiner cette requête en fonction de la conception de vos tables.

Comme vous venez de terminer vos expériences sur la table temporaire *comptes*, vous pouvez la supprimer, si vous le souhaitez. Le cas échéant, entrez la commande suivante:

```
DROP TABLE comptes;
```

Sauvegarder et restaurer

Quelles que soient les données que vous stockez dans une base de données, elles ont probablement de l'importance à vos yeux, même si le seul cout qu'elles représentent s'établit en termes de temps nécessaire pour les réinsérer en cas de défaillance du disque dur. Par conséquent, il importe de conserver des sauvegardes pour protéger votre investissement. Par ailleurs, certains cas vous amènent à faire migrer une base de données vers un nouveau serveur. Dans ce genre de situation, constituez d'abord une sauvegarde de la base et testez vos sauvegardes de temps à autre pour garantir qu'elles fonctionnent bien comme prévu.

Bien heureusement, MYSQL facilite les opérations de sauvegarde et de restauration grâce à la commande `mysqldump`.

Utiliser mysqldump

Avec `mysqldump`, vous pouvez créer un *dump*, c'est-à-dire littéralement un «vidage», une copie en vrac, d'une base de données ou d'une collection de bases de données dans un ou plusieurs fichiers qui contiennent toutes les instructions nécessaires pour recréer toutes les tables et les remplir avec des données sauvegardées. L'outil permet aussi de générer des fichiers au format CSV (valeurs séparées par des virgules) et en d'autres formats de texte délimité, ou même en XML. Le seul inconvénient de cet outil est qu'aucun autre utilisateur ne peut écrire dans une table pendant que vous la sauvegardez. Plusieurs moyens permettent d'éviter ce genre de problème, le plus simple étant encore d'arrêter le serveur MySQL avant de lancer `mysqldump` et démarrer de nouveau le serveur après que `mysqldump` a terminé.

Une autre possibilité consiste à verrouiller les tables que vous souhaitez sauvegarder avant de démarrer `mysqldump`. Pour verrouiller des tables en écriture (et n'autoriser que la lecture, puisque nous devons lire les données pour les sauvegarder), à partir de la console de MySQL, entrez la commande suivante :

```
LOCK TABLES nomtable1 READ, nomtable2 READ ...
```

Ensuite, pour déverrouiller les tables, entrez la commande :

```
UNLOCK TABLES;
```

Par défaut, `mysqldump` ne fait qu'afficher les données extraites donc, pour rediriger le contenu de la sauvegarde vers un fichier, utilisez le symbole `>`.

La syntaxe de base de la commande `mysqldump` est :

```
mysqldump -u utilisateur -pmotdepasse basededonnees
```

Cependant, avant de vider le contenu d'une base de données, vous devez vérifier que `mysqldump` se trouve dans votre variable d'environnement `PATH`, sinon vous devez indiquer le chemin complet du fichier. Le tableau 9-13 illustre les emplacements probables du programme en fonction des différents types d'installation et de systèmes d'exploitation, comme indiqué au chapitre 2. Si vous avez installé XAMPP ailleurs, corrigez l'emplacement, qui peut différer légèrement.

Tableau 9-13. Emplacements probables de mysqldump selon différentes installations

Système d'exploitation et programme	Emplacement probable du dossier
Windows XAMPP	C:\xampp\mysql\bin
OS X XAMPP	/Applications/xampp/bin
Linux XAMPP	/opt/lampp/bin

Pour vider le contenu de la base de données *publications* que vous avez créée au chapitre 8 à l'écran, entrez `mysqldump` (ou le chemin complet si nécessaire) et la commande de l'exemple 9-7, dans le terminal ou l'Invite de commandes (et non plus dans la console de MySQL).

Exemple 9-7. Vidage de la base de données publications à l'écran

```
mysqldump -u utilisateur -pmotdepasse publications
```

N'oubliez pas de remplacer *utilisateur* et *motdepasse* par les informations de connexion de votre installation de MySQL. Si aucun mot de passe n'est défini pour l'utilisateur, éludez cette partie de la commande, mais la partie `-u utilisateur` est obligatoire. Surtout sur un serveur de production, il n'est pas recommandé d'utiliser l'utilisateur `root` sans mot de passe. Les résultats de l'envoi de cette commande devraient ressembler à la figure 9-4.

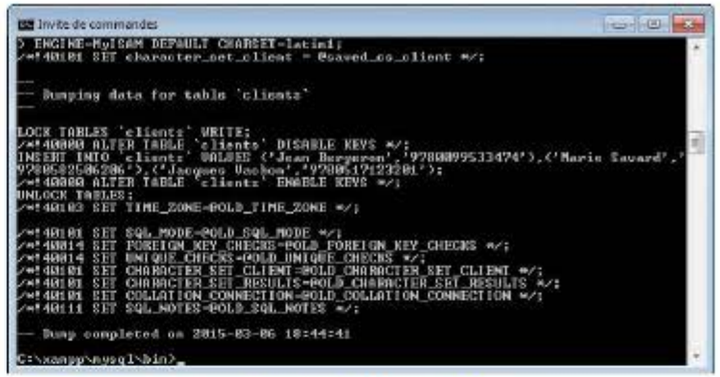


Figure 9-4. Vidage de la base de données publications à l'écran

Créer un fichier de sauvegarde

Dès que vous avez la confirmation que `mysqldump` fonctionne correctement et que vous avez vérifié ses résultats à l'écran, vous pouvez envoyer les données de sauvegarde vers un fichier à l'aide du symbole de redirection `>`. En supposant que vous souhaitiez appeler le fichier de backup *publications.sql*, entrez la commande de l'exemple 9-8, en n'oubliant pas de remplacer *utilisateur* et *motdepasse* par les informations adéquates.

Exemple 9-8. Vidage de la sauvegarde de la base de données publications dans un fichier

```
mysqldump -u utilisateur -pmotdepasse publications > publications.sql
```

La commande de l'exemple 9-8 crée le fichier de sauvegarde dans le dossier courant. Si vous souhaitez l'enregistrer ailleurs, indiquez le chemin avant le nom du fichier. N'oubliez pas de vérifier que vous disposez des permissions nécessaires dans le dossier cible pour y écrire le fichier.

Si vous envoyez le fichier de sauvegarde à l'écran (commande `echo` ou `type` du système d'exploitation) ou dans un éditeur de texte (ou de programme), vous constatez qu'il contient des séquences de commandes SQL telles que les suivantes :

```
DROP TABLE IF EXISTS `classiques`;
CREATE TABLE `classiques` (
  `auteur` varchar(128) DEFAULT NULL,
  `titre` varchar(128) DEFAULT NULL,
  `categorie` varchar(16) DEFAULT NULL,
  `annee` smallint(6) DEFAULT NULL,
  `isbn` char(13) NOT NULL DEFAULT '',
  PRIMARY KEY (`isbn`),
  KEY `auteur` (`auteur`(20)),
  KEY `titre` (`titre`(20)),
```

```
  KEY `categorie` (`categorie`(4)),
  KEY `annee` (`annee`),
  FULLTEXT KEY `auteur_2` (`auteur`,`titre`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Il s'agit là d'un code intelligent utilisable pour restaurer une base de données à partir d'une sauvegarde, même si elle existe déjà, parce qu'il supprime d'abord les tables avant de les recréer, pour éviter des erreurs potentielles opposables par MySQL.

Sauvegarder une seule table

Pour ne sauvegarder qu'une seule table d'une base de données (comme la table *classiques* de la base de données *publications*), verrouillez d'abord la table à partir de la console de commande de MySQL, à l'aide d'une commande du genre de la suivante :

```
LOCK TABLES publications.classiques READ;
```

Ceci garantit que MySQL demeure en fonctionnement à des fins de lecture, mais qu'aucune écriture n'est permise. Ensuite, tout en conservant la console de MySQL ouverte, ouvrez un autre terminal ou Invite de commandes pour accéder à la ligne de commande du système d'exploitation, et entrez

```
mysqldump -u utilisateur -pmotdepasse publications classiques > classiques.sql
```

Vous devez ensuite, aussi vite que possible après la fin de l'exécution de la commande, libérer les verrous sur la table à l'aide de la ligne de commande suivante dans la première console de MySQL, cette fois, pour libérer toutes tables qui ont été verrouillées au cours de la session :

```
UNLOCK TABLES;
```

Sauvegarder toutes les tables

Pour sauvegarder toutes les bases de données MySQL en une seule fois (y compris les bases de données systèmes telles que *mysql*), faites appel à une commande telle que celle de l'exemple 9-9, qui permet de restaurer la totalité d'une installation de base de données de MySQL. Veillez à appliquer les verrouillages requis.

Exemple 9-9. Vidage de toutes les bases de données de MySQL dans un fichier

```
mysqldump -u utilisateur -pmotdepasse --all-databases > toutes_bdd.sql
```

Les fichiers de sauvegarde de base de données contiennent beaucoup plus que quelques lignes de code. Je vous conseille de réserver quelques minutes pour examiner les genres de commandes générées et pour vous familiariser avec elles, dans le but de comprendre le fonctionnement de ces commandes.

Restaurer à partir d'un fichier de sauvegarde

Pour exécuter une restauration à partir d'un fichier de sauvegarde, dans le terminal ou l'Invite de commandes du système d'exploitation, appelez l'exécutable *mysql* et passez-lui le fichier à restaurer après le symbole de redirection <. Ainsi, pour restaurer la totalité des bases de données que vous avez sauvegardées avec l'option `-all-databases`, utilisez une commande telle que celle de l'exemple 9-10.

Exemple 9-10. Restaurer toutes les bases de données

```
mysql -u utilisateur -pmotdepasse < toutes_bdd.sql
```

Ensuite, pour ne restaurer qu'une seule base de données, utilisez l'option `-D` suivie du nom de la base de données, comme dans l'exemple 9-11, qui restaure la base de données *publications* à partir de la sauvegarde réalisée à l'exemple 9-8.

Exemple 9-11. Restaurer la base de données publications

```
mysql -u utilisateur -pmotdepasse -D publications < publications.sql
```

Enfin, pour restaurer une seule table d'une base de données, utilisez une commande du style de l'exemple 9-12, qui ne restaure que la table *classiques* dans la base de données *publications*.

Exemple 9-12. Restaurer la table classiques dans la base de données publications

```
mysql -u utilisateur -pmotdepasse -D publications < classiques.sql
```

Descendre des données en vrac au format CSV

Comme mentionné précédemment, le programme `mysqldump` offre une souplesse remarquable et prend en charge une panoplie de formats de sortie, comme le format CSV. L'exemple 9-13 montre comment vider les données des tables *classiques* et *clients* de la base de données *publications* dans les fichiers respectifs classiques.*txt* et *clients.txt* du dossier *c:\temp*. Sous les systèmes d'exploitation OS X et Linux, adaptez le chemin du dossier cible pour correspondre à un dossier existant.

Exemple 9-13. Vidage de données dans des fichiers au format CSV

```
mysqldump -u utilisateur -pmotdepasse --no-create-info --tab=c:/temp
  --fields-terminated-by=',' publications
```

Cette commande un peu longue est présentée ici sur deux lignes, mais vous devez l'entrer sur une même ligne. Les résultats sont les suivants :

```
Mark Twain (Samuel Langhorne Clemens)','Les aventures de Tom Sawyer',
 'Roman classique','1876','9781598184891
Jane Austen','Orgueil et préjugés','Roman classique','1811','9780582506206
Charles Darwin','De l\'origine des espèces','Scientifique','1856','9780517123201
Charles Dickens','Le Magasin d\'antiquités','Roman classique','1841','9780099533474
William Shakespeare','Roméo et Juliette','Tragédie','1594','9780192814968
```

```
Jean Bergeron','9780099533474
Marie Savard','9780582506206
Jacques Vachon','9780517123201
```

Planifier vos sauvegardes

La règle d'or est d'effectuer des sauvegardes aussi souvent que vous le pouvez, sur un plan pratique. Plus les données revêtent de valeur, plus souvent vous devez les sauvegarder et plus vous devez en effectuer de copies différentes. Si votre base de données reçoit des modifications au moins une fois par jour, pensez alors à les sauvegarder réellement selon un rythme quotidien. Si au contraire les modifications ont lieu à une fréquence plus espacée dans le temps, vous pouvez réaliser des sauvegardes moins fréquentes.

Une autre règle d'or de la sauvegarde indique d'effectuer des copies de sauvegarde dans des emplacements différents. Si vous travaillez sur plusieurs serveurs, il devient simple d'effectuer des sauvegardes croisées et de déposer ces sauvegardes sur les autres serveurs. Lorsque vous gérez des données importantes, voire vitales pour la santé de votre activité (professionnelle), retenez qu'il est préférable aussi d'effectuer une sauvegarde sur un support physique (CD, DVD, bande magnétique et ainsi de suite) et d'en placer une copie dans un endroit protégé de tout risque d'incendie et, pourquoi pas, dans un coffre de sureté à la banque.

Lorsque vous avez bien assimilé le contenu de ce chapitre, vous êtes prêt à utiliser conjointement PHP et MySQL. Le chapitre suivant explique comment faire collaborer ces deux technologies.

Questions

1. Que signifie le mot relation dans le contexte d'une base de données relationnelle ?
2. Quel terme désigne le processus de suppression de données en double et d'optimisation des tables ?
3. Quelles sont les trois règles de la première forme normale ?
4. Comment une table peut-elle satisfaire la deuxième forme normale ?
5. Que placez-vous dans une colonne pour associer deux tables qui contiennent des éléments selon une relation un-à-plusieurs ?
6. Dans une base de données, comment créer une relation plusieurs-à-plusieurs ?
7. Quelles commandes débutent et terminent une transaction MySQL ?
8. Quelle est la fonctionnalité que MySQL fournit pour vérifier les détails de fonctionnement d'une requête ?
9. Quelle commande utilisez-vous pour sauvegarder la base de données *publications* dans un fichier nommé *publications.sql* ?

Retrouvez les réponses du chapitre 9 dans l'annexe A.

CHAPITRE 10

Accéder à MySQL à l'aide de PHP

Si vous avez suivi les chapitres précédents, vous êtes prêt à utiliser MySQL en conjonction avec PHP. Ce chapitre-ci vous explique comment les intégrer à l'aide des fonctions de PHP qui permettent d'accéder à MySQL.

D'emblée, vous allez devoir faire un effort de mémorisation de termes anglais, car toutes les fonctions de PHP d'accès à MySQL sont en anglais. Vous devrez prendre l'habitude des termes comme *user*, *username*, *password* et ainsi de suite, car c'est sous cette forme que vous les retrouverez souvent dans la documentation de PHP.

Interroger une base de données MySQL en PHP

L'intérêt d'utiliser PHP en guise d'interface à MySQL réside dans la mise en forme des résultats de requêtes SQL pour les présenter dans une page web. Tant que vous pouvez vous connecter à MySQL à l'aide d'un nom d'utilisateur et d'un mot de passe, vous pouvez le faire aussi à partir de PHP.

Cependant, au lieu d'utiliser la ligne de commande de MySQL pour entrer des instructions et visualiser les sorties, vous créez des requêtes à passer à MySQL. Lorsque MySQL renvoie ses réponses, elles se présentent sous la forme d'une structure de données reconnaissables par PHP, au lieu des sorties mises en forme que vous avez l'habitude de voir lorsque vous travaillez dans la console de MySQL. D'autres commandes de PHP permettent de récupérer les données et de les mettre en forme pour une page web.

Dans les précédentes éditions de cet ouvrage, ce chapitre présentait l'ancienne extension *mysql* pour accéder à une base de données MySQL, avant d'évoluer vers l'étude des nouvelles extensions *mysqli* dans le chapitre suivant. Mais le temps poursuit sa marche, dit-on, et actuellement ne persistent que de rares anciennes installations qui utilisent encore l'ancien code, donc nous allons aborder directement l'utilisation de la nouvelle extension, qui correspond plus à la norme actuelle.

Le procédé

Le procédé d'utilisation de MySQL avec PHP est le suivant :

1. Connectez-vous à MySQL et sélectionnez la base de données à utiliser.
2. Édifiez une chaine de requête.
3. Exécutez la requête.
4. Récupérez les résultats et présentez-les dans une page web.
5. Répétez les étapes 2 à 4 jusqu'à l'obtention de toutes les données souhaitées.
6. Déconnectez-vous de MySQL.

Nous allons parcourir tour à tour toutes ces étapes, mais le plus important pour l'instant consiste à définir les détails de connexion de manière sécurisée, de sorte que les gens qui fouillent dans votre système n'aient que des soucis s'ils essaient d'accéder à votre base de données.

Créer un fichier d'ouverture de session

La plupart des sites web développés en PHP contiennent de nombreux fichiers de programmes qui accèdent à MySQL, donc qui doivent connaitre les détails de connexion (*login*), c'est-à-dire un nom d'utilisateur (*username*) et un mot de passe (*password*). Il est donc plutôt raisonnable de ne concevoir qu'un seul fichier avec ces informations et de l'inclure selon les nécessités. L'exemple 10-1 illustre un tel fichier, que nous appelons *login.php*.

Exemple 10-1. Le fichier login.php

```
<?php // login.php
  $hn = 'localhost';
  $db = 'publications';
  $un = 'username';
  $pw = 'password';
?>
```

Entrez cet exemple, remplacez `username` et `password` par ceux que vous avez utilisés pour accéder à votre base de données MySQL et enregistrez ce fichier PHP à la racine des documents web définie au chapitre 2. Nous utiliserons bientôt ce fichier.

Le nom d'hôte (*hostname*) `localhost` doit fonctionner, tant que vous utilisez une base de données MySQL sur votre ordinateur local, et la base de données (*database*) *publications* doit également fonctionner si vous avez suivi soigneusement les exemples présentés jusqu'ici.

Les balises `<?php` et `?>` qui entourent le code sont particulièrement importantes dans ce fichier *login.php* de l'exemple 10-1, car elles indiquent que les lignes qu'elles entourent ne doivent être interprétées *que* comme du code PHP. Si vous les omettez, le fichier

s'affichera en clair et révèlera tous vos secrets quand quelqu'un essayera d'accéder directement au fichier à partir de votre site web. Cette même personne ne voit plus qu'une page vide lorsque ces balises sont en place. De plus, ce fichier vient discrètement s'insérer dans vos autres fichiers de code en PHP.

La variable `$hn` correspondant au nom d'hôte indique à PHP quel ordinateur utiliser pour se connecter à une base de données. Elle est obligatoire car vous pouvez vous connecter aux bases de données MySQL présentes sur tout autre ordinateur et, notamment, sur n'importe quel ordinateur accessible par l'internet. Bien entendu, les exemples de ce chapitre supposent que vous travailliez sur un serveur local, placé sur votre ordinateur, donc au lieu de préciser un nom de domaine du Web, tel que *mysql.monserveur.com*, nous utilisons simplement le nom `localhost`, qui correspond à l'adresse IP 127.0.0.1.

La variable `$db` indique que nous utilisons la base de données *publications* créée au chapitre 8. Cependant, si la base de données vous a été fournie par un administrateur, n'oubliez pas de corriger *login.php* pour faire correspondre toutes les informations.

Le fait de conserver les informations de connexion dans un seul fichier présente un autre intérêt : vous pouvez changer le mot de passe de connexion aussi souvent que vous le souhaitez et vous ne devez changer cette information que dans un seul fichier, quel que soit le nombre de fichiers qui y accèdent.

Se connecter à une base de données MySQL

Le fichier *login.php* créé et enregistré, vous pouvez donc l'inclure dans n'importe quel autre fichier PHP qui requiert un accès à la base de données à l'aide de l'instruction `require_once`. Nous avons vu que celle-ci est préférable par rapport à une instruction `include`, car elle produit une erreur fatale si le fichier requis n'existe pas ou est introuvable. Et vous pouvez me croire, le fait de ne pas trouver le fichier avec les informations de connexion à la base de données est bien une erreur fatale!

Contrairement à `require`, `require_once` présente l'avantage que le fichier n'est lu qu'une seule fois et uniquement s'il n'a pas été déjà inclus auparavant, ce qui évite des accès inutiles au disque. L'exemple 10-2 illustre le code à utiliser.

Exemple 10-2. Connexion au serveur MySQL à l'aide de mysqli

```
<?php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);
?>
```

Ce code crée un nouvel objet de connexion `$conn` par l'appel d'une nouvelle instance de la classe `mysqli`, et lui passe en arguments toutes les valeurs récupérées à partir

du fichier *login.php*. La vérification d'erreur est réalisée à l'aide de la référence à la propriété `$conn->connect_error`.

L'opérateur `->` indique que l'élément de droite est une propriété ou une méthode de l'objet de gauche. Dans ce cas-ci, si `connect_error` contient une autre valeur que `FALSE` ou `NULL`, alors une erreur s'est produite et nous appelons la fonction `die` (littéralement « mourir »), qui affiche le contenu de cette propriété, avec les détails de l'erreur de connexion, puis arrête le programme.

L'objet `$conn` est utilisé dans les exemples suivants pour accéder à la base de données MySQL.

La fonction `die` s'avère très pratique en cours de développement de code PHP mais, sur un serveur de production, vous devez bien entendu afficher des messages d'erreur plus conviviaux. Dans ce dernier cas, vous n'arrêtez pas brutalement le programme mais vous rédigez un message destiné à l'affichage au moment où le programme se termine normalement, dans le genre :

```
function mysql_fatal_error($msg)
{
    $msg2 = mysql_error();
    echo <<< _END
Nous sommes désolés mais il n'est pas possible d'effectuer
la tâche demandée. L'erreur rencontrée est la suivante :

    <p>$msg: $msg2</p>

Cliquez sur le bouton Précédent de votre navigateur
et réessayez. Si vous rencontrez encore des problèmes,
communiquez avec <a href="mailto:admin@serveur.com">
notre administrateur</a>. Merci.
_END;
}
```

Rédiger et exécuter une requête

L'envoi d'une requête à MySQL à partir de PHP se résume à l'envoyer par l'entremise de la méthode `query` d'un objet de connexion. L'exemple 10-3 en illustre l'utilisation.

Exemple 10-3. Interrogation d'une base de données à l'aide de mysqli

```
<?php
  $query  = "SELECT * FROM classiques";
  $result = $conn->query($query);
  if (!$result) die($conn->error);
?>
```

La variable `$query` reçoit une chaine contenant la requête à envoyer. Cette variable est passée en argument à la méthode `query` de l'objet `$conn`, qui renvoie les résultats dans

l'objet `$result`. Si `$result` contient `FALSE`, alors un problème s'est produit, dont la propriété `error` de l'objet de connexion contient les détails. Dans ce cas, l'appel à la fonction `die` affiche l'erreur.

Si tout s'est bien passé, toutes les données renvoyées par MySQL sont rangées dans un format de lecture aisée dans l'objet `$result`.

Avant d'aller plus loin dans les détails du code, voici une petite astuce bien utile, dès lors que vous voulez afficher correctement des caractères accentués dans une page web, extraits d'une base de données MySQL. Du fait des divergences de systèmes de codage des caractères entre MySQL (du moins par défaut) et HTML, les caractères accentués des résultats de la table *Titres* apparaissent bizarrement dans la page web, sous forme d'un point d'interrogation blanc sur fond de losange noir (). Pour éviter cela, avant d'envoyer une requête `SELECT` pour récupérer des données, placez d'abord une requête `SET NAMES utf8` pour forcer en UTF8 le codage de caractères récupérés ensuite. C'est le rôle des trois lignes de code suivantes, à placer avant toute autre requête mais après l'ouverture de la connexion à la base de données :

```
<?php
  $query  = "SET NAMES utf8";    // Affichage des caractères accentués
  $result = $conn->query($query);
  if (!$result) die($conn->error);
?>
```

Récupérer un résultat

Dès que vous disposez de l'objet retourné dans `$result`, vous pouvez en extraire les données que vous souhaitez, un élément à la fois, à l'aide de la méthode `fetch_assoc` de l'objet. L'exemple 10-4 regroupe et étend les exemples précédents en un programme que vous pouvez entrer et exécuter vous-même (comme indiqué à la figure 10-1). Je vous suggère de nommer ce fichier *query.php*. Mais ce fichier est également disponible dans l'archive des fichiers sources sur le site d'accompagnement du livre.

Exemple 10-4. Récupération des résultats, une cellule à la fois

```
<?php // query.php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);

  $query  = "SET NAMES utf8"; // Affichage des caractères accentués
  $result = $conn->query($query);
  if (!$result) die($conn->error);

  $query  = "SELECT * FROM classiques";
  $result = $conn->query($query);
  if (!$result) die($conn->error);
```

```
  $rows = $result->num_rows;  // Nombre de lignes de données

  for ($j = 0 ; $j < $rows ; ++$j)
  {
    $result->data_seek($j);
    echo 'Auteur: '     . $result->fetch_assoc()['auteur']    . '<br>';
    $result->data_seek($j);
    echo 'Titre : '     . $result->fetch_assoc()['titre']     . '<br>';
    $result->data_seek($j);
    echo 'Catégorie : ' . $result->fetch_assoc()['categorie'] . '<br>';
    $result->data_seek($j);
    echo 'Année : '     . $result->fetch_assoc()['annee']     . '<br>';
    $result->data_seek($j);
    echo 'ISBN: '       . $result->fetch_assoc()['isbn']      . '<br><br>';
  }

  $result->close();
  $conn->close();
?>
```

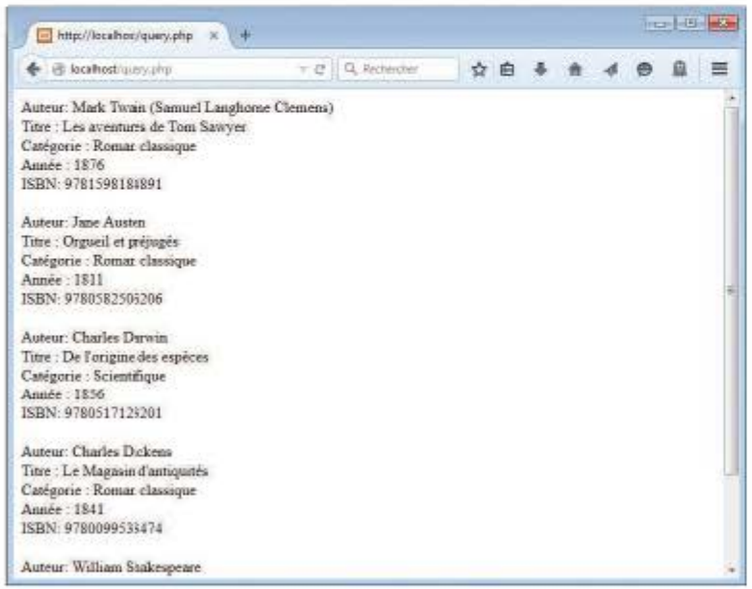


Figure 10-1. Résultats de sortie du programme query.php de l'exemple 10-4

Ici, pour retrouver la ligne adéquate lors de chaque passage par la boucle, nous appelons la méthode `data_seek` (lire des données) de l'objet `$result` avant de récupérer chaque donnée. Nous appelons ensuite la méthode `fetch_assoc` pour récupérer la valeur stockée dans chaque cellule et afficher le résultat à l'aide d'instructions `echo`.

Vous conviendrez que cette procédure de récupération des données est un peu lourde. Il doit donc y avoir une méthode plus efficace et plus claire pour obtenir le même résultat. Et, de fait, une meilleure méthode existe, qui extrait toute une ligne en même temps.

Au chapitre 9, j'ai insisté sur les premières, deuxième et troisième formes normales et vous avez sans doute remarqué que la table *classiques* ne les respecte pas, car les informations spécifiques aux auteurs et aux livres figurent dans la même table. Tout cela provient du fait que nous avons créé cette table avant de découvrir la normalisation. Cependant, pour les besoins de l'illustration de l'accès à MySQL à partir de PHP, nous réutilisons cette table pour éviter les tracas de créer un nouveau jeu de données.

Récupérer une ligne

Pour récupérer (*fetch*) toute une ligne (*row*) à la fois, remplacez la boucle `for` de l'exemple 10-4 par celle mise en évidence dans l'exemple 10-5, et vous obtenez exactement le même résultat à l'affichage qu'à la figure 10-1. Enregistrez ce fichier revisité sous le nom *fetchrow.php*.

Exemple 10-5. Récupération des résultats à raison d'une ligne à la fois

```
<?php //fetchrow.php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);

  $query  = "SET NAMES utf8";
  $result = $conn->query($query);
  if (!$result) die($conn->error);

  $query  = "SELECT * FROM classiques";
  $result = $conn->query($query);
  if (!$result) die($conn->error);

  $rows = $result->num_rows;

  for ($j = 0 ; $j < $rows ; ++$j)
  {
    $result->data_seek($j);
    $row = $result->fetch_array(MYSQLI_ASSOC);

    echo 'Auteur : '    . $row['auteur']    . '<br>';
    echo 'Titre : '     . $row['titre']     . '<br>';
    echo 'Catégorie : ' . $row['categorie'] . '<br>';
    echo 'Année: '      . $row['annee']     . '<br>';
    echo 'ISBN: '       . $row['isbn']      . '<br><br>';
  }

  $result->close();
  $conn->close();
?>
```

Cette version du code n'effectue qu'un cinquième des interrogations de l'objet `$result` par rapport au précédent exemple, et une seule lecture de l'objet est réalisée dans chaque itération de la boucle, car la lecture porte sur chaque ligne en totalité par l'entremise de la méthode `fetch_array` (récupère tableau). Celle-ci renvoie une seule ligne de données sous forme d'un tableau, affecté ensuite au tableau `$row`.

La méthode `fetch_array` peut renvoyer trois types de tableaux, selon la valeur qui lui est passée :

`MYSQLI_NUM`
: Un tableau numérique. Chaque colonne apparait dans le tableau dans l'ordre où vous l'avez placée au moment de la création (ou de la modification) de la table. Dans notre cas, l'élément d'indice 0 du tableau contient la colonne *Auteur*, l'élément d'indice 1 contient le *Titre* et ainsi de suite.

`MYSQLI_ASSOC`
: Un tableau associatif. Chaque clé correspond au nom de la colonne. Comme les données sont référencées par leur nom de colonne (au lieu d'un indice numérique), utilisez autant que possible cette option car elle facilite le débogage de l'application et aide les autres développeurs à mieux gérer votre code, sans interroger la structure des tables.

`MYSQLI_BOTH`
: Un tableau associatif et numérique.

Les tableaux associatifs s'avèrent souvent plus pratiques que les numériques, car vous pouvez faire référence à chaque colonne par le nom du champ, par exemple `$row['auteur']`, au lieu de vous imposer de retenir l'indice numérique de la colonne en fonction de son ordre de création. C'est la raison pour laquelle ce script utilise un tableau associatif, déterminé par le passage de `MYSQLI_ASSOC`.

Fermer une connexion

PHP finit par libérer la mémoire qu'il alloue aux objets lorsque l'exécution d'un script s'achève. En conséquence, dans le cas de courts scripts, vous ne devez généralement pas vous préoccuper de la libération de la mémoire. En revanche, lorsque vous allouez un grand nombre d'objets de résultats ou récupérez de grandes quantités de données, il demeure de bon ton de libérer la mémoire dont vous n'avez plus besoin pour éviter des problèmes ultérieurs dans votre script.

Cette démarche revêt d'autant plus d'importance dans le cas de pages qui attirent un gros trafic, car la quantité de mémoire consommée dans une session peut croitre très rapidement. Par conséquent, soyez attentif aux appels des méthodes `close` des objets `$result` et `$conn` dans les deux scripts précédents, dès que ces objets ne sont plus utiles, comme suit :

```
$result->close();
$conn->close();
```


L'idéal serait de clôturer tout objet de résultat dès que vous en avez fini avec lui, puis de clôturer l'objet de connexion lorsque le script ne doit plus y faire appel pour accéder à MySQL. Une bonne pratique de programmation consiste à restituer les ressources au système aussi vite que possible, pour conserver à MySQL un fonctionnement optimal et éviter tout doute quant au fait que PHP ait libéré à temps la mémoire inutilisée la fois suivante où vous en aurez besoin.

Un exemple pratique

Rédigeons notre premier exemple d'insertion et de suppression de données d'une table MySQL à l'aide de PHP. Je vous conseille de taper l'exemple 10-6 et de l'enregistrer dans votre dossier racine de documents web sous le nom *sqltest.php*. La figure 10-2 montre les résultats de l'exécution de ce script.

Exemple 10-6. Insertion et suppression de données à l'aide de sqltest.php

```
<?php // sqltest.php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);

  $query  = "SET NAMES utf8"; // Force l'affichage en utf-8
  $result = $conn->query($query);
  if (!$result) die($conn->error);

  if (isset($_POST['supprimer']) && isset($_POST['isbn']))
  {
    $isbn   = get_post($conn, 'isbn');
    $query  = "DELETE FROM classiques WHERE isbn='$isbn'";
    $result = $conn->query($query);
        if (!$result) echo "Échec de la suppression : $query<br>" .
      $conn->error . "<br><br>";
  }

  if (isset($_POST['auteur'])    &&
      isset($_POST['titre'])     &&
      isset($_POST['categorie']) &&
      isset($_POST['annee'])     &&
      isset($_POST['isbn']))
  {
    $auteur    = get_post($conn, 'auteur');
    $titre     = get_post($conn, 'titre');
    $categorie = get_post($conn, 'categorie');
    $annee     = get_post($conn, 'annee');
    $isbn      = get_post($conn, 'isbn');
    $query     = "INSERT INTO classiques VALUES" .
      "('$auteur', '$titre', '$categorie', '$annee', '$isbn')";
    $result    = $conn->query($query);
```

```
        if (!$result) echo "Échec de l'insertion : $query<br>" .
      $conn->error . "<br><br>";
  }

  echo <<<_END
  <form action="sqltest.php" method="post"><pre>
    Auteur <input type="text" name="auteur">
     Titre <input type="text" name="titre">
 Catégorie <input type="text" name="categorie">
     Année <input type="text" name="annee">
      ISBN <input type="text" name="isbn">
           <input type="submit" value="AJOUTER FICHE">
  </pre></form>
_END;

  $query  = "SELECT * FROM classiques";
  $result = $conn->query($query);
  if (!$result) die ("Échec de l'accès à la base de données : " . $conn->error);

  $rows = $result->num_rows;

  for ($j = 0 ; $j < $rows ; ++$j)
  {
    $result->data_seek($j);
    $row = $result->fetch_array(MYSQLI_NUM);

    echo <<<_END
  <pre>
    Auteur $row[0]
     Titre $row[1]
 Catégorie $row[2]
     Année $row[3]
      ISBN $row[4]
  </pre>
  <form action="sqltest.php" method="post">
  <input type="hidden" name="supprimer" value="yes">
  <input type="hidden" name="isbn" value="$row[4]">
  <input type="submit" value="SUPPRIMER FICHE"></form>
_END;
  }

  $result->close();
  $conn->close();

  function get_post($conn, $var)
  {
    return $conn->real_escape_string($_POST[$var]);
  }
?>
```

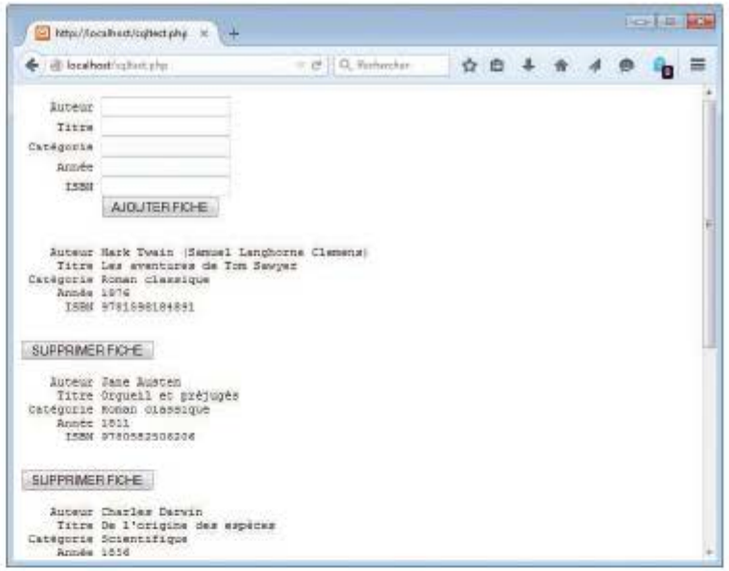



Figure 10-2. La sortie obtenue à partir de l'exemple 10-6, sqltest.php

Avec plus de 80 lignes de code, ce programme peut sembler effrayant mais rassurez-vous, nous avons déjà vu une bonne partie des notions qu'il renferme à l'exemple 10-5 et ce que fait réellement ce code est assez simple.

Il vérifie avant tout la présence d'entrées effectuées puis insère les nouvelles données dans la table *classiques* de la base de données publication, ou il supprime une ligne de ces données, selon les entrées fournies dans le formulaire. Qu'il y ait une entrée ou non, le programme affiche ensuite toutes les lignes de la table. Voyons en détail comment cela fonctionne.

La première section du nouveau code débute par l'utilisation de la fonction `isset` (est définie ?) pour évaluer si des valeurs pour tous les champs ont été fournies au programme (par le biais de `$_POST`). À la confirmation, chaque ligne de l'instruction appelle la fonction `get_post`, qui figure à la fin du programme. Cette fonction remplit un rôle succinct mais très important : récupérer une entrée du navigateur.

Le tableau $_POST

Le chapitre 6 (page 134) évoquait que le navigateur envoie les entrées de l'utilisateur dans un formulaire par l'entremise d'une requête, soit Get, soit Post. La requête Post est généralement préférable (parce qu'elle évite de faire apparaitre des données sans intérêt dans la barre d'adresse du navigateur), donc nous l'utilisons ici. Le serveur web « emballe » toutes les entrées de l'utilisateur, même si le formulaire est rempli de centaines de champs, et les place dans un tableau nommé `$_POST`.

`$_POST` est un tableau associatif, comme ceux que nous avons vus au chapitre 6. Selon que le formulaire est défini pour utiliser la méthode Post ou Get, le tableau associatif correspondant `$_POST` ou `$_GET` est rempli avec les données du formulaire. Ils se lisent exactement de la même manière du côté de PHP.

Chaque champ correspond à un élément du tableau, nommé en fonction de ce champ. Donc, si le formulaire contient un champ nommé `isbn`, le tableau contient un élément dont la clé s'appelle aussi `isbn`. Le programme PHP lit la valeur du champ de formulaire par référence à `$_POST['isbn']` ou `$_POST["isbn"]` car les apostrophes et les guillemets ont le même effet, dans ce cas-ci.

Si la syntaxe de `$_POST` vous semble encore compliquée, retenez que vous pouvez simplement suivre la convention illustrée dans l'exemple 10-6, copier l'entrée de l'utilisateur dans d'autres variables et oublier ensuite tout ce qui concerne `$_POST`. Il est assez fréquent dans les programmes PHP de récupérer les champs de `$_POST` en début de programme et d'ignorer ensuite ce dernier.

Le fait d'écrire dans le tableau `$_POST` à partir d'un programme PHP n'a pas de sens en soi. Son seul but consiste à transmettre des informations du navigateur vers le programme, donc il vaut mieux copier les données dans vos propres variables pour les modifier ensuite.

Pour en revenir à la fonction `get_post`, celle-ci passe chaque élément récupéré par la moulinette de la méthode `real_escape_string` de l'objet de connexion, afin d'éliminer tout caractère qu'un pirate risquerait d'avoir inséré pour s'infiltrer dans la base de données et essayer de l'altérer. Elle agit comme suit :

```
function get_post($conn, $var)
{
  return $conn->real_escape_string($_POST[$var]);
}
```

Supprimer un enregistrement

Avant de vérifier si de nouvelles données ont été publiées, le programme vérifie si la variable `$_POST['supprimer']` contient une valeur. Si c'est le cas, l'utilisateur a cliqué sur le bouton SUPPRIMER FICHE pour effacer un enregistrement. Conjointement, la valeur de `$isbn` est publiée et est disponible dans `$_POST`.

Pour rappel, l'ISBN identifie tout enregistrement de façon unique. Le formulaire HTML ajoute l'ISBN à la chaine de requête `DELETE FROM` créée dans la variable `$query`, puis la passe à la méthode `query` de l'objet `$conn` pour exécution par MySQL.

Si `$_POST['supprimer']` ne contient aucune valeur et qu'aucun enregistrement ne doit être supprimé, alors le programme vérifie le contenu de `$_POST['auteur']` et des autres valeurs publiées. Si elles ont toutes une valeur, alors la chaine de requête `$query` reçoit une commande `INSERT INTO`, suivie des cinq valeurs à insérer. Cette chaine passe à la méthode `query` qui renvoie `TRUE` ou `FALSE` à son achèvement. Si elle renvoie `FALSE`, alors le message contenu dans la propriété `error` de l'objet `$conn` s'affiche, comme suit :

```
if (!$result) echo "Échec de l'insertion : $query<br>" .
  $conn->error . "<br><br>";
```

Afficher le formulaire

L'étape suivante consiste à afficher le petit formulaire présent tout en haut de la figure 10-2. Rappelez-vous la structure `echo <<<_END..._END` que nous avons vue dans des chapitres précédents, qui sort dans le navigateur la totalité de ce qui figure entre les balises `_END`.

À la place de la commande `echo`, le programme aurait pu sortir de PHP avec la balise fermante `?>`, insérer le texte en HTML, puis rentrer dans PHP à l'aide de la balise ouvrante `<?php`. Le choix du style dépend des préférences du programmeur, mais je recommande de toujours demeurer dans le code PHP pour deux raisons :

- C'est une question de clarté au débogage (et à la lecture par d'autres développeurs) que de considérer que tout ce qui figure dans un fichier d'extension *.php* est du code PHP et rien d'autre. Par conséquent, il n'est plus nécessaire de surveiller la présence éventuelle d'injections de code HTML pur.
- En PHP, lorsque vous souhaitez inclure une variable PHP dans du HTML, il suffit de la citer, tandis que si vous essayez d'y accéder dans du HTML pur, vous êtes obligé de rentrer temporairement dans PHP, d'accéder à la variable, puis de retourner en HTML, et ainsi de suite.

La section du formulaire (*form*) HTML définit d'abord l'action liée au formulaire comme étant *sqltest.php*. Cela signifie qu'au moment de la soumission (*submit*) du formulaire, le contenu des champs du formulaire est envoyé au fichier *sqltest.php*, c'est-à-dire le programme lui-même. La déclaration du formulaire règle aussi à Post la méthode de transfert des variables au programme, au lieu d'une requête Get. Pour rappel, la méthode de transfert Get injecte les variables et leurs valeurs dans la barre d'adresse du navigateur, à la suite de l'URL de la page, ce qui manque d'élégance. De plus, cette approche laisse les valeurs à vue et à disposition d'un pirate éventuel, qui serait tenté de les modifier et d'injecter ce qu'il faut pour pirater votre serveur. Par conséquent, chaque fois que c'est possible, utilisez des soumissions Post, qui offrent l'avantage de masquer à toute vue les données publiées.

Les champs du formulaire affichés, le HTML fait aussi apparaître un bouton de soumission intitulé AJOUTER FICHE et clôture le formulaire. Remarquez la présence des balises `<pre>` et `</pre>`, qui permettent de sélectionner une police à espacement fixe pour obtenir un bel alignement des étiquettes et des champs. Entre ces balises, les retours à la ligne et les espaces sont également respectés et affichés tels quels.

Interroger la base de données

Ensuite, le code retourne dans le territoire connu de l'exemple 10-5, avec une requête MySQL de sélection pour obtenir tous les enregistrements de la table classiques, comme suit :

```
$query  = "SELECT * FROM classiques";
$result = $conn->query($query);
```

Outre la gestion de l'erreur éventuelle de lecture, vient ensuite l'affectation à la variable `$rows` du nombre de lignes obtenues :

```
$rows = $result->num_rows;
```

La valeur de `$rows` permet à une boucle `for` d'afficher le contenu de chaque ligne de données. Dans chaque itération de la boucle, l'appel de la méthode `data_seek` de l'objet `$result` cherche les données qui nous intéressent, comme suit :

```
$result->data_seek($j);
```

L'instruction suivante assure le remplissage du tableau `$rows` par la méthode `fetch_array` de `$result`, qui reçoit la constante `MYSQLI_NUM`, correspondant au renvoi d'un tableau à indices numériques (au lieu d'un tableau associatif) :

```
$row = $result->fetch_array(MYSQLI_NUM);
```

Le tableau `$row` contient à présent les données d'une ligne de la table. L'affichage de ces données vient s'insérer dans une nouvelle instruction `echo` sur plusieurs lignes, avec à nouveau les balises `<pre>` et `</pre>` pour aligner joliment les étiquettes et les données de l'enregistrement.

Juste après l'affichage de chaque enregistrement, un deuxième formulaire permet de publier des données, de nouveau avec la méthode Post et l'appel au programme *sqltest.php*, mais cette fois avec deux champs masqués (*hidden*) : `supprimer` et `isbn`. Le champ `supprimer` est réglé à `yes` (peu importe la valeur, pourvu qu'il y en ait une), tandis qu'`isbn` reçoit la valeur contenue dans `$row[4]`, qui correspond à l'ISBN de cet enregistrement. C'est là que se situe probablement toute la subtilité de ce programme.

Le bouton de soumission suit immédiatement, avec son intitulé `SUPPRIMER FICHE`, juste avant la clôture du formulaire. L'accolade finale clôture le corps de la boucle `for`. La boucle effectue autant d'itérations qu'il y a d'enregistrements à afficher. Enfin, les méthodes `close` des objets `$result` et `$conn` permettent de restituer les ressources à PHP :

```
$result->close();
$conn->close();
```


Tout à la fin du programme figure la définition de la fonction `get_post`, que nous avons déjà examinée. Voilà, le premier programme de manipulation d'une base de données MySQL en PHP est terminé. Voyons comment il se comporte.

Lorsque vous avez tapé le programme (et corrigé les erreurs de frappe), lancez-le dans la barre d'adresse de votre navigateur et essayez d'entrer les données suivantes dans les différents champs du premier formulaire, pour ajouter un nouvel enregistrement à la base de données correspondant au livre *Moby Dick* :

```
Herman Melville
Moby Dick
Roman
1851
9780199535729
```

Exécuter le programme

Lorsque vous avez entré ces données et soumis le formulaire à l'aide du bouton AJOUTER FICHE, faites défiler la fenêtre jusqu'au bas de la page web pour constater l'ajout. La figure 10-3 montre les résultats obtenus.

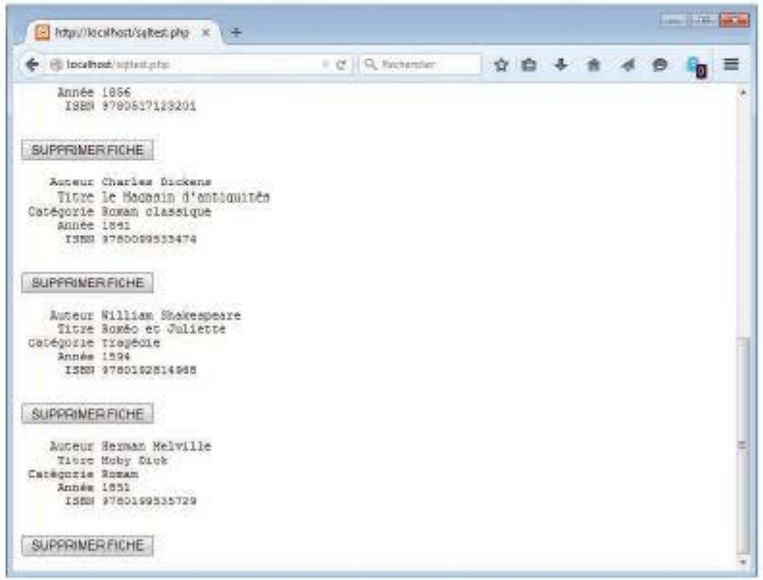


Figure 10-3. Les résultats de l'ajout de Moby Dick dans la base de données

Pour vérifier que la suppression fonctionne aussi, créez un enregistrement factice.

N'entrez que le chiffre 1 dans chacun des cinq champs du premier formulaire, cliquez sur AJOUTER FICHE. Faites défiler la fenêtre jusqu'au bas de la page pour voir la nouvelle entrée et son bouton. Comme cet enregistrement n'est d'aucune utilité, cliquez sur son bouton SUPPRIMER FICHE pour l'éliminer. Faites de nouveau défiler la fenêtre pour vérifier que l'enregistrement factice a disparu de la page.

En supposant que tout fonctionne comme prévu, vous êtes à présent en mesure d'ajouter et de supprimer des enregistrements à volonté. Essayez le programme quelques fois mais laissez les enregistrements principaux en place, y compris celui de Moby Dick, car nous en aurons besoin par la suite. Si vous essayez d'entrer deux fois l'enregistrement avec uniquement des 1, vous verrez que vous recevrez un message d'erreur la deuxième fois, indiquant qu'il existe déjà un ISBN de valeur 1.

MySQL en pratique

Vous allez maintenant découvrir quelques techniques bien pratiques utilisables en PHP pour accéder à la base de données MySQL, afin de vous livrer à des tâches qui concernent les tables elles-mêmes et leur structure, avec la création, la suppression de tables, mais également l'insertion, la mise à jour et la suppression de données, puis la protection de la base de données et du site web contre les utilisateurs malveillants. Les exemples qui suivent reposent sur l'hypothèse que vous disposez déjà du programme *login.php*, créé au début de ce chapitre.

Créer une table

Supposons que vous êtes employé d'un parc animalier et que vous avez besoin d'une base de données pour conserver les informations sur tous les types de félins qu'il héberge. On vous indique qu'il existe neuf *familles* de félins : Lion, Tigre, Jaguar, Léopard, Puma, Panthère, Lynx, Caracal et Domestique. Vous avez donc besoin d'une colonne pour retenir cette information. Ensuite, chaque félin porte un *nom*, donc voici une autre colonne, et vous devez suivre leur *âge*, une autre colonne. Par la suite, vous aurez sans doute besoin d'autres colonnes, par exemple pour retenir les exigences diététiques, suivre les vaccins et autres détails, mais pour l'heure, les trois premières colonnes suffisent pour débuter. Un identifiant unique est nécessaire pour chaque animal donc vous décidez de créer une colonne pour contenir l'identifiant, *id*. En outre, évitez les accents dans les noms de tables et de champs.

L'exemple 10-7 montre le genre de code utilisable pour créer une table MySQL destinée à contenir ces données, avec la principale affectation de requête en surbrillance.

Exemple 10-7. Création de la table dénommée felins

```
<?php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);

  $query = "CREATE TABLE felins (
    id SMALLINT NOT NULL AUTO_INCREMENT,
    famille VARCHAR(32) NOT NULL,
    nom VARCHAR(32) NOT NULL,
    age TINYINT NOT NULL,
    PRIMARY KEY (id)
  ) CHARACTER SET utf8";

  $result = $conn->query($query);
  if (!$result) die ("Échec d'accès à la base de données : " . $conn->error);
?>
```

Cet exemple montre que la requête MySQL ressemble beaucoup à celle que vous entreriez dans la console de MySQL. La principale différence réside dans l'absence de point-virgule final, car il n'est pas nécessaire pour accéder à MySQL à partir de PHP.

Remarquez la présence de la ligne suivante, que nous n'avons pas encore vue jusqu'ici :

```
    ) CHARACTER SET utf8";
```

Elle indique à MySQL d'utiliser directement le codage des caractères en utf8. Cette ligne évite les soucis d'interprétation des caractères que nous avons contournés précédemment (note de la page 236) par l'ajout d'une requête juste après l'ouverture de connexion à la base de données.

Comme tous ces scripts contiennent des caractères accentués, enregistrez également les fichiers *.php* en utf8 pour ne pas rencontrer de problèmes d'interprétation de ces caractères. Les éditeurs de programmes permettent ce genre de conversion.

Décrire une table

Lorsque vous ne disposez pas tout de suite de la console MySQL et que vous n'êtes pas connecté à la base de données, voici un extrait de code qui permet de vérifier dans le navigateur qu'une table a bien été créée. Il envoie la requête `DESCRIBE felins`, puis affiche un tableau HTML avec quatre colonnes, *Colonne*, *Type*, *Null* et *Clé*, avec en dessous la liste des colonnes et leurs propriétés. Pour l'utiliser avec d'autres tables, remplacez simplement le nom de table `felins` dans la requête par celui d'une autre table (exemple 10-8).

Exemple 10-8. Description de la table felins

```
<?php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);

  $query  = "DESCRIBE felins";
  $result = $conn->query($query);
  if (!$result) die ("Échec d'accès à la base de données : " . $conn->error);

  $rows = $result->num_rows;
```

```
  echo "<table><tr><th>Colonne</th><th>Type</th><th>Null</th><th>Clé</th></tr>";

  for ($j = 0 ; $j < $rows ; ++$j)
  {
    $result->data_seek($j);
    $row = $result->fetch_array(MYSQLI_NUM);

    echo "<tr>";
    for ($k = 0 ; $k < 4 ; ++$k) echo "<td>$row[$k]</td>";
    echo "</tr>";
  }

  echo "</table>";
?>
```

La sortie produite par ce programme est la suivante :

```
Colonne  Type         Null  Clé
id       smallint(6)  NO    PRI
famille  varchar(32)  NO
nom      varchar(32)  NO
age      tinyint(4)   NO
```

Supprimer une table

La suppression d'une table est tellement facile qu'elle en devient très dangereuse, donc soyez prudent. L'exemple 10-9 illustre le code nécessaire. Je vous demande toutefois de ne pas exécuter ce script, car nous aurons besoin de la table *felins* pour la suite des exemples et, si vous la supprimez, vous devrez la recréer avec l'exemple 10-7.

Exemple 10-9. Suppression de la table felins

```
<?php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);

  $query  = "DROP TABLE felins";
  $result = $conn->query($query);
  if (!$result) die ("Échec d'accès à la base de données : " . $conn->error);
?>
```

Ajouter des données

Ajoutons quelques lignes de données à l'aide du code de l'exemple 10-10.

Exemple 10-10. Ajout de données à la table felins

```
<?php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);

  $query  = "SET NAMES utf8"; // Force l'encodage en utf-8
  $result = $conn->query($query);
  if (!$result) die($conn->error);

  $query  = "INSERT INTO felins VALUES(NULL, 'Lion', 'Léo', 4)";
  $result = $conn->query($query);
  if (!$result) die ("Échec d'accès à la base de données : " . $conn->error);
?>
```

Pour ajouter deux autres lignes de données, modifiez la requête dans `$query` comme suit et appelez de nouveau le script dans le navigateur :

```
$query = "INSERT INTO felins VALUES(NULL, 'Puma', 'Grondeur', 2)";
$query = "INSERT INTO felins VALUES(NULL, 'Panthère', 'Charlot', 3)";
```

Avez-vous remarqué les `NULL` passés en premier argument ? Cela provient du fait que la colonne *id* est de type `AUTO_INCREMENT`. Par conséquent, MySQL décide de la valeur à affecter en fonction du numéro suivant disponible dans la séquence, donc nous ne pouvons passer que la valeur `NULL`, qui est ignorée.

La manière la plus efficace de remplir une table MySQL de données implique de créer un tableau et d'insérer les données en une seule requête.

Rechercher des données

À présent, la table *felins* contient quelques données. L'exemple 10-11 montre comment vérifier qu'elles ont été bien ajoutées.

Exemple 10-11. Récupérer les lignes de la table felins

```
<?php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);

  $query  = "SET NAMES utf8"; // Force l'affichage en utf-8
  $result = $conn->query($query);
  if (!$result) die($conn->error);

  $query  = "SELECT * FROM felins";
  $result = $conn->query($query);
  if (!$result) die ("Échec d'accès à la base de données : " . $conn->error);

  $rows = $result->num_rows;
  echo "<table><tr> <th>Id</th> <th>Famille</th><th>Nom</th><th>Âge</th></tr>";

  for ($j = 0 ; $j < $rows ; ++$j)
  {
    $result->data_seek($j);
```

```
        $row = $result->fetch_array(MYSQLI_NUM);

        echo "<tr>";
        for ($k = 0 ; $k < 4 ; ++$k) echo "<td>$row[$k]</td>";
        echo "</tr>";
  }

  echo "</table>";
?>
```

Ce programme envoie la requête MySQL `SELECT * FROM felins` et affiche les lignes reçues, comme suit :

```
Id Famille  Nom      Âge
1  Lion     Léo      4
2  Puma     Grondeur 2
3  Panthère Charlot  3
```

Et vous constatez au passage que la colonne *id* a bien été auto-incrémentée à chaque ligne.

Mettre à jour des données

La modification de données déjà insérées est tout aussi simple. Par exemple, le nom *Charlot* a été donné à notre panthère mais son vrai nom est *Charlie*. L'exemple 10-12 effectue la correction et donc, la mise à jour (*update*).

Exemple 10-12. Modification du nom Charlot en Charlie

```
<?php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);

  $query  = "SET NAMES utf8"; // Force l'encodage en utf-8
  $result = $conn->query($query);
  if (!$result) die($conn->error);

  $query  = "UPDATE felins SET nom='Charlie' WHERE nom='Charlot'";
  $result = $conn->query($query);
  if (!$result) die ("Échec d'accès à la base de données : " . $conn->error);
?>
```

Relancez le script de l'exemple 10-11 et vous obtenez les résultats suivants :

```
Id Famille  Nom      Âge
1  Lion     Léo      4
2  Puma     Grondeur 2
3  Panthère Charlie  3
```

Supprimer des données

Le puma nommé Grondeur a été transféré dans un autre zoo, donc vous pouvez supprimer sa fiche de la base de données, comme à l'exemple 10-13.

Exemple 10-13. Suppression de Grondeur le puma de la table felins

```
<?php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);

  $query  = "DELETE FROM felins WHERE nom='Grondeur'";
  $result = $conn->query($query);
  if (!$result) die ("Échec d'accès à la base de données : " . $conn->error);
?>
```

Le script utilise une requête `DELETE FROM` normale et, lorsque vous relancez le script de l'exemple 10-11, vous constatez que la ligne a été supprimée des résultats :

```
Id Famille  Nom      Âge
1  Lion     Léo      4
3  Panthère Charlie  3
```

Utiliser l'AUTO_INCREMENT

Lorsque vous insérez une ligne de données dont une colonne possède l'attribut `AUTO_INCREMENT`, vous ignorez la valeur que MySQL donnera à cette colonne, or cette valeur s'avère souvent indispensable pour d'autres traitements. Si vous devez absolument connaitre cette valeur, vous pouvez la déterminer par la suite à l'aide de la méthode MySQL `insert_id` de l'objet de connexion. Imaginez par exemple que vous traitez un achat et que cet achat implique la création d'un nouveau client dans la table *Clients*. Vous devez associer l'achat au client, mais vous en ignorez encore le *NumCli* qu'il vous faut pour enregistrer l'achat.

L'exemple 10-14 suivant reprend le code de l'exemple 10-10 et affiche la valeur de l'*id* après chaque insertion.

Exemple 10-14. Ajout de données à la table felins et affichage de l'id associé

```
<?php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);

  $query  = "SET NAMES utf8"; // Force l'encodage en utf-8
  $result = $conn->query($query);
  if (!$result) die($conn->error);

  $query  = "INSERT INTO felins VALUES(NULL, 'Lynx', 'Courtaud', 5)";
  $result = $conn->query($query);
  if (!$result) die ("Échec d'accès à la base de données : " . $conn->error);

  echo "L'id de l'insertion est : " . $conn->insert_id;
?>
```

Le contenu de la table est à présent le suivant (remarquez que la précédente valeur de l'*id* libérée, 2, n'est pas réutilisée parce que cela risquerait de provoquer des complications dans certains cas) :

```
Id Famille  Nom      Âge
1  Lion     Léo      4
3  Panthère Charlie  3
4  Lynx     Countaud 5
```

Exploiter les identifiants d'insertion

L'insertion a souvent lieu dans plusieurs tables à la fois : un nouveau livre, suivi d'un nouvel auteur ou un nouveau client à la suite d'un nouvel achat et ainsi de suite. Lorsque l'auto-incrémentation intervient dans une colonne d'une des tables impliquées, vous devez souvent retenir l'identifiant d'insertion pour le recopier dans la table associée.

Par exemple, admettons que ces félins puissent être « parrainés » par le public en vue de lever des fonds et que, quand un nouveau félin est inséré dans la table *felins*, nous voulons aussi créer une clé pour l'associer à un parrain ou une marraine de l'animal. Le code nécessaire pour parvenir à cela repart du code de l'exemple 10-14, sauf que l'*id* d'insertion renvoyé est stocké dans la variable `$insertID` et sert ensuite dans la requête suivante :

```
$query    = "INSERT INTO felins VALUES(NULL, 'Lynx', 'Courtaud', 5)";
$result   = $conn->query($query);
$insertID = $conn->insert_id;

$query    = "INSERT INTO parrains VALUES($insertID, 'Alain', 'Martel')";
$result   = $conn->query($query);
```

À partir de cette étape, ce félin est relié à son parrain par l'entremise de l'identifiant unique du félin, créé automatiquement par l'`AUTO_INCREMENT`.

Tirer parti des verrous

Pour que la procédure de liaison de tables par l'identifiant d'insertion soit parfaitement sûre et sécuritaire, il est préférable d'utiliser des verrous (ou des transactions, comme décrit au chapitre 9). Cela augmente légèrement le temps de réponse lorsque de nombreuses personnes soumettent des données à la même table, mais cela vaut la peine de les mettre en œuvre. La séquence à suivre est la suivante :

1. Verrouiller la première table (par ex. *felins*).
2. Insérer les données dans la première table.
3. Récupérer l'identifiant unique de la première table avec la propriété `insert_id`.
4. Déverrouiller la première table.
5. Insérer les données dans la deuxième table.

Vous pouvez libérer sans danger le verrou avant d'insérer les données dans la deuxième table, parce que vous disposez de l'identifiant d'insertion, affecté à une variable du programme. L'autre option consiste à passer par l'entremise d'une transaction à la place du verrouillage, mais cela ralentit encore un peu plus le serveur MySQL.

Exécuter d'autres requêtes

Nous avons fini de parler de chats plus ou moins gros ! Pour explorer des requêtes plus complexes, retournons aux tables *classiques* et *clients* créées au chapitre 8. Nous avons trois clients dans la table *clients* et la table *classiques* contient des informations sur quelques livres. Les tables partagent en commun la colonne des ISBN, nommée *isbn*, que nous pouvons emprunter pour effectuer des requêtes plus étendues.

Par exemple, pour afficher tous les clients avec les titres et les auteurs des livres qu'ils ont achetés, utilisez le code de l'exemple 10-15.

Exemple 10-15. Exécution d'une requête secondaire

```
<?php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);

  $query  = "SET NAMES utf8"; // force l'encodage en utf-8
  $result = $conn->query($query);
  if (!$result) die($conn->error);

  $query  = "SELECT * FROM clients";
  $result = $conn->query($query);
  if (!$result) die ("Échec d'accès à la base de données : " . $conn->error);

  $rows = $result->num_rows;

  for ($j = 0 ; $j < $rows ; ++$j)
  {
    $result->data_seek($j);
    $row = $result->fetch_array(MYSQLI_NUM);
    echo "$row[0] a acheté l'ISBN $row[1] :<br>";

    $subquery  = "SELECT * FROM classiques WHERE isbn='$row[1]'";
    $subresult = $conn->query($subquery);
    if (!$subresult) die ("Échec d'accès à la base de données : " . $conn->error);

    $subrow = $subresult->fetch_array(MYSQLI_NUM);
    echo "  '$subrow[1]' de $subrow[0]<br>";
  }
?>
```

Le programme utilise une requête initiale sur la table *clients* pour obtenir tous les clients, un à un, puis, à partir de l'ISBN du livre que chaque client a acheté, il émet une nouvelle requête sur la table *classiques* pour connaitre le titre et l'auteur de chaque livre. Voici les résultats de ce programme :

```
Jean Bergeron a acheté l'ISBN 9780099533474:
  'Le Magasin d'antiquités' de Charles Dickens
Marie Savard a acheté l'ISBN 9780582506206:
  'Orgueil et préjugés' de Jane Austen
Jacques Vachon a acheté l'ISBN 9780517123201:
  'De l'origine des espèces' de Charles Darwin
```

Dans ce cas précis, qui illustre la succession de deux requêtes, nous aurions pu obtenir les mêmes informations en une seule requête à l'aide d'une jointure naturelle, `NATURAL JOIN` (chapitre 8), comme la suivante :

```
SELECT nom,isbn,titre,auteur FROM clients
  NATURAL JOIN classiques;
```

Prévenir les tentatives de piratage

Si vous ne vous en êtes jamais préoccupé, vous aurez sans doute quelques difficultés à évaluer la dangerosité du passage d'un mot de passe non vérifié à MySQL. Ainsi, supposons que vous mettez en place du code simple pour vérifier un utilisateur, qui ressemble à ceci :

```
$user  = $_POST['user'];
$pass  = $_POST['pass'];
$query = "SELECT * FROM users WHERE user='$user' AND pass='$pass'";
```

Au premier abord, ce code semble correct. Si l'utilisateur entre les valeurs `ericdubé` et `monmdp` respectivement pour `$user` et `$pass`, alors la chaine de requête transmise à MySQL devient la suivante :

```
SELECT * FROM users WHERE user='ericdubé' AND pass='monmdp'
```

Cela fonctionne très bien ainsi. Mais qu'arrive-t-il si un utilisateur entre ce qui suit pour `$user` (et n'entre même rien pour le mot de passe) ?

```
admin' #
```

Examinons la requête envoyée à MySQL :

```
SELECT * FROM users WHERE user='admin' #' AND pass=''
```

Voyez-vous le problème qui apparait ici ? En MySQL, le symbole `#` représente le début d'un commentaire. Par conséquent, l'utilisateur s'identifie en tant qu'utilisateur *admin* (pour autant qu'un utilisateur *admin* existe), sans devoir entrer de mot de passe. La ligne suivante met en évidence la partie de la requête réellement exécutée, tandis que le reste est ignoré.

```
SELECT * FROM users WHERE user='admin' #' AND pass=''
```


Et vous pouvez vous estimer heureux si c'est là la seule tentative de malveillance à laquelle vous êtes confronté. Au moins, vous pourrez encore entrer dans votre application pour défaire toutes les modifications que l'utilisateur aura effectuées en tant qu'*admin*. Mais allons plus loin et supposons que le code de votre application supprime un utilisateur de la base de données. Le code est du genre :

```
$user  = $_POST['user'];
$pass  = $_POST['pass'];
$query = "DELETE FROM users WHERE user='$user' AND pass='$pass'";
```

De nouveau, ce code semble correct au premier abord, mais imaginez que quelqu'un entre ce qui suit dans `$user` :

```
nimportequoi' OR 1=1 #
```

MySQL interprète la requête comme suit :

```
DELETE FROM users WHERE user='nimportequoi' OR 1=1 #' AND pass=''
```

Aïe ! Cette requête est vraie dans tous les cas car `1=1` est toujours vrai, donc vous perdez tout le contenu de votre table *users* ! Alors, que faire pour contrer ce genre d'attaque ?

Étapes à suivre

La première chose à faire consiste à ne jamais vous fier aux *magic quotes* de PHP, qui convertissent en séquence d'échappement tout caractère tel que l'apostrophe et le guillemet, en le préfixant par une barre oblique inverse (\). Pourquoi ? Parce que cette fonctionnalité peut être désactivée, nombre de programmeurs préférant mettre en œuvre leurs propres mécanismes de sécurité. Donc vous n'avez aucune garantie que ce ne soit le cas sur le serveur que vous utilisez. En pratique, cette fonctionnalité est même devenue obsolète depuis PHP 5.3.0 et a été supprimée à partir de PHP 6.0.0.

À la place, utilisez toujours la méthode `real_escape_string` pour tous vos appels à MySQL. L'exemple 10-16 propose une fonction utilisable à volonté pour supprimer tous les *magic quotes* ajoutés dans une chaine entrée par l'utilisateur et l'assainir pour vous.

Exemple 10-16. Comment assainir proprement une entrée utilisateur destinée à MySQL

```
<?php
  function mysql_fix_string($conn, $string)
  {
    if (get_magic_quotes_gpc()) $string = stripslashes($string);
    return $conn->real_escape_string($string);
  }
?>
```

La fonction `get_magic_quotes_gpc` renvoie `TRUE` si les *magic quotes* sont activés. Dans ce cas, toutes les barres obliques qui ont été ajoutées doivent disparaitre, sinon la méthode `real_escape_string` risque de produire des doubles barres obliques devant certains caractères et de pervertir le contenu de certaines chaines. L'exemple 10-17 montre comment intégrer `mysql_fix_string` dans votre propre code.

Exemple 10-17. Comment accéder à MySQL en toute sécurité avec une entrée utilisateur

```
<?php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);

  $user  = mysql_fix_string($conn, $_POST['user']);
  $pass  = mysql_fix_string($conn, $_POST['pass']);
  $query = "SELECT * FROM users WHERE user='$user' AND pass='$pass'";

  // Etc...

  function mysql_fix_string($conn, $string)
  {
    if (get_magic_quotes_gpc()) $string = stripslashes($string);
    return $conn->real_escape_string($string);
  }
?>
```

Ces précautions revêtent toutefois de moins en moins d'importance parce qu'une autre méthode plus sécuritaire existe pour accéder à MySQL, qui évite de devoir utiliser de telles fonctions : l'utilisation d'espaces réservés, comme indiqué ci-après.

Utiliser des espaces réservés

Les instructions préparées avec des espaces réservés (*placehoders*) fournissent une méthode dans laquelle seules les données sont transmises à la base de données, sans aucune possibilité pour MySQL d'interpréter les données soumises par l'utilisateur (ou autres) comme des instructions, ce qui écarte donc toute potentialité de piratage.

Le fonctionnement exige que vous prépariez d'abord l'instruction que vous souhaitez exécuter dans MySQL, et que vous laissiez les portions de l'instruction réservées aux données sous forme de simples points d'interrogation.

En langage MySQL pur, les instructions préparées adoptent l'allure illustrée dans l'exemple 10-18.

Exemple 10-18. Espaces réservés en MySQL

```
PREPARE instruction FROM "INSERT INTO classiques VALUES(?,?,?,?,?)";

SET @auteur    = "Emily Brontë",
    @titre     = "Les Hauts de Hurlevent",
    @categorie = "Roman classique",
    @annee     = "1847",
    @isbn      = "9780553212587";
```



```
EXECUTE instruction USING @auteur,@titre,@categorie,@annee,@isbn;
DEALLOCATE PREPARE instruction;
```

Comme tout ceci est un peu lourd à soumettre à MySQL, l'extension `mysqli` simplifie la gestion des espaces réservés grâce à une méthode toute prête, dénommée `prepare`, dont l'appel prend l'allure suivante :

```
$instruction = $conn->prepare('INSERT INTO classiques VALUES(?,?,?,?,?)');
```

L'objet `$instruction` (ou quel que soit le nom choisi) renvoyé par cette méthode sert ensuite à envoyer les données au serveur à la place des points d'interrogation. Sa première utilisation se situe dans l'association de certaines variables dans l'ordre à chacun des points d'interrogation (les paramètres des espaces réservés), comme suit :

```
$instruction->bind_param('sssss', $auteur, $titre, $categorie, $annee, $isbn);
```

Le premier argument de `bind_param` est constitué d'une chaine représentant, dans l'ordre, le type de chaque argument suivant. Dans ce cas-ci, elle contient une suite de cinq caractères `s`, qui représentent des chaines, mais ce pourrait être n'importe quelle combinaison de caractères de la liste suivante :

- `i` Si la donnée est un entier (*integer*).
- `d` Si la donnée est un double.
- `s` Si la donnée et une chaine (*string*).
- `b` Si la donnée est un BLOB (auquel cas, elle sera transmise par paquets).

Lorsque les variables sont toutes associées à l'instruction préparée, vous devez ensuite remplir ces données avec les données à transmettre à MySQL, comme suit :

```
$auteur    = "Emily Brontë";
$titre     = "Les Hauts de Hurlevent";
$categorie = "Roman classique";
$annee     = "1847";
$isbn      = "9780553212587";
```

À ce stade, PHP dispose de tout le nécessaire pour exécuter l'instruction préparée et il ne reste plus qu'à exécuter l'instruction avec la méthode `execute` de l'objet `$instruction` précédent, comme dans la commande suivante :

```
$instruction->execute();
```

Avant de poursuivre, il est préférable de vérifier que la commande s'est exécutée correctement. Pour cela, l'objet `$instruction` possède une propriété `affected_rows` disponible après l'exécution :

```
printf("%d ligne insérée.\n", $stmt->affected_rows);
```

Cette instruction doit en principe indiquer qu'une ligne a été insérée.

Lorsque vous êtes satisfait de l'exécution de l'instruction préparée (à condition que vous n'ayez pas reçu de message d'erreur), il ne vous reste plus qu'à clôturer l'objet `$instruction` comme suit :

```
$instruction->close();
```

Et enfin, n'oubliez pas de clôturer l'objet `$conn` (si vous avez terminé avec la connexion), comme ceci :

```
$conn->close();
```

Lorsque vous rassemblez toutes ces instructions dans un programme fonctionnel, vous obtenez celui de l'exemple 10-19.

Exemple 10-19. Envoi d'une commande préparée

```
<?php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);

  $query  = "SET NAMES utf8"; // Force l'encodage en utf-8
  $result = $conn->query($query);
  if (!$result) die($conn->error);

  $instruction = $conn->prepare('INSERT INTO classiques VALUES(?,?,?,?,?)');
  $instruction->bind_param('sssss', $auteur, $titre, $categorie, $annee, $isbn);

  $auteur    = "Emily Brontë";
  $titre     = "Les Hauts de Hurlevent";
  $categorie = "Roman classique";
  $annee     = "1847";
  $isbn      = '9780553212587';

  $instruction->execute();
  printf("%d ligne insérée.\n", $instruction->affected_rows);
  $instruction->close();
  $conn->close();
?>
```

Chaque fois que vous utilisez des instructions préparées au lieu des instructions normales, vous comblez une faille de sécurité potentielle, donc cela vaut vraiment la peine de prendre le temps de les maitriser.

Prévenir l'injection de HTML

Un autre type d'injection existe, auquel vous devez porter toute votre attention, non pour la sécurité de votre site web mais pour protéger les données personnelles de vos utilisateurs. Il s'agit des *scripts intersites*, connus sous l'acronyme XSS (*cross-site scripting*).

Cela se produit lorsque vous autorisez les utilisateurs à entrer du code HTML (ou plus souvent JavaScript) dans un champ de formulaire, réaffiché ensuite par votre site. L'endroit le plus commun pour ce genre de chose se situe au niveau d'un formulaire de commentaires. Dans les cas les plus fréquents, un utilisateur malveillant tente d'écrire du code pour voler les témoins de connexion, ou *cookies*, des utilisateurs de votre site, ce qui lui permet de découvrir les paires de nom d'utilisateur et de mot de passe ou d'autres informations. Au pire, l'utilisateur malveillant peut lancer une attaque pour déposer un cheval de Troie sur l'ordinateur d'un utilisateur.

Pourtant, pour éviter cela, il suffit d'appeler la fonction `htlmentities`, qui élimine tous les codes de balisage HTML et les remplace par des formes qui affichent les caractères mais ne permettent plus à un navigateur d'agir sur eux. Ainsi, prenons le HTML suivant :

```
<script src='http://x.com/piratage.js'>
</script><script>piratage();</script>
```

Ce code charge un programme JavaScript puis exécute des fonctions malveillantes. Tandis que si vous le passez par la moulinette de `htmlentities`, celle-ci le convertit en la chaine suivante, complètement inoffensive :

```
&lt;script src='http://x.com/piratage.js'&gt; &lt;/script&gt;
&lt;script&gt;piratage();&lt;/script&gt;
```

Par conséquent, si vous devez afficher quoi que ce soit qui a été entré par des utilisateurs, soit immédiatement, soit après le stockage dans la base de données, assainissez d'abord le contenu à l'aide de la fonction `htmlentities`. Pour que ce soit systématique et plus pratique, je vous recommande de créer une nouvelle fonction, comme la première de l'exemple 10-20, qui nettoie les injections, tant en SQL qu'en XSS.

Exemple 10-20. Fonctions de prévention des attaques par injections SQL et XSS

```
<?php
  function mysql_entities_fix_string($conn, $string)
  {
    return htmlentities(mysql_fix_string($conn, $string));
  }

  function mysql_fix_string($conn, $string)
  {
    if (get_magic_quotes_gpc()) $string = stripslashes($string);
    return $conn->real_escape_string($string);
  }
?>
```

La fonction `mysql_entities_fix` appelle d'abord la fonction `mysql_fix_string`, puis passe le résultat à htmlentities avant de renvoyer la chaine aseptisée. Pour utiliser une de ces fonctions, vous devez déjà disposer d'un objet de connexion ouvert sur une base de données MySQL.

L'exemple 10-21 illustre l'utilisation de votre nouvelle « protection absolue » sur une nouvelle version de l'exemple 10-17.

Exemple 10-21. Comment accéder à MySQL en toute sécurité et éviter les attaques par XSS

```
<?php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);

  $user  = mysql_entities_fix_string($conn, $_POST['user']);
  $pass  = mysql_entities_fix_string($conn, $_POST['pass']);
  $query = "SELECT * FROM users WHERE user='$user' AND pass='$pass'";

  //Etc…

  function mysql_entities_fix_string($conn, $string)
  {
    return htmlentities(mysql_fix_string($conn, $string));
  }

  function mysql_fix_string($conn, $string)
  {
    if (get_magic_quotes_gpc()) $string = stripslashes($string);
    return $conn->real_escape_string($string);
  }
?>
```

Utiliser mysqli de manière procédurale

Si vous préférez, il existe un jeu alternatif de fonctions utilisables pour accéder à `mysqli` de manière procédurale (par opposition à la manière orientée objet que nous avons utilisée jusqu'ici).

Au lieu de créer un objet `$conn` comme ceci :

```
$conn = new mysqli($hn, $un, $pw, $db);
```

Vous pouvez écrire cela :

```
$link = mysqli_connect($hn, $un, $pw, $db);
```

Pour vérifier la réussite de la connexion et la gérer, utilisez un code du genre :

```
if (mysqli_connect_errno()) die(mysqli_connect_error());
```

Pour envoyer une requête MySQL, utilisez la forme suivante :

```
$result = mysqli_query($link, "SELECT * FROM classiques");
```

Au retour, `$result` contient les données et vous pouvez connaitre le nombre de lignes reçues comme suit :

```
$rows = mysqli_num_rows($result);
```


La variable `$rows` reçoit un entier. Pour récupérer les données réelles ligne par ligne, écrivez ce qui suit, qui retourne un tableau numérique dans `$row` :

```
$row = mysqli_fetch_array($result, MYSQLI_NUM);
```

Dans le tableau, `$row[0]` contient la première colonne de donnée, `$row[1]`, la deuxième et ainsi de suite. Comme l'illustre l'exemple 10-5, le renvoi des lignes peut se faire dans un tableau associatif, à condition de modifier la constante dans le second argument.

Lorsque vous devez déterminer l'identifiant d'insertion dans une opération d'ajout de ligne, vous pouvez utiliser la fonction `mysqli_insert_id`, comme suit :

```
$insertID = mysqli_insert_id($result);
```

En procédural, la mise en séquence d'échappement d'une chaine est également simple :

```
$echappement = mysqli_real_escape_string($link, $val);
```

Pour préparer une instruction `$inst` avec `mysqli`, utilisez ce qui suit :

```
$inst = mysqli_prepare($link, 'INSERT INTO classiques VALUES(?,?,?,?,?)');
```

L'association des variables à une instruction préparée prend la forme suivante :

```
mysqli_stmt_bind_param($inst, 'sssss', $auteur, $titre, $categorie, $annee, $isbn);
```

Ensuite, pour exécuter l'instruction préparée après l'affectation des valeurs adéquates aux variables associées, exécutez l'appel suivant :

```
mysqli_stmt_execute($inst);
```

Puis, pour clôturer une instruction et libérer ses ressources, entrez la commande :

```
mysqli_stmt_close($inst);
```

Et enfin, pour clôturer une connexion à MySQL, entrez cette commande :

```
mysqli_close($link);
```

Pour de plus amples informations sur l'utilisation d'instructions préparées (en procédural ou autre), consultez *http://us3.php.net//manual/fr/mysqli.prepare.php*. Pour de plus amples conseils sur tous les aspects de `mysqli`, consultez *http://uk1.php.net/manual/fr/book.mysqli.php*.

Vous connaissez tout ce que vous devez savoir à propos de l'intégration de PHP et de MySQL. Le chapitre suivant aborde la création de formulaires conviviaux et le traitement des données extraites de ceux-ci.

Questions

1. Comment créez-vous une connexion à une base de données MySQL à l'aide de `mysqli` ?
2. Comment soumettez-vous une requête à MySQL à l'aide de `mysqli` ?
3. Que faites-vous pour obtenir la chaine avec le message d'erreur lorsqu'une erreur `mysqli` se produit ?
4. Comment déterminez-vous le nombre de lignes de données renvoyées par une requête `mysqli` ?
5. Comment pouvez-vous récupérer une ligne de données particulière d'un ensemble de résultats en `mysqli` ?
6. Quelle méthode de `mysqli` permet de placer correctement des séquences d'échappement dans les entrées d'utilisateur pour éviter l'injection de code ?
7. Quels effets négatifs peuvent se produire lorsque vous ne clôturez pas les objets créés par les méthodes de `mysqli` ?

Retrouvez les réponses du chapitre 10 dans l'annexe A.

CHAPITRE 11

Gestion de formulaires

Les formulaires HTML constituent la voie principale d'interaction avec PHP et MySQL. Leur introduction date du tout début du développement du World Wide Web en 1993, avant même l'avènement du commerce électronique, et ils se sont toujours maintenus à cette place depuis lors, du fait de leur simplicité et de leur facilité d'utilisation.

Ils ont connus de nombreuses améliorations au cours des années, bien entendu, pour ajouter des fonctionnalités à la gestion en HTML des formulaires. Ce chapitre vous emmène à travers les toutes dernières innovations en matière de gestion de formulaire et vous présente les meilleures manières de mettre en œuvre des formulaires conviviaux et sécuritaires. De plus, vous verrez un peu plus loin que la norme HTML5 a encore amélioré un peu plus l'utilisation des formulaires.

Créer des formulaires

Le processus de gestion de formulaire compte plusieurs parties. La première se situe dans la création du formulaire, dans lequel l'utilisateur peut entrer les informations adéquates. Ces données sont envoyées au serveur web, qui les interprète, souvent en même temps qu'il vérifie l'absence d'erreurs. Si le code PHP identifie un ou plusieurs champs qui nécessitent une ressaisie, le serveur affiche de nouveau la page avec un message d'erreur ou d'avertissement. Lorsque le code est satisfait de la précision et de l'exactitude des entrées, il entreprend une certaine action qui implique généralement la base de données, par exemple pour enregistrer les informations d'un achat.

Pour constituer un formulaire, vous avez besoin au moins des éléments suivants :

- de balises d'ouverture `<form>` et de fermeture `</form>` ;
- d'un type de soumission indiquant la méthode, soit Get, soit Post ;
- d'un ou plusieurs champs d'entrée, `input` ;
- de l'URL de destination à laquelle les données sont envoyées.

L'exemple 11-1 illustre un formulaire très simple créé en PHP, que vous pouvez entrer et enregistrer dans un fichier *formtest.php*.

Exemple 11-1. Un gestionnaire de formulaire très simple en PHP, formtest.php

```
<?php // formtest.php
echo <<<_END
  <html>
    <head>
      <title>Formulaire de test</title>
    </head>
    <body>
      <form method="post" action="formtest.php">
        Quel est ton nom?
        <input type="text" name="nom">
        <input type="submit">
      </form>
    </body>
  </html>
_END;
?>
```

La première chose à remarquer à propos de cet exemple, c'est que, comme vous l'avez déjà vu précédemment dans le livre, au lieu de quitter le code PHP et d'y replonger ensuite, la construction echo `<<<_END...._END` permet de sortir du HTML sur plusieurs lignes au sein même du code PHP.

Dans cette sortie sur plusieurs lignes figure du code plutôt standard pour entamer un document HTML, avec l'affichage de son titre (`<title>...</title>`), puis le début de son corps (`<body>...</body>`). Le formulaire vient s'insérer entre les balises du corps. Le formulaire (`<form>...</form>`) est réglé pour utiliser la méthode d'échange Post avec le programme *formtest.php*, qui correspond au programme PHP lui-même.

Le reste du programme clôture tous les éléments ouverts auparavant : le formulaire, le corps du document HTML, puis l'instruction PHP `echo <<<_END`. Le résultat de l'appel de ce programme dans un navigateur web correspond à l'illustration de la figure 11-1.

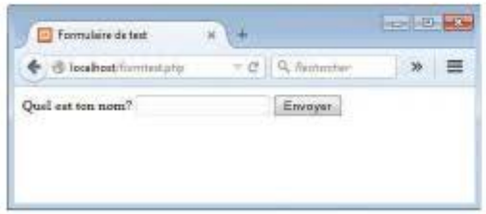


Figure 11-1. Résultat de l'ouverture de formtest.php dans un navigateur web

Rechercher les données soumises

L'exemple 11-1 ne représente qu'une seule partie d'un processus qui en comporte plusieurs. Lorsque vous entrez un nom et cliquez sur *Envoyer*, absolument rien ne se passe, à part un nouvel affichage du formulaire, sans donnée. Pour aller plus loin, nous devons ajouter un peu de code PHP pour traiter les données soumises par le formulaire.

L'exemple 11-2 étend le programme précédent pour inclure du traitement de données. Entrez-le ou modifiez *formtest.php* et ajoutez les nouvelles lignes. Enregistrez le nouveau fichier sous *formtest2.php* et essayez ce programme. La figure 11-2 montre les résultats de ce programme et l'entrée d'un nom.

Exemple 11-2. Version modifiée de formtest.php

```
<?php // formtest2.php
  if (isset($_POST['nom'])) $nom = $_POST['nom'];
  else $nom = "(Inconnu)";

  echo <<<_END
  <html>
    <head>
      <title>Formulaire de test</title>
    </head>
    <body>
      Ton nom est : $nom<br>
      <form method="post" action="formtest2.php">
        Quel est ton nom?
        <input type="text" name="nom">
        <input type="submit">
      </form>
    </body>
  </html>
_END;
?>
```

Les seuls changements apportés par rapport à l'exemple précédent se situe au niveau des deux lignes au début du programme, qui vérifient la présence, dans le tableau associatif `$_POST`, du champ nom qui a été envoyé. Le chapitre 10 a présenté le tableau associatif `$_POST`, qui contient un élément pour chaque champ d'un formulaire HTML associé. Dans l'exemple 11-2, le nom du champ d'entrée s'appelle *nom* et la méthode d'envoi du formulaire correspond à Post, donc l'élément `nom` du tableau `$_POST` contient la valeur, accessible par `$_POST['nom']`.

La fonction PHP `isset` permet de vérifier si `$_POST['nom']` contient une valeur. Si rien n'a été publié, alors le programme affecte `(Inconnu)` à la variable `$nom` ; sinon, il affecte la valeur entrée à la variable. Une seule ligne vient s'ajouter dans le corps du document, juste avant le formulaire, pour afficher cette valeur, mémorisée dans la variable `$nom`.

Remarquez que les éléments `<input>` de cet exemple n'utilisent pas la forme `/>` d'autofermeture, car dans le monde nouveau du HTML5, ce style est facultatif (d'ailleurs, il n'était même pas exigé dans HTML4 ; il n'était recommandé que parce qu'il était prévu que XHTML prenne le pas sur HTML à un certain niveau mais cela n'a jamais été le cas). Comme je suis partisan du moindre effort en matière de programmation, je n'utilise plus ces caractères, sauf pour du véritable XHTML, auquel cas ce type de fermeture demeure d'actualité, pour économiser la frappe d'une espace et d'une barre oblique dans toutes les balises de fermeture de ce type.

Figure 11-2. Réaffichage de formtest.php avec gestion des données

Une vieille solution sur la sellette : register_globals

Avant que les préoccupations relatives à la sécurité ne deviennent si importantes, le comportement par défaut de PHP consistait à affecter les tableaux `$_POST` et `$_GET` directement à des variables PHP. Ainsi, il n'était pas nécessaire de passer par une instruction du genre `$nom=$_POST['nom']`, parce que `$nom` recevait automatiquement sa valeur au démarrage du programme PHP.

Au départ, avant la version 4.2.0 de PHP, cela semblait utile pour économiser beaucoup d'écriture de code mais, dans les faits, cette pratique a été abandonnée et cette possibilité est désormais désactivée par défaut. Si vous deviez constater que `registrer_globals` est activé sur un serveur web de production, demandez d'urgence à votre administrateur système de le désactiver !

Pourquoi désactiver `registrer_globals` ? Il permet à n'importe qui d'entrer une entrée Get avec quelque chose à la fin d'une URL, du genre *http://monserveur.com?ecraser=1*. Si jamais votre code utilise une variable `$ecraser` et que vous avez oublié de l'initialiser (par exemple avec l'instruction `$ecraser=0;`), cela laisse le programme en situation de vulnérabilité face à cette faille.

En fait, comme de nombreuses installations sur la Toile ont encore et toujours cette faille de sécurité, je vous conseille de toujours initialiser les variables que vous utilisez, au cas où votre code s'exécuterait sur un tel système. L'initialisation figure aussi parmi les bonnes pratiques de programmation, car vous avez la possibilité de commenter chaque initialisation pour vous rappeler (et aux autres programmeurs) à quoi sert une variable.

Lorsque vous vous retrouvez à maintenir du code qui semble supposer des valeurs pour certaines variables pour des raisons non évidentes, vous pouvez être à peu près certain que le programmeur a rédigé le code en pensant à l'utilisation de `register_globals` et que ces valeurs sont censées provenir de l'extraction automatique d'un Post ou d'un Get. Si c'est le cas, je vous conseille instamment de récrire ce code pour charger ces variables de manière explicite à partir du bon tableau `$_POST` ou `$_GET`.

Valeurs par défaut

Les visiteurs apprécient que vous leur offriez parfois des valeurs prédéfinies dans un formulaire web. Supposons que vous placez un widget de calculatrice de calcul de prêt bancaire sur un site consacré à l'immobilier. Il est raisonnable de proposer des valeurs par défaut, disons de 25 ans et 6 % d'intérêt annuel, pour que l'utilisateur n'ait plus qu'à entrer le montant du capital emprunté ou le montant maximal qu'il peut rembourser par mois.

Dans ce genre de cas, le code HTML pour établir ces deux valeurs peut prendre l'aspect illustré dans l'exemple 11-3.

Exemple 11-3. Définition de valeurs par défaut

```
<form method="post" action="calc.php"><pre>
 Capital emprunté <input type="text" name="capital">
       Mensualité <input type="text" name="mensualite">
  Nombre d'années <input type="text" name="annees" value="25">
    Taux d'intérêt <input type="text" name="taux"  value="6">
                   <input type="submit">
</pre></form>
```

Si vous souhaitez essayer ce code (et les autres exemples de code HTML), entrez-le et enregistrez-le sous un nom de fichier d'extension *.html* (ou *.htm*), comme *calc.html* (ou *calc.htm*), puis chargez le fichier dans le navigateur web.

Examinez les troisième et quatrième champs de formulaire. Le fait de donner une valeur à l'attribut `value` permet d'afficher une valeur prédéfinie dans le champ, que l'utilisateur peut modifier. Si vous choisissez des valeurs raisonnables, vous apportez plus de convivialité à vos formulaires et évitez aux utilisateurs des entrées non indispensables. La figure 11-3 illustre les résultats produits par ce formulaire. Bien entendu, il n'a été conçu que dans un esprit didactique et, comme le programme *calc.php* n'existe pas, le formulaire ne produit qu'une erreur de page introuvable si vous cliquez sur *Envoyer*.

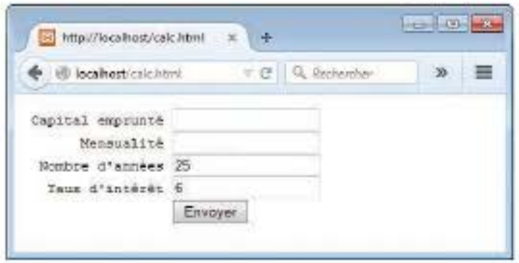


Figure 11-3. Utilisation de valeurs par défaut dans des champs de formulaire

Les valeurs par défaut sont aussi valables pour des champs masqués (*hidden*), si vous souhaitez transmettre des informations supplémentaires de la page web au programme, en plus des entrées de l'utilisateur. Nous examinons les champs masqués plus loin dans ce chapitre.

Types d'entrées

Les formulaires HTML présentent l'avantage d'une grande souplesse et de permettre l'utilisation d'une vaste gamme de types d'entrées, des zones de texte et zones de texte étendue (*text area*) aux cases à cocher, boutons d'option et bien d'autres encore.

Zones de texte : <input type="text">

La zone de texte (*text box*) constitue le type d'entrée le plus fréquemment utilisé. Elle accepte toute une variété de textes alphanumériques et autres caractères dans une zone d'une seule ligne. La forme générale d'une zone de texte d'entrée est la suivante :

```
<input type="text" name="nom" size="taille" maxlength="longueurmax" value="valeur">
```

Nous avons déjà évoqué les attributs `name` et `value` mais nous voyons ici deux nouveaux attributs : `size` et `maxlength`. L'attribut `size` spécifie la largeur de la zone de texte (en nombre de caractères de la police courante), telle qu'elle doit apparaitre à l'écran, tandis que `maxlength` définit le nombre maximum de caractères que l'utilisateur peut entrer dans ce champ.

Les seuls attributs obligatoires sont `type`, qui indique au navigateur le type de champ d'entrée auquel il doit s'attendre, et `name`, pour donner à cette entrée le nom qui servira au traitement du champ lors de la réception du formulaire soumis.

Zones de texte étendues : <textarea>

Lorsque vous devez accepter du texte qui occupe plus d'une ligne de texte court, utilisez une zone de texte étendue (*text area*). Elle ressemble beaucoup à une zone de texte normale mais, comme elle accepte plusieurs lignes, elle possède quelques attributs différents. Sa forme générale est la suivante :


```
<textarea name="nom" cols="largeur" rows="hauteur" wrap="type">
</textarea>
```

La première chose que vous devez remarquer réside dans le fait que `<textarea>` (en un seul mot !) possède sa propre balise, qui ne constitue pas un sous-type de la balise `<input>`. Par conséquent, elle exige la balise de fermeture `</textarea>`, au lieu de `</input>`.

Il y a également une différence au niveau de la valeur par défaut : au lieu d'un attribut par défaut, lorsque vous souhaitez préafficher du texte, placez-le avant la balise de fermeture `</textarea>`. Il s'affiche dans la zone de texte étendue et l'utilisateur a tout le loisir de le modifier :

```
<textarea name="nom" cols="largeur" rows="hauteur" wrap="type">
Voici du texte prédéfini.
</textarea>
```

Pour contrôler la largeur et la hauteur, précisez les attributs `cols` et `rows`, qui utilisent l'espacement de caractères de la police en cours pour déterminer la taille de la zone. Si vous omettez ces valeurs, une zone d'entrée s'affiche par défaut, dont les dimensions varient selon le navigateur utilisé pour l'afficher, donc précisez toujours ces dimensions pour garantir un affichage adéquat du formulaire, quel que soit le navigateur.

Enfin, vous pouvez contrôler la manière d'appliquer le retour à la ligne du texte entré dans la zone (et si ce retour à la ligne sera répercuté dans le texte envoyé au serveur) à l'aide de l'attribut `wrap`. Le tableau 11-1 énumère les types de retour à la ligne disponibles. Si vous omettez l'attribut `wrap`, c'est le retour `soft` qui s'applique par défaut.

Tableau 11-1. Les types de retour à la ligne disponibles dans une zone de texte étendue

Type	Action
`off`	Pas de retour à la ligne automatique et les lignes apparaissent comme l'utilisateur les entre.
`soft`	Retour à la ligne à l'affichage mais le texte est envoyé au serveur comme une longue chaine de texte, sans retour à la ligne ni de nouvelle ligne.
`hard`	Retour à la ligne à l'affichage et le texte est envoyé au serveur avec des retours à la ligne et des nouvelles lignes adaptés.

Cases à cocher : <input type="checkbox">

Lorsque vous souhaitez proposer à l'utilisateur plusieurs choix (ou options) possibles, parmi lesquels il est susceptible d'en sélectionner un ou plusieurs, les cases à cocher (*checkbox*) constituent la meilleure manière de les présenter. Voici leur forme générale :

```
<input type="checkbox" name="nom" value="valeur" checked="checked">
```

Si vous incluez l'attribut `checked` (qui signifie cochée), alors la case est précochée au moment de l'affichage dans le navigateur. La chaine affectée à l'attribut peut être soit un double guillemet, soit la valeur `"checked"`, ou sans aucune valeur. Si vous ne précisez pas l'attribut, la case s'affiche décochée. Ainsi, pour créer une case non cochée, écrivez :

```
J'accepte <input type="checkbox" name="accepte">
```

Si l'utilisateur ne coche pas la case, aucune valeur n'est envoyée à la soumission. Mais s'il coche la case, la valeur `"on"` est envoyée à la soumission pour le champ dénommé `accepte`. Si vous préférez utiliser votre propre valeur lors de la soumission d'une case cochée, au lieu du mot *on*, indiquez une autre valeur, par exemple le nombre 1, selon la syntaxe suivante :

```
J'accepte <input type="checkbox" name="accepte" value="1">
```

En revanche, si vous souhaitez proposer à l'utilisateur de s'abonner à votre bulletin d'information au moment de soumettre un formulaire, alors vous pourriez afficher la case correspondante déjà cochée en guise de valeur par défaut :

```
Je m'abonne? <input type="checkbox" name="bulletin" checked="checked">
```

Si vous voulez permettre la sélection d'un groupe d'éléments en même temps, affectez-leur tous le même `name`. Cependant, seul le dernier élément coché sera envoyé en soumission, à moins que vous passiez un tableau de valeurs à `name`. Ainsi, l'exemple 11-4 permet à l'utilisateur de sélectionner ses glaces préférées (la figure 11-4 montre le résultat dans un navigateur).

Exemple 11-4. Proposition de plusieurs choix par cases à cocher

```
  Vanille <input type="checkbox" name="glace" value="Vanille">
Chocolat <input type="checkbox" name="glace" value="Chocolat">
   Fraise <input type="checkbox" name="glace" value="Fraise">
```

Figure 11-4. Utilisation de cases à cocher pour des sélections rapides

Si une seule case parmi toutes celles du groupe est cochée, la deuxième, par exemple, alors ce seul élément sera envoyé à la soumission (le champ `glace` reçoit la valeur `"Chocolat"`). Mais, si deux cases ou plus sont cochées, seule la valeur de la dernière qui a reçu la coche est envoyée, les valeurs précédentes étant ignorées.

Si vous souhaitez un comportement exclusif, de sorte qu'une seule case puisse être cochée parmi toutes, alors utilisez plutôt un bouton d'option (voir section suivante). Mais pour des sélections multiples dans un groupe de cases à cocher, modifiez légèrement le code HTML, comme dans l'exemple 11-5, pour ajouter des crochets droits accolés à la fin du nom de la variable (`glace[]`).

Exemple 11-5. Soumission de plusieurs valeurs dans un tableau

```
  Vanille <input type="checkbox" name="glace[]" value="Vanille">
Chocolat <input type="checkbox" name="glace[]" value="Chocolat">
   Fraise <input type="checkbox" name="glace[]" value="Fraise">
```

Dans ce cas-ci, lors de l'envoi du formulaire, si au moins une de ces cases est cochée, un tableau dénommé `glace` est envoyé, avec les valeurs des cases cochées. Dans tous les cas, vous pouvez extraire soit la valeur soumise seule, soit le tableau de valeurs vers une variable, comme suit :

```
$glace = $_POST['glace'];
```

Si vous publiez le champ `glace` sous forme d'une valeur unique, alors `$glace` se présente sous forme d'une simple chaine, par exemple `"Fraise"`. En revanche, si vous définissez l'entrée `glace` comme un tableau (comme à l'exemple 11-5), alors `$glace` est un tableau et son nombre d'éléments correspond au nombre de valeurs soumises (et de cases cochées dans le formulaire). Le tableau 11-2 dénombre les sept jeux de valeurs possibles qui peuvent être soumises pour une, deux ou les trois coches disponibles. Dans chaque cas, un tableau est créé contenant un, deux ou trois éléments.

Tableau 11-2. Les sept possibilités de jeux de valeurs pour le tableau $glace

Une valeur soumise	Deux valeurs soumises	Trois valeurs soumises

`$glace[0] => Vanille`	`$glace[0] => Vanille`
	`$glace[1] => Chocolat`
		`$glace[0] => Vanille`
		`$glace[1] => Chocolat`
		`$glace[2] => Fraise`

`$glace[0] => Chocolat`	`$glace[0] => Vanille` `$glace[1] => Fraise`	
`$glace[0] => Fraise`	`$glace[0] => Chocolat` `$glace[1] => Fraise`	

En PHP, le code est très simple pour afficher le contenu de `$glace`, si c'est un tableau :

```
foreach($glace as $element) echo "$element<br>";
```

Il s'agit là d'une construction `foreach` standard en PHP, pour itérer parmi le contenu du tableau `$glace` et affecter la valeur de chaque élément à la variable `$element`, affichée ensuite par une commande `echo`. Le `<br>` est une balise de mise en forme de HTML qui impose un retour à la ligne après chaque parfum à afficher. Par défaut, les cases à cocher sont carrées.

Boutons d'options : <input type="radio">

Les boutons d'options ou boutons radio portent ce nom parce qu'ils ressemblent aux boutons de commande présents sur les anciens postes de radio, où lorsque l'on pressait un bouton, les autres boutons déjà enfoncés ressautaient. Ils servent à sélectionner une seule valeur parmi plusieurs. Tous les boutons d'un même groupe doivent porter le même nom et, comme une seule option de sélection est possible, une seule valeur est retournée, donc vous n'êtes pas obligé de lui passer un tableau.

Ainsi, si votre site web propose une sélection de délais de livraison pour les articles achetés dans votre boutique en ligne, vous pourriez écrire du code HTML comme celui de l'exemple 11-6 (la figure 11-5 illustre les résultats d'affichage).

Exemple 11-6. Utilisation de bouton d'options

```
 De 8 h à 12 h<input type="radio" name="heure" value="1">
De 12 h à 16 h<input type="radio" name="heure" value="2" checked="checked">
De 16 h à 20 h<input type="radio" name="heure" value="3">
```

Figure 11-5. Utilisation de boutons d'options pour une seule sélection parmi plusieurs

Ici, la deuxième option, *De 12 h à 16 h*, est sélectionnée par défaut. Ce choix garantit la sélection d'au moins un horaire par l'utilisateur, que celui-ci peut changer pour une des deux autres possibilités qu'il préfère. Si une des options n'est pas présélectionnée, l'utilisateur risque d'oublier d'en choisir une et aucune valeur n'est envoyée à la soumission. Par défaut, les boutons à options sont ronds.

Champs masqués : <input type="hidden">

Il est souvent pratique de disposer de champs de formulaire, notamment pour suivre l'état d'une entrée de l'utilisateur. Ainsi, vous pouvez savoir si un formulaire a déjà été soumis. Pour y parvenir, ajoutez un peu de code HTML dans votre code PHP du genre :

```
echo '<input type="hidden" name="soumis" value="oui">'
```

Cette simple instruction `echo` en PHP ajoute un champ d'entrée au formulaire HTML. Supposons que le formulaire a été rédigé hors du programme et qu'il s'affiche à l'écran de l'utilisateur. La première fois que le programme PHP reçoit les entrées, cette ligne de programme n'a pas encore été entrée, donc le champ dénommé `soumis` n'existe pas. Le programme PHP recrée le formulaire, lui ajoute le champ masqué. Lorsque le visiteur soumet de nouveau le formulaire, le programme en reçoit les entrées mais, cette fois, avec le champ `soumis` affecté de la valeur `"oui"`. Le code PHP peut vérifier si le champ est présent, comme suit :

```
if (isset($_POST['soumis']))
{...
```

Les champs masqués s'avèrent utiles pour mémoriser également d'autres détails, comme la chaine d'un identifiant de session pour identifier un utilisateur, et ainsi de suite.

Ne vous fiez jamais aux champs masqués comme s'ils étaient sécuritaires, parce qu'ils ne le sont pas. N'importe qui peut visualiser le HTML qui les contient, à l'aide de la fonction d'affichage du code source du navigateur.

Liste déroulante : <select>...<option>

La balise `<select>` permet de créer une liste déroulante d'options, avec une sélection simple ou multiple. Elle respecte la syntaxe suivante :

```
<select name="nom" size="taille" multiple="multiple">
```

L'attribut `size` indique le nombre de lignes à afficher. Un clic sur l'affichage déroule la liste et affiche toutes les options. L'attribut `multiple` permet à l'utilisateur de sélectionner plusieurs options de la liste, en pressant la touche *Ctrl* pendant les clics. Ainsi, pour permettre à l'utilisateur de sélectionner son légume favori parmi une liste de cinq, utilisez du code HTML du genre de celui de l'exemple 11-7, qui n'autorise qu'une seule sélection.

Exemple 11-7. Utilisation de select

```
Légumes
<select name="legumes" size="1">
  <option value="Pois">Petits pois</option>
  <option value="Haricots">Haricots</option>
  <option value="Carottes">Carottes</option>
  <option value="Laitue">Laitue</option>
  <option value="Broccoli">Broccoli</option>
</select>
```

Ce code HTML affiche cinq options dont la première, *Petits pois*, est sélectionnée parce que c'est la première de la liste. La figure 11-6 illustre la sortie lors d'un clic sur la flèche de la liste déroulante et avec l'option *Carottes* mise en évidence. Pour sélectionner une autre option comme la valeur présélectionnée, comme *Haricots*, utilisez la balise `<selected>`, comme suit :

```
<option selected="selected" value="Haricots">Haricots</option>
```

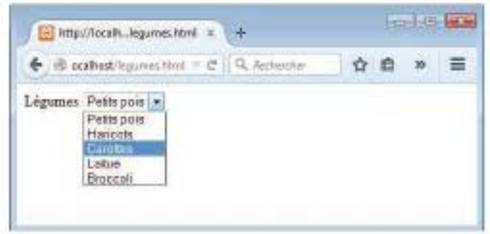


Figure 11-6. Création d'une liste déroulante avec <select>

Pour autoriser l'utilisateur à sélectionner plusieurs éléments, suivez l'exemple 11-8.

Exemple 11-8. Sélection avec l'attribut multiple

```
Légumes
<select name="legumes" size="5" multiple="multiple">
  <option value="Pois">Petits pois</option>
  <option value="Haricots">Haricots</option>
  <option value="Carottes">Carottes</option>
  <option value="Laitue">Laitue</option>
  <option value="Broccoli">Broccoli</option>
</select>
```

Ce code diffère à peine du précédent : seule la taille, `size`, est portée à 5 et l'attribut `multiple` apparaît. La figure 11-7 montre que, cette fois, l'utilisateur peut sélectionner plusieurs options et, pour ce faire, il lui suffit de maintenir la touche *Ctrl* pressée pendant qu'il clique sur d'autres options. Vous pouvez ignorer l'attribut `size` si vous le souhaitez car l'affichage reviendra au même. Cependant, pour une plus longue liste, la liste déroulée risque de prendre trop de place à l'écran, donc je vous conseille de choisir un nombre acceptable de lignes et de le respecter chaque fois. Je vous déconseille aussi d'utiliser des listes déroulantes à sélections multiples de moins de deux lignes affichées, car certains navigateurs risquent de ne pas afficher correctement les barres de défilement nécessaires pour accéder aux différentes options.

Figure 11-7. Liste déroulante avec sélection multiple

L'attribut `selected` permet aussi de présélectionner plusieurs options dans une liste à sélections multiples.

Étiquettes : <label>

La balise `<label>` offre un peu plus de convivialité dans l'utilisation des formulaires. Entourez un élément du formulaire avec cette balise et vous permettez ainsi de sélectionner ce dernier d'un clic sur toute partie visible comprise entre les balises ouvrante `<label>` et fermante `</label>`.

Ainsi, pour reprendre l'exemple du choix d'un horaire de livraison, l'utilisateur peut cliquer sur le bouton d'option et sur son étiquette associée, à l'aide de ce qui suit :


```
<label>De 8 h à 12 h<input type="radio" name="heure" value="1"></label>
```

Le texte n'est pas souligné comme s'il s'agissait d'un lien hypertexte mais, lorsque la souris le survole, le pointeur se change en flèche au lieu du pointeur de texte, pour indiquer que tout l'élément peut recevoir les clics.

Bouton de soumission

Pour correspondre au type de formulaire à soumettre, vous pouvez changer le texte du bouton de soumission en ce que vous voulez, à l'aide de l'attribut `value`, comme suit :

```
<input type="submit" value="Rechercher">
```

Pour remplacer le bouton textuel classique par une image de votre choix, utilisez la variante suivante :

```
<input type="image" name="submit" src="imageRechercher.gif">
```

Aseptiser les entrées

Nous revenons à présent à la programmation en PHP. Nous ne pourrions jamais trop insister sur l'aspect champ de mines en termes de sécurité que représente la saisie d'entrées par un utilisateur et il est impératif d'apprendre à traiter toutes ces données avec la plus grande prudence dès le départ. Nettoyer, aseptiser (*sanitize*) les entrées de l'utilisateur de toute tentative de piratage n'est pas une opération si difficile et elle doit être menée avec soin.

La première chose à garder à l'esprit réside dans le fait que, quelles que soient les contraintes que vous placez dans un formulaire HTML pour restreindre les types et les tailles des entrées, c'est un jeu d'enfant pour un pirate que d'utiliser la fonctionnalité d'affichage du code source d'une page pour en extraire le formulaire et de le modifier pour injecter des entrées malveillantes dans votre site web.

Par conséquent, vous ne pouvez jamais faire confiance à aucune variable que vous extrayez des tableaux `$_GET` et `$_POST`, tant que vous ne l'avez pas aseptisée. Si vous ne le faites pas, les utilisateurs tenteront d'injecter du JavaScript dans les données pour interférer avec le fonctionnement de votre site ou même pour tenter d'ajouter des commandes MySQL et de compromettre votre base de données.

En conséquence, au lieu de simplement écrire du code comme le suivant pour lire une entrée de l'utilisateur,

```
$variable = $_POST['entree_utilisateur'];
```

vous devriez utiliser une ou plusieurs lignes de code du style des suivantes. Par exemple, pour empêcher d'injecter des séquences d'échappement dans une chaine à présenter à MySQL, préférez la ligne suivante. Rappelez-vous que cette fonction prend en compte le jeu de caractères d'une connexion à MySQL donc elle doit s'appliquer à un objet de connexion `mysqli` (ici, `$connexion`), comme l'indique le chapitre 10.

```
$variable = $connexion->real_escape_string($variable);
```

Pour rappel également, le moyen le plus sûr de sécuriser MySQL contre les tentatives de piratage consiste à utiliser des espaces réservés et des instructions préparées, comme décrits au chapitre 10. Si vous le faites pour tous les accès à MySQL, il devient moins indispensable de convertir en séquences d'échappement les données à transférer vers et à partir de la base de données. Vous devez toutefois toujours assainir les entrées que vous insérez dans le code HTML.

Pour supprimer toutes les barres obliques indésirables, utilisez ceci :

```
$variable = stripslashes($variable);
```

Et, pour retirer tout HTML d'une chaine, utilisez ce qui suit :

```
$variable = htmlentities($variable);
```

Ceci convertirait par exemple la chaine `<b>Salut</b>` en `&lt;b&gt;Salut&lt;/b&gt;`, qui s'affiche en tant que texte et n'est plus interprétée comme des balises HTML.

Enfin, si vous souhaitez radicalement supprimer tout code HTML d'une entrée, utilisez la fonction suivante (mais appelez-la *avant* `htmlentities`, qui remplace déjà les parenthèses en chevrons appartenant aux balises HTML, comme ci-dessus) :

```
$variable = strip_tags($variable);
```

En pratique, jusqu'à ce que vous sachiez exactement le niveau d'aseptisation dont vous avez besoin pour un programme, l'exemple 11-9 illustre deux fonctions qui rassemblent toutes ces vérifications pour assurer un très bon niveau de sécurité.

Exemple 11-9. Les fonctions sanitizeString et sanitizeMySQL

```
<?php
  function sanitizeString($var)
  {
    if (get_magic_quotes_gpc())
      $var = stripslashes($var);
    $var = strip_tags($var);
    $var = htmlentities($var);
    return $var;
  }

  function sanitizeMySQL($connexion, $var)
  {
    $var = $connexion->real_escape_string($var);
    $var = sanitizeString($var);
    return $var;
  }
?>
```

Ajoutez ce code à la fin de vos programmes PHP et appelez les fonctions pour toute entrée d'utilisateur à nettoyer, comme suit :

```
$var = sanitizeString($_POST['entree_utilisateur']);
```


Ou, si vous disposez d'une connexion ouverte à MySQL, et d'un objet de connexion `mysqli` (appelé `$connexion`, ici), écrivez :

```
$var = sanitizeMySQL($connexion, $_POST['entree_utilisateur']);
```

Si vous utilisez la version procédurale de l'extension `mysqli` (auquel cas `$connexion` est un descripteur de fichier et non plus un objet), modifiez la fonction `sanitizeMySQL` pour appeler la fonction `mysqli_real_escape_string`, comme suit :

```
$var = mysqli_real_escape_string($connexion, $var);
```

Exemple de programme

À présent, examinons un programme PHP avec un formulaire HTML qui pourrait servir dans le monde réel. Créez le fichier *convert.php* de l'exemple 11-10, entrez le programme et essayez-le par vous-même.

Exemple 11-10. Programme de conversion de degrés Fahrenheit en degrés Celsius et inversement

```
<?php // convert.php
  $f = $c = '';

  if (isset($_POST['f'])) $f = sanitizeString($_POST['f']);
  if (isset($_POST['c'])) $c = sanitizeString($_POST['c']);

  if ($f != '')
  {
    $c = intval((5 / 9) * ($f - 32));
    $out = "$f °F équivaut à $c °C";
  }
  elseif($c != '')
  {
    $f = intval((9 / 5) * $c + 32);
    $out = "$c °C équivaut à $f °F";
  }
  else $out = "";

  echo <<<_END
<html>
  <head>
    <title>Convertisseur de températures</title>
  </head>
  <body>
    <pre>
      Entrez soit les °F, soit les °C et cliquez sur Convertir

      <b>$out</b>
```

```
      <form method="post" action="convert.php">
        Fahrenheit <input type="text" name="f" size="7">
           Celsius <input type="text" name="c" size="7">
                   <input type="submit" value="Convertir">
      </form>
    </pre>
  </body>
</html>
_END;

  function sanitizeString($var)
  {
    $var = stripslashes($var);
    $var = strip_tags($var);
    $var = htmlentities($var);
    return $var;
  }
?>
```

Lorsque vous appelez *convert.php* dans un navigateur, les résultats prennent l'allure illustrée par la figure 11-8.

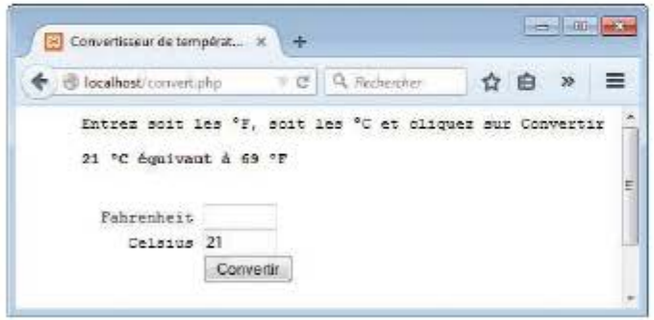


Figure 11-8. Le programme de conversion de températures en action

Examinons ce programme en détail. La première ligne initialise les variables `$c` et `$f` au cas où elles n'auraient pas été publiées vers le programme. Les deux lignes suivantes prennent les valeurs des champs nommés respectivement `f` et `c`, pour une entrée en degrés Fahrenheit ou en degrés Celsius. Si l'utilisateur entre les deux, la partie Celsius est simplement ignorée et la valeur en °F est convertie. Par mesure de sécurité, la nouvelle fonction `sanitizeString` de l'exemple 11-9 intervient pour aseptiser l'entrée.

À ce stade, nous avons des valeurs soumises dans une des variables, dans les deux ou des chaines vides dans les deux. La partie de code suivante emprunte une structure `if...elseif...else` qui vérifie d'abord si `$f` a une valeur. Sinon, elle vérifie si `$c` a une valeur, sinon la variable d'affichage `$out` reçoit la chaine vide (nous y reviendrons dans un instant).

Si `$f` contient une valeur, la variables `$c` reçoit la valeur issue d'une expression mathématique simple qui convertit la valeur de `$f` de degrés Fahrenheit en degrés Celsius. La formule utilisée est °C = (5 / 9) × (°F -32). La variable `$out` reçoit le résultat de la conversion avec un message qui explique celle-ci.

Si `$f` est vide et si `$c` contient une valeur, l'opération inverse est effectuée pour convertir la valeur de `$c` de degrés Celsius en degrés Fahrenheit et `$f` reçoit le résultat. La formule de conversion est °F = (9 / 5) × °C + 32. Comme dans la section précédente, le résultat est versé dans `$out` avec un message explicatif de la conversion.

Dans ces deux conversions, la fonction PHP `intval` permet de convertir le résultat du calcul en une valeur entière. Elle n'est pas indispensable mais donne un résultat plus joli.

Toutes les opérations mathématiques effectuées, le programme affiche à présent le code HTML, qui débute par les classiques entête (*head*) et titre (*title*), puis un peu de texte de présentation avant d'afficher le contenu de la variable `$out`. Si aucune conversion de température n'a eu lieu, `$out` a la valeur `NULL` et rien n'apparait, ce qui correspond à nos attentes quand le formulaire s'affiche pour la première fois. En revanche, quand une conversion a eu lieu, `$out` affiche le texte avec la conversion effectuée.

Ensuite, nous entrons dans le formulaire, défini avec la méthode Post et l'action égale au programme lui-même, *convert.php*. Au sein du formulaire, deux entrées permettent de saisir une valeur soit en °F, soit en °C. Le bouton de soumission reçoit le texte `Convertir` et s'affiche, puis le formulaire se ferme.

Après le code nécessaire pour clôturer proprement le document HTML, nous reprenons la fonction `sanitizeString` extraite de l'exemple 11-9. Pour tester le programme, entrez des valeurs différentes dans les champs du formulaire. Juste pour le plaisir, pouvez-vous trouver une valeur de température pour laquelle les valeurs en °F et en °C sont identiques ?

Tous les exemples de ce chapitre ont utilisé la méthode Post pour envoyer les données de formulaires. Je vous recommande instamment de préférer cette méthode, car c'est la plus propre et la plus sécuritaire. Il est toutefois facile de modifier un formulaire pour utiliser la méthode Get, tant que les valeurs sont récupérées ensuite dans le tableau `$_GET` au lieu de celui de `$_POST`. Les raisons de passer par la méthode Get résident dans la possibilité de mémoriser dans un favori ou un marque-page l'URL et les valeurs d'une recherche, par exemple, ou de permettre de placer l'URL avec ses variables dans un lien hypertexte sur une autre page.

Quoi de neuf en HTML5 ?

Avec HTML5, les développeurs peuvent compter sur un certain nombre d'améliorations très pratiques de la gestion des formulaires pour en faciliter plus que jamais l'utilisation, grâce à de nouveaux attributs, des sélecteurs de couleurs, de dates, d'heures, ainsi que de

nouveau types d'entrées, même si tous les navigateurs ne reconnaissent pas encore toutes ces fonctionnalités. Les nouvelles fonctionnalités suivantes sont toutefois prises en charge par tous les navigateurs.

Attribut autocomplete

L'attribut `autocomplete` s'applique soit à l'élément `<form>`, soit à n'importe quel des types `color`, `date`, `password` (mot de passe, avec les caractères masqués par des étoiles), `range` (plage de valeurs), `search` (recherche), `tel` (téléphone), `text` ou `url` associables à l'élément `<input>`.

Lorsque vous activez la saisie semi-automatique, les entrées précédentes de l'utilisateur sont automatiquement rappelées et insérées dans les champs en guise de suggestions. Pour désactiver cette possibilité, placez l'attribut à `off`. Voici comment activer (`on`) la saisie semi-automatique pour un formulaire entier et la désactiver pour un champ particulier (en gras) :

```
<form action='monform.php' method='post' autocomplete='on'>
  <input type='text'     name='nomutilisateur'>
  <input type='password' name='motdepasse' autocomplete='off'>
</form>
```

Attribut autofocus

L'attribut `autofocus` dirige immédiatement la cible de saisie (*focus*) vers un élément au moment du chargement d'une page. Il s'applique à n'importe quel élément, et un seul, `<input>`, `<textarea>` ou `<button>`, comme suit :

```
<input type='text' name='requete' autofocus='autofocus'>
```

Les navigateurs basés sur une interface tactile (comme sous Android, iOS ou Windows Phone) ignorent généralement l'attribut `autofocus` et obligent l'utilisateur à taper sur un champ pour en faire la cible de saisie; sinon, les effets de zoom, de focus et de clavier contextuel que l'`autofocus` génère deviendraient rapidement très gênants.

Comme cette fonctionnalité place la cible de saisie dans un élément d'entrée, la touche *Ret. arr.* ne peut plus servir à l'utilisateur pour retourner à la page web précédente. Cependant, les touches *Alt+Gauche* et *Alt+Droite* permettent encore de se déplacer en arrière et en avant dans l'historique de navigation.

Attribut placeholder

L'attribut `placeholder` (espace réservé) permet de déposer dans un champ d'entrée une suggestion utile pour expliquer aux utilisateurs ce qu'ils doivent entrer dans ce champ. Il s'utilise comme suit :

```
<input type='text' name='nomprenom' size='50' placeholder='Nom et prénom'>
```

Dans le navigateur, le champ d'entrée affiche la suggestion, qui disparait dès que l'utilisateur tape un caractère dans le champ.

Attribut required

L'attribut `required` (obligatoire) impose à un champ d'avoir reçu des données avant toute soumission du formulaire. Il s'utilise comme suit :

```
<input type='text' name='cartedecredit' required='required'>
```

Quand le navigateur détecte une tentative d'envoi du formulaire alors qu'un champ obligatoire n'a reçu aucune entrée, il affiche un message et invite l'utilisateur à remplir ce champ.

Redéfinir des attributs

La redéfinition (*override*) d'attributs permet de remplacer des réglages du formulaire au niveau des éléments. Ainsi, si l'attribut `action` du formulaire permet d'indiquer au bouton de soumission d'envoyer le formulaire à un programme déterminé dans un champ au sein du formulaire, vous pouvez indiquer, avec `formaction`, que le bouton de soumission doit envoyer le formulaire à une URL différente de celle du formulaire. Dans l'exemple suivant, les actions par défaut et redéfinie sont présentées en gras :

```
<form action='url1.php' method='post'>
  <input type='text' name='champ'>
  <input type='submit' formaction='url2.php'>
</form>
```

HTML5 prend également en charge d'autres redéfinitions d'attributs, tels que `formenctype` (type d'encodage du formulaire), `formmethod` (méthode de formulaire), `formnovalidate` (pas de validation), et `formtarget` (cible du formulaire), qui s'utilisent exactement de la même manière que dans le cas de `formaction`, pour remplacer les réglages correspondants.

Les principaux navigateurs prennent en charge les redéfinitions d'attributs de formulaire depuis longtemps déjà, mais Internet Explorer ne les accepte que depuis sa version 10.

Attributs width et height

Avec les attributs `width` (largeur) et `height` (hauteur), vous pouvez désormais modifier les dimensions d'une image entrée, comme suit :

```
<input type='image' src='photo.png' width='120' height='80'>
```

Fonctionnalités en attente de mise en application complète

Comme la mise en place de HTML5 en est encore à ses débuts (même s'il est là déjà depuis quelques années), les développeurs de navigateurs mettent progressivement en œuvre les fonctionnalités selon leurs propres plannings. Par conséquent, de nombreuses portions de la norme ne sont encore disponibles que sur quelques navigateurs. Au cours de la durée de vie de cette édition du livre, de plus en plus de ces fonctionnalités deviendront accessibles, donc cela vaut la peine de mentionner celles que vous allez voir arriver afin de les exploiter dès qu'elles seront là.

Attribut form

En HTML5, il n'est plus nécessaire de placer les éléments `<input>` au sein d'éléments `<form>`, car vous pouvez préciser le formulaire auquel correspond une entrée à l'aide de l'attribut `form`. Le code suivant illustre la création d'un formulaire, mais avec son entrée déclarée en dehors des balises `<form>` et `</form>` :

```
<form action='monscript.php' method='post' id='form1'>
</form>

<input type='text' name='nomutilisateur' form='form1'>
```

Pour que cela fonctionne, vous devez donner un identifiant au formulaire à l'aide de l'attribut `id`. Et c'est cet identifiant que vous utilisez ensuite dans l'attribut `form` du champ pour désigner le formulaire auquel le champ se réfère.

Au moment de l'écriture de ces lignes, cet attribut n'est pas pris en charge par Internet Explorer.

Attribut list

HTML5 prend en charge l'association de listes à des entrées pour permettre aux utilisateurs de sélectionner facilement des choix à partir de listes prédéfinies. Au moment de la rédaction de cet ouvrage, seuls Firefox, Chrome, Safari et Internet Explorer prennent en charge l'attribut `list`. Quoi qu'il en soit, lorsqu'Opera le mettra en œuvre, vous disposerez d'un outil bien pratique, que vous pourrez utiliser comme suit :

```
Sélectionnez le moteur de recherche :
<input type='url' name='site' list='liens'>

<datalist id='liens'>
  <option label='Google' value='https://www.google.fr'>
  <option label='Yahoo!' value='http://yahoo.fr'>
  <option label='Bing'   value='http://bing.com'>
  <option label='Ask'    value='http://ask.com'>
</datalist>
```

Attributs min et max

Les attributs `min` et `max` permettent d'imposer des valeurs minimale et maximale pour certaines entrées, mais pour le moment, pas dans Firefox ni dans Internet Explorer. Utilisez ces attributs comme suit :

```
<input type='time' name='alarme' value='07:00' min='05:00' max='09:00'>
```

Le navigateur propose, selon son type, un sélecteur haut-bas pour la plage de valeurs autorisées, ou interdit simplement les valeurs en dehors de la plage. Au cours des tests, j'ai trouvé cet attribut un peu flou dans certaines implantations donc je vous suggère de vous livrer à des essais approfondis avant de mettre en œuvre cette fonctionnalité, même quand elle sera disponible sur tous les navigateurs.

Attribut step

Souvent utilisé conjointement avec `min` et `max`, l'attribut `step` permet d'imposer des paliers parmi des nombres ou des dates, comme ceci :

```
<input type='time' name='reunion' value='12:00'
  min='09:00' max='16:00' step='3600'>
```

Lorsque les paliers concernent des dates ou des heures, l'unité représente une seconde. Donc, dans l'exemple ci-dessus, un palier de 3600 secondes représente une heure. Firefox et Internet Explorer ne prennent pas en charge cet attribut.

Type d'entrée color

Le type d'entrée `color` affiche un sélecteur de couleurs qui permet à l'utilisateur de choisir une couleur. Utilisez-le comme suit :

```
Choisissez une couleur <input type='color' name='couleur'>
```

Au moment de la rédaction de ce livre, Firefox et Internet Explorer ne prennent pas en charge ce type d'entrée.

Types d'entrée number et range

Les types d'entrées `number` et `range` visent à restreindre les entrées soit à un nombre, soit à un nombre parmi une plage de valeurs possibles, comme ceci :

```
<input type='number' name='age'>
<input type='range' name='num' min='0' max='100' value='50' step='1'>
```

Firefox semble ne pas prendre en charge le type d'entrée `number` au moment de la rédaction de cet ouvrage.

Sélecteurs de date et heure

Dans les navigateurs qui les prennent en charge, lorsque vous définissez un type d'entrée `date`, `month`, `week`, `time`, `datetime` ou `datetime-local`, un sélecteur de date (calendrier) apparait, dans lequel l'utilisateur peut effectuer sa sélection, comme le suivant, qui entre une heure :

```
<input type='time' name='heure' value='12:34'>
```

Cependant, comme ces sélecteurs ne sont pas encore pris en charge par Internet Explorer et Firefox, il vaut mieux ne pas y faire appel pour l'instant dans vos pages web.

D'autres améliorations relatives aux formulaires en HTML5 sont encore en cours de développement, que vous pouvez suivre sur *http://tinyurl.com/h5forms* (ou sur *http://www.w3.org/TR/html5/forms.html*, son URL directe).

Le chapitre suivant montre comment exploiter les *cookies* et l'authentification pour conserver une trace des préférences des utilisateurs et maintenir leur connexion, ainsi que comment maintenir une session d'utilisateur complète.

Questions

1. Il est possible de soumettre des données de formulaire avec la méthode, soit post, soit Get. Quel est le tableau associatif PHP correspondant à chacune de ces méthodes ?
2. Qu'est-ce que `registrer_globals` et pourquoi est-il préférable de ne pas l'utiliser ?
3. Quelle est la différence entre une zone de texte et une zone de texte étendue ?
4. Si un formulaire doit proposer trois choix à l'utilisateur, mutuellement exclusifs, de sorte que l'utilisateur ne puisse sélectionner qu'un seul parmi les trois, quel type d'entrée utilisez-vous, entre les cases à cocher et le bouton d'options ?
5. Comment pouvez-vous soumettre un groupe de sélections d'un formulaire web à l'aide d'un seul nom de champ ?
6. Quel type de champ de formulaire permet d'insérer une entrée, sans qu'elle soit visible dans le navigateur ?
7. Quelles balises HTML permettent d'encapsuler un élément de formulaire et son texte explicatif ou un graphisme, pour que la totalité de cette unité soit sélectionnée d'un seul clic de souris ?
8. Quelle fonction de PHP convertit du HTML dans une forme telle qu'un navigateur puisse l'afficher mais pas l'interpréter comme du code HTML ?
9. Quel attribut de formulaire permet de suggérer des informations à l'utilisateur pour remplir un champ ?
10. Comment pouvez-vous garantir qu'un champ de formulaire soit rempli, avant que l'utilisateur puisse soumettre le formulaire ?

Retrouvez les réponses du chapitre 11 dans l'annexe A.

CHAPITRE 12

Cookies, sessions et authentification

Vos projets web prennent de plus en plus d'ampleur et vous ressentez la nécessité de conserver une trace de vos utilisateurs. Même si vous n'empruntez pas la voie des informations de connexion et de mots de passe, il est souvent nécessaire de mémoriser des informations à propos de la session courante d'un utilisateur et éventuellement de le reconnaitre lorsqu'il revient sur votre site.

Plusieurs technologies permettent de prendre en compte ce genre d'interactions, qui vont des simples témoins de connexion, ou *cookies*, à la gestion de session, en passant par l'authentification HTTP. Combinées, ces technologies vous donnent la possibilité de configurer un site selon les préférences des utilisateurs et d'assurer une transition agréable et en douceur parmi les pages.

Utiliser les cookies en PHP

Nous utiliserons ici le terme plus familier *cookies* au lieu de celui plus littéraire *témoin de connexion*, pour deux raisons : d'abord c'est un terme du jargon du métier (cherchez « témoin de connexion » dans la documentation, juste pour vous en convaincre) ; ensuite, pour bien l'isoler de l'authentification et des sessions, que nous verrons plus loin dans ce chapitre.

Un *cookie* est une donnée qu'un site web enregistre sur le disque dur de l'ordinateur du visiteur par l'entremise de son navigateur web. Il contient à peu près n'importe quelle information alphanumérique (tant que sa longueur demeure inférieure à 4 Ko), qui peut être récupérée sur l'ordinateur du client et renvoyée au serveur. Les utilisations habituelles portent sur le suivi de session, la conservation de données d'une visite à la suivante, la mémorisation du contenu d'un panier d'achat, le stockage d'informations de connexion et d'autres encore.

Du fait des implications au niveau de la vie privée, seul le domaine qui a créé un cookie peut le relire. En d'autres termes, si le site qui a implanté un cookie était, par exemple, *goulet.ca*, seul un serveur web de ce nom de domaine pourrait récupérer ce cookie. Ceci vise à empêcher d'autres sites web d'accéder aux informations stockées, alors qu'ils n'y sont pas autorisés.

De la manière dont l'internet fonctionne, plusieurs éléments d'une page web peuvent provenir de plusieurs domaines qui génèrent chacun leurs propres cookies. Lorsque cela se produit, ce sont des *cookies de tierce partie*, ou *cookies tiers*. Généralement, ils sont issus de sociétés de publicité et de marketing, et destinés à suivre à la trace des utilisateurs entre plusieurs sites web.

De ce fait, la plupart des navigateurs permettent aux utilisateurs de désactiver les cookies pour le domaine du serveur courant, de serveurs tiers ou des deux. Heureusement, la plupart des gens qui désactivent les cookies le font surtout à propos des cookies des sites web de tiers.

Remarquez que les législations diffèrent selon les pays et certaines législations, en France notamment, imposent d'obtenir l'accord de l'utilisateur pour utiliser des cookies sur son ordinateur. Dans ce cas, l'utilisateur doit cliquer sur un lien (ou un bouton) du genre « J'accepte » pour poursuivre l'utilisation des cookies et la navigation sur le site. Ce genre de message d'avertissement peut apparaître dans un bandeau surgissant dans une des premières pages web, ou constituer une page web à part entière, intermédiaire.

Les cookies s'échangent durant le transfert des entêtes, avant l'envoi du HTML réel d'une page web et il est possible de n'envoyer un cookie que lorsque du HTML a été transféré. Par conséquent, la planification soigneuse de l'usage d'un cookie importe beaucoup. La figure 12-1 illustre un dialogue type de demande et de réponse entre un navigateur et un serveur web qui se transmettent des cookies.

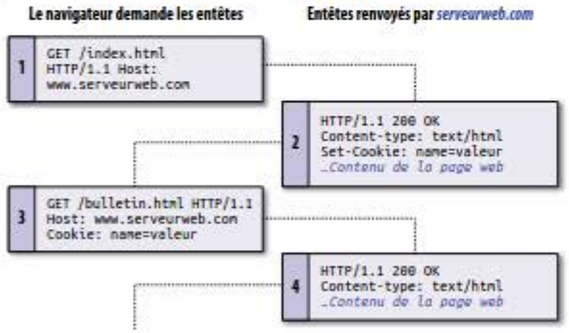


Figure 12-1. Un dialogue demande-réponse entre navigateur et serveur avec des cookies

Cet échange illustre la réception de deux pages par un navigateur :

1. Le navigateur envoie une demande pour obtenir la page principale, *index.html*, sur le site web *http://www.serveurweb.com*. Le premier entête précise le fichier, tandis que le deuxième indique le serveur.

2. Lorsque le serveur, à *serveurweb.com*, reçoit ces deux entêtes, il en renvoie quelques-uns qui lui sont spécifiques. Le deuxième entête définit le type de contenu à envoyer (`text/html`) et le troisième envoie un cookie de nom *name* et de valeur *valeur*. Ce n'est qu'ensuite que le transfert de la page web débute.
3. Dès que le navigateur reçoit le cookie, il le renvoie lors de toute demande à venir de la part de son serveur d'émission, jusqu'à ce que le cookie arrive à expiration ou à sa suppression. Donc, lorsque le navigateur demande la nouvelle page */bulletin.html*, il retourne aussi le cookie *name* avec sa *valeur*.
4. Comme le cookie a déjà été défini, lorsque le serveur reçoit la demande d'envoi de */bulletin.html*, il n'a plus besoin de renvoyer le cookie mais simplement d'envoyer la page demandée.

Définir un cookie

La définition d'un cookie est une opération simple. Tant qu'aucun code HTML n'est transféré, vous pouvez appeler la fonction `setcookie`, qui respecte la syntaxe suivante (tableau 12-1) :

```
setcookie(name, value, expire, path, domain, secure, httponly);
```

Tableau 12-1. Paramètres de la fonction setcookie

Paramètre	Description	Exemple
`name`	Le nom du cookie. C'est le nom utilisé par votre serveur pour accéder au cookie lors des demandes suivantes du navigateur.	`nomutilisateur`
`value`	La valeur du cookie ou son contenu. Le contenu peut atteindre jusqu'à 4 Ko de texte alphanumérique.	`ericdube`
`expire`	(Facultatif.) Un horodatage ou `timestamp` Unix qui précise la date d'expiration. En toute généralité, vous utiliserez la fonction `time()` additionnée d'un nombre de secondes. Si vous ne le précisez pas, le cookie expire à la fermeture du navigateur.	`time() + 2592000`
`path`	(Facultatif.) Le chemin du cookie sur le serveur. S'il contient une barre oblique (/), alors le cookie est disponible sur l'ensemble du domaine, www.serveurweb.com, par exemple. S'il contient un sous-dossier, le cookie n'est valable qu'au sein de ce sous-dossier. Par défaut, il reçoit le dossier courant à partir duquel le cookie a été défini, ce qui correspond à l'usage normal et habituel.	`/`
`domain`	(Facultatif.) Le domaine d'internet du cookie. S'il indique `.serveurweb.com`, alors le cookie est accessible à la totalité de serveurweb.com et à ses sous-domaines, comme images.serveurweb.com. Si le domaine indique images.serveurweb.com, alors le cookie n'est accessible qu'à ce sous-domaine et aux sous-domaines de ce dernier, comme autres.images.serveurweb.com, mais pas à www.serveurweb.com.	`.serveurweb.com`
`secure`	(Facultatif.) Indique si le cookie doit emprunter une connexion sécurisée (https://). Si sa valeur est TRUE, alors l'envoi de cookie doit passer par une connexion sécurisée. La valeur par défaut est FALSE.	FALSE
`httponly`	(Facultatif et mis en œuvre depuis PHP 5.2.0.) Indique si le cookie doit emprunter le protocole HTTP. Si sa valeur est TRUE, les langages de script tels que JavaScript ne peuvent pas accéder au cookie. (Certains navigateurs ne prennent pas cette fonctionnalité en charge.) FALSE est la valeur par défaut.	FALSE

Ainsi, pour créer un cookie de nom `nomutlisateur` et de valeur `ericdube`, accessible sur l'ensemble du serveur web dans le domaine courant, qui soit supprimé de la mémoire cache du navigateur dans sept jours, écrivez ce qui suit :

```
setcookie('nomutilisateur', 'ericdube', time() + 60 * 60 * 24 * 7, '/');
```

Accéder à un cookie

La lecture d'un cookie revient à accéder au tableau système `$_COOKIE`. Ainsi, pour vérifier si le navigateur courant possède déjà un cookie nommé `nomutilisateur` et, si c'est le cas, en lire le contenu, écrivez l'instruction suivante :

```
if (isset($_COOKIE['nomutilisateur'])) $nomutilisateur = $_COOKIE['nomutilisateur'];
```

Remarquez que vous ne pouvez lire un cookie qu'après qu'il a été envoyé au navigateur web. Cela signifie que lorsque vous envoyez un cookie, vous ne pouvez pas le relire tant que le navigateur n'a pas rechargé la page (ou une autre, capable d'accéder au cookie) à partir de votre site web, pour renvoyer de ce fait le cookie au serveur pendant ce processus.

Éliminer un cookie

Pour supprimer un cookie, envoyez-le de nouveau, mais avec une date d'expiration située dans le passé. Pour que la suppression fonctionne, il est très important que tous les paramètres de l'appel soient identiques à ceux de la première définition du cookie, à l'exception de l'horodatage. Par conséquent, pour supprimer le cookie créé précédemment, écrivez par exemple :

```
setcookie('nomutilisateur', 'ericdube', time() - 2592000, '/');
```

Du moment que l'heure indiquée se situe dans le passé, le cookie parvient à expiration et le navigateur le supprime. Notez que la valeur de 2 592 000 secondes correspond à un mois et tient compte du risque que l'ordinateur du visiteur ne soit pas réglé à l'heure exacte.

Authentification HTTP

L'authentification HTTP fait appel au serveur web pour gérer les utilisateurs et leurs mots de passe pour une application. Elle s'avère adéquate pour la plupart des applications qui demandent aux utilisateurs de s'identifier, tandis que certaines applications spécialisées nécessitent des règles de sécurité plus strictes, qui imposent de faire appel à d'autres techniques.

Pour utiliser l'authentification HTTP, PHP envoie une requête d'entête qui demande de démarrer un dialogue d'authentification au sein du navigateur. Cette fonctionnalité doit être activée sur le serveur pour que cela fonctionne mais, comme il s'agit d'une procédure habituelle, votre serveur offre très probablement cette possibilité.

Bien qu'elle soit généralement installée en même temps qu'Apache, l'authentification HTTP n'est pas nécessairement installée sur le serveur que vous utilisez. Par conséquent, il se peut qu'à l'exécution, les quelques exemples qui suivent provoquent des erreurs indiquant que cette fonctionnalité n'est pas installée. Dans ce cas, vous devez installer le module, modifier le fichier de configuration pour charger ce module ou demander à votre administrateur système de régler le problème.

Après l'entrée de votre URL dans un navigateur ou avoir suivi un lien qui y mène, la boite de dialogue « Authentification requise » apparait et demande de préciser deux champs : *Utilisateur* et *Mot de passe* (la figure 12-2 montre l'apparence de cette boite dans Firefox).

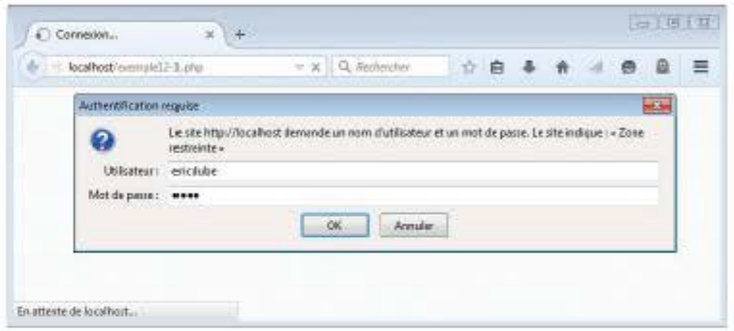


Figure 12-2. Une invite d'identification à l'aide de l'authentification HTTP

L'exemple 12-1 détaille le code qui permet d'aboutir à ce résultat.

Exemple 12-1. Authentification HTTP

```
<?php
  if (isset($_SERVER['PHP_AUTH_USER']) &&
          isset($_SERVER['PHP_AUTH_PW']))
  {
    echo "Bienvenue,<br>Utilisateur : " . $_SERVER['PHP_AUTH_USER'] .
         "<br>Mot de passe : "    . $_SERVER['PHP_AUTH_PW'];
  }
  else
  {
    header('WWW-Authenticate: Basic realm="Restricted Section"');
    header('HTTP/1.0 401 Unauthorized');
    die("Entrez votre nom d'utilisateur et votre mot de passe.");
  }
?>
```

Le serveur Zend utilise par défaut le type d'interface *cgi-fcgi*, qui s'avère incompatible avec l'authentification de base. Or, comme la configuration de Zend échappe à la portée de cet ouvrage, si vous souhaitez essayer les exemples 12-1 à 12-5, il est préférable de le faire sur un autre type de serveur. Le serveur Apache fourni avec XAMPP fonctionne très bien, en revanche. Pour connaitre le type d'interface d'un serveur, vous pouvez appeler la fonction `php_sapi_name`, qui renvoie une chaine du genre `'cgi-fcgi'`, `'cli'` et ainsi de suite. L'utilisation de l'authentification de base n'est toutefois pas conseillée sur un serveur de production car elle est très peu sécuritaire, mais vous devez savoir comment elle fonctionne au cas où vous seriez amené à maintenir du code ancien. Pour plus d'information, consultez *http://php.net/manual/fr/function.php-sapi-name.php*.

En tout premier lieu, le programme vérifie la présence de deux valeurs particulières d'un tableau : `$_SERVER['PHP_AUTH_USER']` et `$_SERVER['PHP_AUTH_PW']`. Si elles existent, alors elles représentent respectivement le nom d'utilisateur et le mot de passe entrés par l'utilisateur dans l'invite d'authentification.

Si une de ces valeurs n'existe pas, alors l'utilisateur ne s'est pas encore identifié et vous affichez l'invite sous la forme d'une boite de dialogue telle que celle de la figure 12-2, par l'envoi de l'entête suivant, où `Basic realm` correspond à la section protégée et qui apparait dans la boite de dialogue :

```
WWW-Authenticate: Basic realm="Zone restreinte"
```

Si l'utilisateur remplit les champs, le programme PHP s'exécute de nouveau depuis le début. Mais si l'utilisateur clique sur *Annuler*, le programme poursuit l'exécution des deux lignes suivantes, qui envoient l'entête suivant et un message d'erreur :

```
HTTP/1.0 401 Unauthorized
```

L'instruction `die` provoque l'affichage du texte « Entrez votre nom d'utilisateur et votre mot de passe pour vous identifier » (figure 12-3).

Figure 12-3. Résultat du clic sur Annuler

Lorsque l'utilisateur s'est identifié, il n'est pas évident d'afficher à nouveau la boite de dialogue d'invite d'authentification, sans fermer toutes les fenêtres liées à ce site dans le navigateur, parce que le navigateur renvoie les mêmes nom d'utilisateur et mot de passe au code PHP du serveur. Il suffit en fait de copier l'URL de l'exemple dans la barre d'adresse du navigateur, d'ouvrir un nouvel onglet, de fermer l'ancien onglet, puis de coller l'URL précédente, éventuellement modifié pour respecter les noms de fichiers des exemples suivants, puis d'appuyer sur *Entrée* pour repartir à zéro.

Tout cela est bien joli mais n'effectue aucune vérification de validité des informations d'identification. L'étape suivante consiste à vérifier leur validité. Le code de l'exemple 12-2 n'ajoute pas grand-chose au précédent, sauf qu'il modifie le code d'affichage du message d'accueil pour vérifier que le nom d'utilisateur et le mot de passe sont corrects, puis il affiche un autre message de bienvenue. Un échec d'authentification entraine l'affichage d'un message d'erreur.

Exemple 12-2. Authentification en PHP avec vérification de validité

```
<?php
  $username = 'admin';
  $password = 'laissezmoientrer';

  if (isset($_SERVER['PHP_AUTH_USER']) &&
      isset($_SERVER['PHP_AUTH_PW']))
  {
    if ($_SERVER['PHP_AUTH_USER'] == $username &&
        $_SERVER['PHP_AUTH_PW']   == $password)
          echo "Vous êtes connecté";
    else die("Nom d'utilisateur ou mot de passe non valide");
  }
  else
  {
    header('WWW-Authenticate: Basic realm="Zone restreinte"');
    header('HTTP/1.0 401 Unauthorized');
    die ("Entrez votre nom d'utilisateur et votre mot de passe pour vous identifier");
  }
?>
```

En corolaire, examinez la manière choisie pour présenter le message d'erreur : « Nom d'utilisateur ou mot de passe non valide ». Cette façon de présenter les choses n'indique pas que c'est le nom d'utilisateur qui n'est pas valide, ni que c'est précisément le mot de passe, mais que c'est l'un ou l'autre, ou les deux. Moins vous fournissez d'informations aux pirates, mieux vous protégez vos mécanismes de sécurité.

Vous disposez à présent d'un mécanisme d'identification de vos utilisateurs, mais seulement pour une paire de nom d'utilisateur et de mot de passe. De plus, le mot de passe apparait en clair (c'est-à-dire en texte tel quel) dans le fichier PHP. Si quelqu'un réussit à s'introduire dans votre serveur, il connait donc instantanément ces informations. Examinons une autre manière de stocker les informations d'identification.

Stocker des noms d'utilisateurs et des mots de passe

Bien entendu, le premier réflexe consiste à stocker naturellement les informations d'identification dans une base de données, MySQL en l'occurrence. Mais encore une fois, il ne faut pas stocker ce genre d'informations sensibles en texte clair, parce qu'un site web peut parfaitement s'avérer compromis si l'attaque d'un pirate permet à celui-ci d'accéder à la base de données. À la place, nous allons utiliser une astuce qui s'appelle une *fonction à sens unique*.

Ce genre de fonction est d'un usage simple et convertit une chaine de texte en une chaine apparemment complètement aléatoire. Par leur nature à sens unique, l'inversion de ces fonctions est quasi impossible, donc leur résultat peut être stocké sans danger dans une base de données. Et quiconque parvient à le voler sera incapable de deviner le mot de passe réel.

L'algorithme de hachage *md5* pour la sécurité de vos données est considéré de nos jours comme facilement piratable, donc non sécuritaire. Et même les suggestions précédentes de remplacement par *sha1* perdent de leur intérêt, puisque cet algorithme a apparemment déjà été brisé (cassé, dans le jargon) par des pirates (d'autant plus que le *sha1* et le *sha2* ont été conçus par l'agence de sécurité américaine, la NSA, ce qui suggère une certaine prudence dans leur utilisation pour des installations de haute sécurité).

J'ai donc choisi d'évoluer vers l'utilisation de la fonction de hachage PHP `hash` et de lui passer une version de l'algorithme *ripemd*, conçu par la communauté académique libre et qui, comme *md5*, renvoie un nombre hexadécimal de 32 caractères, ce qui lui permet de remplacer facilement du *md5* dans la plupart des bases de données. Elle s'utilise comme suit :

```
$jeton = hash('ripemd128', 'unMotDePasseEnClair');
```

Le jeton, ou *token*, c'est-à-dire le mot de passe issu de la « moulinette » de la fonction de hachage indiquée donne ce qui suit :

```
1e1b590515f85d597cdac97ef726e820
```

L'utilisation de la fonction `hash` permet de vous adapter aisément aux futurs développements en matière de sécurité et d'indiquer l'algorithme de hachage que vous souhaitez mettre en œuvre, ce qui réduit implicitement le travail nécessaire pour la maintenance du code (même si vous devez éventuellement vous adapter à des longueurs de résultats de hachage supérieures à 32 bits et modifier vos bases de données).

Salage en cours

Sur un terrain glissant, en hiver, rien de tel qu'un bon salage. Malheureusement, `hash` ne suffit pas pour protéger une base de données de mots de passe, car elle est toujours susceptible de subir une attaque, exhaustive, qui utilise par exemple une autre base de données connue de jetons hexadécimaux de 32 caractères. De telles bases de données existent et, pour vous en convaincre, il suffit de les rechercher sur Google ; quoiqu'actuellement, elles n'existent encore probablement que pour *md5*, *sha1* et *sha2*.

Par chance, il existe une technique d'épandage (l'analogie est particulièrement adaptée) qui permet de contrer les risques, par un *salage* en règle des mots de passe avant de les passer à la moulinette de `hash`. Le salage évoque la notion d'insertion d'un texte que vous êtes le seul à connaitre, dans chaque texte à chiffrer, comme suit (la chaine de salage est mise en évidence en gras) :

```
$jeton = hash('ripemd128', 'salageunmotdepasse');
```

Cet exemple montre la chaine de `salage` insérée en préfixe du mot de passe. Bien entendu, libre à vous d'imaginer une chaine de salage aussi obscure que possible et son emplacement dans la chaine originale. J'aime bien employer un peu de poivre aussi, dans le genre :

```
$jeton = hash('ripemd128', 'hqb%$tunmotdepassecg*l');
```

Avec des caractères aléatoires comme ceux-ci, placés avant et après les mots de passe, si l'attaque vise seulement la base de données, sans possibilité de voir le code PHP associé, il devient quasi impossible de deviner les mots de passe stockés.

Au moment de vérifier la validité d'un mot de passe d'identification d'un utilisateur, il vous reste à ajouter les mêmes chaines aléatoires, avant et après le mot de passe, et de vérifier le jeton résultant par un appel à `hash` pour comparer le tout au contenu de la base de données de cet utilisateur.

Créons à présent une table MySQL pour contenir quelques informations sur des utilisateurs et ajoutons-y quelques comptes. Entrez et enregistrez le programme de l'exemple 12-3 sous le nom de fichier *setupusers.php* pour définir et préciser les utilisateurs.

Exemple 12-3. Création de la table utilisateurs et ajout de deux comptes

```
<?php //definitionutili sateurs.php
  require_once 'login.php';
  // Rappel : $hn = nom d'hôte, $un = nom d'utilisateur,
  //          $pw = mdp, $db = base de données
  $connexion = new mysqli($hn, $un, $pw, $db);

  if ($connexion->connect_error) die($connexion->connect_error);

  $query = "CREATE TABLE utilisateurs (
    prenom     VARCHAR(32) NOT NULL,
    nomfamille VARCHAR(32) NOT NULL,
    nomutil    VARCHAR(32) NOT NULL UNIQUE,
    motdepasse VARCHAR(32) NOT NULL
  ) CHARSET utf8";
  $result = $connexion->query($query);
  if (!$result) die($connexion->error);

  $sel1    = "qm&h*";
  $sel2    = "pg!@";
```

```
  $prenom = 'Bertrand';
  $nomfamille  = 'Simard';
  $nomutil = 'bsimard';
  $motdepasse = 'monsecret';
  $jeton    = hash('ripemd128', "$sel1$motdepasse$sel2");

  ajouter_utilisateur($connexion, $prenom, $nomfamille, $nomutil, $jeton);

  $prenom = 'Pauline';
  $nomfamille  = 'Leduc';
  $nomutil = 'pleduc';
  $motdepasse = 'acrobate';
  $jeton    = hash('ripemd128', "$sel1$motdepasse$sel2");

  ajouter_utilisateur($connexion, $prenom, $nomfamille, $nomutil, $jeton);

  function ajouter_utilisateur($connexion, $fn, $sn, $un, $pw)
  {
    $query  = "INSERT INTO utilisateurs VALUES('$fn', '$sn', '$un', '$pw')";
    $result = $connexion->query($query);
    if (!$result) die($connexion->error);
  }
?>
```

Ce script crée la table *utilisateurs* dans la base de données *publications* (ou la base de données dont vous avez défini l'accès dans le script *login.php* au chapitre 10). Dans ce tableau, nous créons deux utilisateurs : Bertrand Simard et Pauline Leduc. Ils ont chacun un nom d'utilisateur et un mot de passe, respectivement *bsimard*/*monsecret* et *pleduc*/*acrobate*.

À partir de ces données dans la table, nous pouvons modifier l'exemple 12-2 pour identifier correctement les utilisateurs afin d'aboutir à l'exemple 12-4 qui détaille le code nécessaire pour y parvenir. Entrez-le et enregistrez ce fichier sous le nom *authentification.php*, puis appelez-le dans votre navigateur.

Exemple 12-4. Authentification PHP avec MySQL

```
<?php // authentification.php
  require_once 'login.php';
  $connexion = new mysqli($hn, $un, $pw, $db);

  if ($connexion->connect_error) die($connexion->connect_error);

  if (isset($_SERVER['PHP_AUTH_USER']) &&
      isset($_SERVER['PHP_AUTH_PW']))
  {
    $nu_temp = mysql_entities_fix_string($connexion, $_SERVER['PHP_AUTH_USER']);
    $mp_temp = mysql_entities_fix_string($connexion, $_SERVER['PHP_AUTH_PW']);
```



```
    $query  = "SELECT * FROM utilisateurs WHERE nomutil='$nu_temp'";
    $result = $connexion->query($query);
    if (!$result) die($connexion->error);
    elseif ($result->num_rows)
    {
        $row = $result->fetch_array(MYSQLI_NUM);

                $result->close();

        $sel1 = "qm&h*";
        $sel2 = "pg!@";
        $jeton = hash('ripemd128', "$sel1$mp_temp$sel2");

        if ($jeton == $row[3]) echo "$row[0] $row[1] :
        Bonjour $row[0], vous êtes connecté en tant que '$row[2]'";
        else die("Nom d'utilisateur ou mot de passe non valide");
    }
    else die("Nom d'utilisateur ou mot de passe non valide");
  }
  else
  {
    header('WWW-Authenticate: Basic realm="Zone restreinte"');
    header('HTTP/1.0 401 Unauthorized');
    die ("Entrez votre nom d'utilisateur et votre mot de passe pour vous identifier");
  }

  $connexion->close();

  function mysql_entities_fix_string($connexion, $string)
  {
    return htmlentities(mysql_fix_string($connexion, $string));
  }

  function mysql_fix_string($connexion, $string)
  {
    if (get_magic_quotes_gpc()) $string = stripslashes($string);
    return $connexion->real_escape_string($string);
  }
?>
```

Comme vous pouvez le constater, à ce stade, certains exemples commencent à prendre une certaine ampleur. Mais ne vous laissez pas impressionner. Les dix dernières lignes proviennent de l'exemple 10-22 du chapitre 10. Elles servent à aseptiser les entrées de l'utilisateur et demeurent très importantes.

Mis à part cela, les lignes qui vous importent le plus débutent à l'affectation aux variables `$nu_temp` et `$mp_temp` du nom d'utilisateur et du mot de passe reçus, mises en évidence en gras dans le programme. Ensuite, une requête envoyée à MySQL recherche l'utilisateur `$nu_temp` et, s'il est trouvé, la première ligne de données renvoyée est

affectée dans `$row` (et, comme les noms d'utilisateurs sont a priori uniques, il n'y a qu'une seule ligne). Nous créons ensuite nos deux pincées de sel, `$sel1` et `$sel2`, que nous ajoutons avant et après le mot de passe soumis, `$mp_temp`. Nous passons ensuite la chaine à la fonction `hash`, qui renvoie une valeur hexadécimale de 32 caractères dans `$jeton`.

L'étape suivante consiste à comparer le contenu de `$jeton` à la valeur stockée dans la base de données, qui se trouve dans la quatrième colonne, soit dans la colonne 3 en commençant par 0 : `$row[3]` contient le jeton calculé précédemment à partir du mot de passe généré et salé. Si les deux valeurs sont identiques, nous affichons un message d'accueil convivial qui appelle l'utilisateur par son prénom (figure 12-4). Dans le cas contraire, nous affichons un message d'erreur. Comme évoqué précédemment, le message est identique que l'utilisateur existe ou non, que le mot de passe soit ou ne soit pas correct, de façon à laisser le moins d'informations possible aux pirates et d'outils de devinette de mots de passe éventuels.

Figure 12-4. L'utilisateur Bertrand Simard s'est identifié

Pour essayer ce code, entrez-le, enregistrez-le et ouvrez le script dans le navigateur. Entrez le nom d'utilisateur `bsimard` et son mot de passe, `monsecret` (ou `pleduc` et `acrobate`), soit les valeurs que nous avons enregistrées dans la table utilisateurs à l'exemple 12-3.

L'aseptisation immédiate des entrées après leur saisie permet de bloquer instantanément toute attaque par injection de code HTML, MySQL ou JavaScript, de sorte qu'il n'est plus nécessaire de nettoyer de nouveau ces données par la suite. Rappelez-vous toutefois que si un utilisateur place des caractères, par exemple, < ou & dans son mot de passe, ils sont convertis respectivement en `<` ou `&` par la fonction `htmlentities`. Mais cela n'a aucune importance car tout ira bien tant que vous prévoyez une longueur de chaine dans la base de données supérieure à la longueur du champ d'entrée de mot de passe dans les formulaires, et à condition que vous passiez toujours ces mots de passe à la même moulinette d'aseptisation.

Utiliser les sessions

Du fait qu'un programme ne peut savoir quelles variables d'autres programmes ont définies, ni même les valeurs que le même programme a définies lors de sa dernière exécution, il est parfois nécessaire de suivre ce que les utilisateurs font, d'une page web à une autre. La définition de champs masqués, détaillés au chapitre 10, permet de récupérer des variables lors de la soumission d'un formulaire, mais PHP fournit une autre solution, plus simple et nettement plus puissante, avec la notion de *sessions*. Ce sont des groupes de variables stockées sur le serveur mais associées uniquement à un utilisateur courant. Pour garantir que les bonnes variables soient associées avec le bon utilisateur, PHP enregistre un *cookie* dit *de session* dans le navigateur web des utilisateurs pour les identifier de manière unique.

Ce cookie n'a de sens que pour le serveur web et ne peut servir à déterminer la moindre information sur l'utilisateur, ce qui lui donne un net avantage de confidentialité par rapport aux cookies (tiers) classiques. Vous vous inquiétez probablement de ce qui se produit chez les utilisateurs dont l'acceptation des cookies est bloquée, comme c'est de plus en plus souvent le cas. En pratique, depuis sa version 4.2.0, ce n'est plus un problème parce que PHP identifie ce genre de cas et, s'il y a lieu, il place un jeton de cookie dans la portion Get de chaque demande d'URL. Par conséquent, les sessions fournissent un moyen efficace pour suivre vos utilisateurs.

Débuter une session

Le démarrage d'une session nécessite d'appeler la fonction PHP `session_start` avant tout envoi de HTML, de la même manière que les cookies sont envoyés durant l'échange des entêtes. Ensuite, pour commencer à enregistrer des variables de session, affectez-les simplement en tant que cellules du tableau `$_SESSION`, comme suit :

```
$_SESSION['variable'] = $valeur;
```

Pour les lire à mesure que le programme s'exécute ou plus tard, écrivez :

```
$variable = $_SESSION['variable'];
```

À présent, supposons qu'une de vos applications nécessite de manière récurrente l'accès au nom d'un utilisateur, à son mot de passe, à son prénom et son nom, tels qu'ils sont présents dans la table *utilisateurs*, que nous avons créée récemment. Modifions le script *authentification.php* de l'exemple 12-4 pour définir une session dès qu'un utilisateur s'est identifié.

L'exemple 12-5 illustre les modifications apportées. La seule différence réside ici dans le contenu de la section `if ($jeton == $row[3])`, qui débute par l'ouverture d'une session et l'enregistrement des quatre variables dans celle-ci. Entrez le programme (ou modifiez celui de l'exemple 12-4) et enregistrez-le sous le nom *authentification2.php*. Ne l'exécutez pas tout de suite dans votre navigateur, car nous allons d'abord créer un autre programme dans quelques instants.

Exemple 12-5. Mise en place d'une session après une authentification réussie

```
<?php // authentification2.php
  require_once 'login.php';
  $connexion = new mysqli($hn, $un, $pw, $db);

  if ($connexion->connect_error) die($connexion->connect_error);

  if (isset($_SERVER['PHP_AUTH_USER']) &&
      isset($_SERVER['PHP_AUTH_PW']))
  {
    $nu_temp = mysql_entities_fix_string($connexion, $_SERVER['PHP_AUTH_USER']);
    $mp_temp = mysql_entities_fix_string($connexion, $_SERVER['PHP_AUTH_PW']);

    $query  = "SELECT * FROM utilisateurs WHERE nomutil='$nu_temp'";
    $result = $connexion->query($query);
    if (!$result) die($connexion->error);
    elseif ($result->num_rows)
    {
      $row = $result->fetch_array(MYSQLI_NUM);

      $result->close();

      $sel1 = "qn&h*";
      $sel2 = "pg!@";
      $jeton = hash('ripemd128', "$sel1$mp_temp$sel2");

      if ($jeton == $row[3])
      {
        session_start();
        $_SESSION['nomutil']    = $nu_temp;
        $_SESSION['motdepasse'] = $mp_temp;
        $_SESSION['prenom']     = $row[0];
        $_SESSION['nomfamille'] = $row[1];
        echo "$row[0] $row[1] :
        Bonjour $row[0], vous êtes connecté en tant que '$row[2]'";
        die ("<p><a href=continue.php>Cliquez ici pour continuer</a></p>");
      }
      else die("Nom d'utilisateur ou mot de passe non valide");
    }
    else die("Nom d'utilisateur ou mot de passe non valide");
  }
  else
  {
    header('WWW-Authenticate: Basic realm="Zone restreinte"');
    header('HTTP/1.0 401 Unauthorized');
    die ("Entrez votre nom d'utilisateur et votre mot de passe pour vous identifier");
  }
```



```
  $connexion->close();

  function mysql_entities_fix_string($connexion, $string)
  {
    return htmlentities(mysql_fix_string($connexion, $string));
  }

  function mysql_fix_string($connexion, $string)
  {
    if (get_magic_quotes_gpc()) $string = stripslashes($string);
    return $connexion->real_escape_string($string);
  }
?>
```

L'autre ajout à ce programme se situe dans l'affichage du lien « Cliquez ici pour continuer » vers l'URL de destination *continue.php*. Cette variante nous permettra d'illustrer le transfert de la session à un autre programme ou à une autre page web en HTML. Donc créez le fichier *continue.php* à partir de l'exemple 12-6 et enregistrez-le.

Exemple 12-6. Récupération des variables de session dans un autre script

```
<?php // continue.php
  session_start();

  if (isset($_SESSION['nomutil']))
  {

    $nomutil    = $_SESSION['nomutil'];
    $motdepasse = $_SESSION['motdepasse'];
    $prenom     = $_SESSION['prenom'];
    $nomfamille = $_SESSION['nomfamille'];

    echo "Rebonjour, $prenom.<br>
          Votre nom complet est $prenom $nomfamille.<br>
          Votre nom d'utilisateur est '$nomutil'
          et votre mot de passe est '$motdepasse'.";
  }
  else echo "Cliquez <a href='authentification2.php'>ici</a> " .
            "pour vous identifier.";
?>
```

Vous disposez maintenant des deux scripts nécessaires pour le test. Appelez *authentification2.php* dans votre navigateur, entrez le nom d'utilisateur `bsimard` et le mot de passe `monsecret` (ou `pleduc` et `acrobate`) à l'invite, puis cliquez sur le lien pour charger *continue.php*. Lorsque le navigateur appelle ce dernier, il doit afficher quelque chose du genre de ce qu'illustre la figure 12-5.

Figure 12-5. Conservation de données par l'entremise d'une session

Les sessions permettent de confiner superbement à un seul sous-programme le vaste code nécessaire pour authentifier un utilisateur et l'identifier dans l'application dans sa globalité. Dès que l'utilisateur s'est authentifié et que vous avez créé une session, le reste du code devient plus simple. Il suffit d'appeler `session_start` et d'ensuite insérer, puis rechercher, toutes les variables dont vous avez besoin dans `$_SESSION`.

Dans l'exemple 12-6, un rapide test de la présence d'une valeur pour `$_SESSION['nomutil']` permet de s'assurer que l'utilisateur courant s'est bien identifié, car les variables de session sont stockées sur le serveur (à l'inverse des cookies classiques, stockés dans le navigateur). De plus, vous pouvez faire confiance à ces variables, qui ne quittent pas le serveur.

Si `$_SESSION['nomutil']` n'a pas encore reçu de valeur, aucune session n'est active donc vous pouvez en déduire que l'utilisateur n'est pas passé par la case de départ. En conséquence, la dernière ligne de code de l'exemple 12-6 redirige l'utilisateur inconnu vers la page d'identification, *authentification2.php*.

Le programme *continue.php* affiche le mot de passe de l'utilisateur identifié pour démontrer le fonctionnement des variables de session. En pratique, vous savez déjà que l'utilisateur s'est identifié et qu'il est authentifié, mais vous n'afficherez jamais de mots de passe de la sorte. Ne transférez même pas de mots de passe par le biais de cookies classiques. Le faire relèverait d'une imprudence aux conséquences potentiellement terribles.

Clôturer une session

Lorsqu'il est temps de terminer une session, généralement parce que l'utilisateur demande à se déconnecter de votre site, utilisez la fonction `session_destroy` en association avec la suppression du cookie de session (exemple 12-7). Ce script propose un exemple de fonction utile pour détruire totalement une session, déconnecter un utilisateur et supprimer toutes les variables de session.

Exemple 12-7. Fonction utile pour détruire une session et toutes ses données

```
<?php
  // Destruction complète de la session courante et de ses données
  function destroy_session_and_data()
  {
     session_start();  // Si la session n'est pas encore récupérée.
     $_SESSION = array();
     setcookie(session_name(), '', time() - 2592000, '/');
     session_destroy();
  }
?>
```

Pour voir la fonction en action, modifiez *continue.php* comme dans l'exemple 12-8.

Exemple 12-8. Récupérer les variables de session puis détruire la session

```
<?php
  session_start();

  if (isset($_SESSION['nomutil']))
  {
    $nomutil    = $_SESSION['nomutil'];
    $motdepasse = $_SESSION['motdepasse'];
    $prenom     = $_SESSION['prenom'];
    $nomfamille = $_SESSION['nomfamille'];

    destroy_session_and_data();

    echo "Rebonjour, $prenom.<br>
          Votre nom complet est $prenom $nomfamille.<br>
          Votre nom d'utilisateur est '$nomutil'
          et votre mot de passe est '$motdepasse'.";
  }
  else echo "Cliquez <a href='authentification2.php'>ici</a> " .
            "pour vous identifier.";

   function destroy_session_and_data()
  {
    $_SESSION = array();
    setcookie(session_name(), '', time() - 2592000, '/');
    session_destroy();
  }
?>
```

La première fois que vous ouvrez *authentification2.php* pour suivre le lien vers la version modifiée de *continue.php*, ce dernier affiche toutes les variables de session mais, du fait de l'appel à `destroy_session_and_data`, si vous cliquez sur le bouton d'actualisation du navigateur, la session est supprimée et vous êtes invité à retourner à la page d'identification.

Définir une durée limite

Il n'est pas rare de devoir fermer vous-même la session d'un utilisateur, notamment parce que celui-ci a oublié ou négligé de se déconnecter adéquatement, de sorte que le programme doit la clôturer pour la propre sécurité de l'utilisateur. Il est possible alors de préciser un délai limite (ou *time-out*), au-delà duquel une déconnexion automatique a lieu si aucune activité n'est signalée.

Pour cela, utilisez la fonction `ini_set` comme suit. Cet exemple définit un délai limite à exactement un jour, exprimé en secondes :

```
ini_set('session.gc_maxlifetime', 60 * 60 * 24);
```

Si ensuite vous souhaitez connaitre la période de délai courante, vous pouvez l'afficher comme suit :

```
echo ini_get('session.gc_maxlifetime');
```

Rappelez-vous : *set* pour définir, *get* pour lire.

Sécurité de session

Si j'ai évoqué le fait qu'une fois qu'un utilisateur s'est identifié et que vous avez mis en place une session, vous pourriez supposer en toute sécurité que les variables de session sont de confiance, mais en pratique, ce n'est pas exactement le cas. La raison des inquiétudes à leur propos se situe dans l'existence de *renifleurs de paquets* (*packet sniffers*), capables de lire des échantillons de données, pour découvrir les identifiants de session qui transitent par le réseau. De plus, l'identifiant de session est transmis dans la partie Get d'une URL, il peut apparaitre dans les journaux des serveurs de sites externes. La seule possibilité vraiment sécuritaire pour empêcher la découverte de ceux-ci consiste à mettre en place le *protocole SSL* (*Secure sockets layer*), c'est-à-dire de couche de sockets sécurisés, et d'exécuter des requêtes HTTPS au lieu des requêtes de pages web HTTP classiques. Si ce sujet échappe à la portée de ce livre, n'hésitez pas à consulter *http://www.apache-ssl.org/* pour de plus amples informations sur la mise en place d'un serveur web sécurisé.

Prévenir le détournement de session

Lorsque le protocole SSL n'est pas disponible, il demeure possible d'authentifier les utilisateurs par le stockage de leurs adresses IP en plus des autres informations, à l'aide d'une ligne telle que la suivante, lorsque vous transmettez les données de session de page en page :

```
$_SESSION['ip'] = $_SERVER['REMOTE_ADDR'];
```

Cette instruction mémorise dans la variable de session `ip` l'adresse distante de l'ordinateur qui exécute le navigateur de l'utilisateur. Ensuite, en guise de vérification supplémentaire, chaque fois qu'une page se charge et qu'une session est disponible, effectuez le test suivant. Il appelle la fonction `different_user` si l'adresse IP mémorisée ne correspond pas à l'adresse IP courante :

```
if ($_SESSION['ip'] != $_SERVER['REMOTE_ADDR']) different_user();
```


Le code que vous placez dans la fonction `different_user` est entièrement à votre discrétion. Je vous conseille cependant d'y supprimer la session courante et de demander à l'utilisateur de s'identifier à nouveau au prétexte d'un problème technique. Ne dites rien de plus que cela, sinon vous risquez de laisser filtrer des informations potentiellement exploitables contre votre site.

Cela dit, soyez conscient du fait que des utilisateurs situés derrière un serveur mandataire (*proxy*), ou qui partagent la même adresse IP externe, parce qu'ils sont dans un réseau domestique ou de bureau, afficheront la même adresse IP. Dans ce cas aussi, si cela vous pose des problèmes, orientez-vous vers l'utilisation de SSL. D'autres possibilités existent, comme mémoriser une copie de la chaine de l'*agent* navigateur *de l'utilisateur* (*user agent*), c'est-à-dire la chaine que les développeurs glissent dans leurs navigateurs pour les identifier selon leurs types et versions, qui permettent aussi accessoirement de distinguer les utilisateurs, du fait de la vaste diversité des types et versions de navigateurs, ainsi que d'ordinateurs. L'instruction suivante permet de mémoriser l'agent de l'utilisateur :

```
$_SESSION['au'] = $_SERVER['HTTP_USER_AGENT'];
```

Comparez ensuite la chaine de l'agent courant à celle mémorisée :

```
if ($_SESSION['au'] != $_SERVER['HTTP_USER_AGENT']) different_user();
```

Ou mieux, combinez les deux vérifications et enregistrez la combinaison dans une chaine de hachage hexadécimale :

```
$_SESSION['verification'] = hash('ripemd128', $_SERVER['REMOTE_ADDR'] .
    $_SERVER['HTTP_USER_AGENT']);
```

Puis, comparez les chaines courante et mémorisée :

```
if ($_SESSION['verification'] != hash('ripemd128', $_SERVER['REMOTE_ADDR'] .
    $_SERVER['HTTP_USER_AGENT'])) different_user();
```

Prévenir la fixation de session

La *fixation de session* se produit lorsqu'un utilisateur malveillant essaie de présenter un identifiant de session au serveur au lieu de lui laisser la possibilité d'en créer un. Cela se produit lorsqu'un utilisateur profite de la possibilité de passer un identifiant de session dans la partie Get d'une URL, comme suit :

```
http://votreserveur.com/authentification.php?PHPSESSID=123456789
```

Cet exemple montre l'identifiant de session `PHPSESSID` de valeur `123456789` transmis au serveur. À présent, examinez le script de l'exemple 12-9, susceptible de subir une fixation de session. Pour l'essayer, entrez le script et enregistrez-le sous le nom de fichier *sessiontest.php*.

Exemple 12-9. Une session susceptible de subir une fixation de session

```
<?php // sessiontest.php
  session_start();

  if (!isset($_SESSION['compteur'])) $_SESSION['compteur'] = 0;
  else ++$_SESSION['compteur'];

  echo $_SESSION['compteur'];
?>
```

Le programme enregistré, appelez-le à l'aide de l'URL suivante (préfixée du chemin du fichier à partir de la racine des documents, par exemple *http://localhost/web/*) :

```
sessiontest.php?PHPSESSID=1234
```

Cliquez plusieurs fois sur le bouton d'actualisation et vous pouvez voir le compteur augmenter de valeur. Ensuite, entrez l'URL suivante, avec le même chemin de document :

```
sessiontest.php?PHPSESSID=5678
```

Cliquez de nouveau plusieurs fois sur le bouton d'actualisation et vous devriez voir le compteur repartir de 0. Laissez le compteur sur un nombre différent de celui de la première URL et revenez à la première URL, actualisez et le compteur continue sur sa lancée. Vous avez créé deux sessions différentes de votre propre chef et vous pouvez en créer autant que vous voulez.

Le plus grand danger de cette approche réside dans le risque qu'un utilisateur malveillant distribue des URL de ce genre à des utilisateurs non vigilants et innocents, et que si ceux-ci suivent effectivement ces liens, l'agresseur est alors en mesure de revenir sur ces sessions et de prendre le contrôle des sessions non supprimées ou encore en cours de validité!

Pour empêcher cela, ajoutez une simple vérification pour changer l'identifiant de session à l'aide de `session_regenerate`. Cette fonction conserve toutes les valeurs des variables de la session courante mais remplace l'identifiant de session par un nouveau, que l'agresseur ne peut connaitre.

Pour cela, testez la présence d'une variable de session particulière que vous inventez de façon arbitraire. Si elle n'existe pas, vous savez qu'il s'agit d'une nouvelle session, donc changez l'identifiant de session et définissez une variable de session spéciale pour indiquer la modification.

L'exemple 12-10 montre le genre de code qui prend en compte ce type de vérification, à l'aide de la variable `initiated`, qui signifie « initialisé? ».

Exemple 12-10. Régénération de session

```
<?php
  session_start();

  if (!isset($_SESSION['initiated']))
  {
    session_regenerate_id();
    $_SESSION['initiated'] = 1;
  }

  if (!isset($_SESSION['compteur'])) $_SESSION['compteur'] = 0;
  else ++$_SESSION['compteur'];

  echo $_SESSION['compteur'];
?>
```

De cette manière, lorsque l'attaquant revient sur votre site avec un des identifiants de session qu'il a générés lui-même, aucun d'eux ne peut appeler la session d'un autre utilisateur car tous les identifiants licites auront été remplacés par des identifiants régénérés. Si vous faites preuve de paranoïa aigüe, vous pouvez même régénérer les identifiants de session lors de chaque requête de page.

Imposer des sessions uniquement à base de cookies

Si vous vous apprêtez à exiger de vos utilisateurs d'activer les cookies pour accéder à votre site (n'oubliez pas de lui en demander l'autorisation notamment si vos utilisateurs sont en France), utilisez la fonction `ini_set`, comme suit :

```
ini_set('session.use_only_cookies', 1);
```

Avec ce réglage, le souci du `?PHPSESSID=` est complètement éludé. Si vous mettez en place ce dispositif de sécurité, je vous recommande d'informer vos visiteurs que la navigation dans votre site nécessite l'utilisation des cookies pour, d'une part, respecter la législation de certains pays qui exige d'obtenir l'acceptation des visiteurs pour poursuivre la navigation avec des cookies et, d'autre part, pour leur faire comprendre que si leur navigation n'aboutit pas aux résultats prévus, la cause se situe au niveau de ces cookies.

Utiliser un serveur mutualisé

Sur un serveur mutualisé, c'est-à-dire partagé par plusieurs sites web et comptes d'hébergement, veillez à ne pas enregistrer les données de vos sessions dans le même dossier que celles des autres utilisateurs. Pour stocker vos sessions, choisissez plutôt un dossier auquel seul votre compte a accès et qui n'est pas visible à partir du Web. Un appel à la fonction `ini_set` placé à proximité du tout début d'un programme permet de préciser le dossier cible des sessions, comme suit :

```
ini_set('session.save_path', '/home/user/noncompte/sessions');
```

Cette option de configuration ne conserve ce réglage que durant l'exécution du programme et la configuration initiale est automatiquement restaurée à la fin de l'exécution du programme.

Retenez que si vos sites web peuvent subir des tentatives de piratage, ils en subiront ! Des *bots* (robots) automatisés fonctionnent en meutes autour de l'internet et essaient de trouver des sites vulnérables aux *exploits*, c'est-à-dire des programmes permettant d'exploiter les failles de sécurité des sites. Par conséquent, chaque fois que vous manipulez des données qui n'ont pas été générées à 100 % au sein de vos propres programmes, traitez-les avec la plus grande prudence.

À ce stade, vous devez disposer d'une très bonne maitrise, tant de PHP que de MySQL. Le chapitre suivant entame la présentation de la troisième technologie majeure exposée dans ce livre, JavaScript.

Questions

1. Quelle raison impose qu'un cookie soit transféré au début d'un programme ?
2. Quelle fonction de PHP permet de stocker un cookie dans un navigateur web ?
3. Comment faire pour détruire un cookie ?
4. Où sont stockés un nom d'utilisateur et un mot de passe dans un programme PHP, lorsque celui-ci utilise l'authentification HTTP ?
5. En quoi la fonction `hash` représente-t-elle une puissante mesure de sécurité ?
6. Que signifie le « salage » d'une chaine ?
7. Qu'est-ce qu'une session PHP ?
8. Comment initialiser une session ?
9. Qu'est-ce que le piratage de session ?
10. Qu'est-ce que la fixation de session ?

Retrouvez les réponses du chapitre 12 dans l'annexe A.

CHAPITRE 13

Explorer JavaScript

JavaScript apporte des fonctionnalités dynamiques aux sites web. Chaque fois que quelque chose surgit à l'écran à la suite du survol de la souris sur un élément dans le navigateur, ou que vous voyez de nouveaux textes, de nouvelles couleurs et des images apparaitre devant vos yeux sur une page, et que vous glissez un objet sur la page pour le déposer dans un autre endroit, vous voyez JavaScript à l'œuvre. Il produit des effets qu'il serait difficile, voire impossible de réaliser autrement, parce qu'il s'exécute au niveau du navigateur et qu'il dispose d'un accès direct à tous les éléments d'un document web.

JavaScript est apparu pour la première fois dans le navigateur Netscape Navigator, en 1995, en coïncidence avec l'ajout de la prise en charge de la technologie Java dans le navigateur. JavaScript fut longtemps perçu, à tort, comme une variante du langage Java. En réalité, le nom JavaScript naquit d'une manœuvre de marketing pour permettre à ce nouveau langage de script de s'imposer en profitant de l'engouement pour le langage de programmation Java.

JavaScript a bénéficié d'encore plus de puissance lorsque les éléments HTML des pages web ont reçu une définition plus formelle, mieux structurée avec ce que l'on appelle le *modèle objet de document*, ou *DOM* (*Document object model*). Le DOM apporte une relative aisance à l'ajout d'un paragraphe ou au ciblage de saisie ou de lecture sur une portion de texte et à sa modification.

Comme JavaScript et PHP reprennent chacun une part importante de la syntaxe du langage de programmation C, ils se ressemblent très fortement. Ce sont tous deux des langages d'assez haut niveau. Ainsi, comme ils sont faiblement typés, il est facile de modifier une variable en un autre type, simplement en l'utilisant dans un contexte différent.

Comme vous avez appris PHP, vous devriez trouver assez facile l'assimilation de JavaScript. Et vous serez heureux d'apprendre ce dernier, parce qu'il forme le cœur de la technologie du Web 2.0, Ajax, qui fournit (en collaboration avec les fonctionnalités de HTML5) ces « web frontends » fluides et conviviaux que les utilisateurs du Web attendent de nos jours.

JavaScript et le texte HTML

JavaScript est un langage de script du côté client qui s'exécute en totalité au sein d'un navigateur web. Pour appeler ce script, placez le code entre les balises HTML ouvrante `<script>` et fermante `</script>`. L'exemple 13-1 illustre le classique script « Bonjour tout le monde » appliqué dans un document HTML 4.01.

Exemple 13-1. Affichage de « Bonjour tout le monde » en JavaScript

```
<html>
  <head><title>Bonjour tout le monde</title></head>
  <body>
    <script type="text/javascript">
      document.write("Bonjour tout le monde")
    </script>
    <noscript>
      Votre navigateur ne prend pas en charge ou a désactivé JavaScript
    </noscript>
  </body>
</html>
```

Vous avez certainement déjà vu des pages qui utilisent la balise HTML `<script language="javascript">` mais celle-ci est désormais obsolète. Cet exemple utilise la balise plus récente et préférable `<script type="text/javascript">` mais vous pouvez aussi vous contenter simplement de `<script>` si vous le souhaitez.

Entre les balises `<script>` figure une ligne de code JavaScript qui emploie l'équivalent des commandes PHP `echo` et `print`, `document.write`. Comme vous le supposez, elle sort simplement la chaine fournie dans le document courant, où elle s'affiche.

Vous avez peut-être remarqué qu'au contraire de PHP, la ligne de JavaScript ne se termine pas par le point-virgule (;). Ceci est dû au fait qu'en JavaScript, le retour à la ligne produit le même effet que le point-virgule. En revanche, si vous placez plusieurs instructions sur une même ligne, vous devez insérer un point-virgule entre chaque commande, sauf après la dernière instruction. Si vous le souhaitez, vous pouvez toutefois placer systématiquement un point-virgule après chaque instruction, car le code JavaScript fonctionnera tout aussi bien.

Cet exemple illustre encore un point important à remarquer : la paire de balises `<noscript>` et `</noscript>`. Elles servent à entourer du code HTML alternatif destiné aux utilisateurs dont le navigateur ne prend pas en charge JavaScript ou qui l'ont désactivé. L'utilisation de ces balises est à votre entière discrétion, du fait qu'elles ne sont pas obligatoires, mais je vous conseille de les conserver car cela ne demande généralement pas beaucoup de travail que de fournir une alternative en HTML pur aux opérations que vous assurez à l'aide de JavaScript. Les exemples de la suite des chapitres du livre sur JavaScript ne reprendront plus ces balises `<noscript>`, parce qu'ils illustrent ce que JavaScript permet de faire, et non ce que vous pouvez faire sans lui.

Lors du chargement de l'exemple 13-1, un navigateur avec JavaScript activé affiche le texte `Bonjour tout le monde` (figure 13-1).

```
Bonjour tout le monde
```

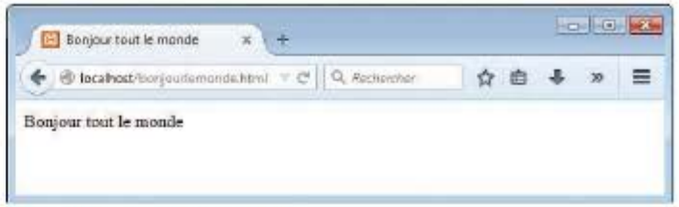


Figure 13-1. JavaScript activé et en action

Un navigateur où JavaScript est désactivé affichera plutôt ce message (figure 13-2) :

```
Votre navigateur ne prend pas en charge ou a désactivé JavaScript
```

Figure 13-2. JavaScript désactivé et les effets résultants

Utiliser des scripts dans la section head d'un document

Outre placer un script dans le corps d'un document, vous pouvez en placer dans la section d'entête `<head>`, auquel cas le script s'exécute au moment du chargement de la page. Si vous placez du code et des fonctions essentielles à cet endroit, vous êtes certain qu'ils sont prêts pour un usage immédiat par les autres sections de script du document qui reposent sur eux.

Un autre intérêt de placer un script dans l'entête de document consiste à permettre à JavaScript d'écrire des choses comme des balises méta dans la section `<head>`, car l'emplacement du script est celui du document où il écrit par défaut.

Navigateurs anciens et hors normes

Si vous devez prendre en charge des navigateurs qui n'offrent pas la possibilité d'intégrer des scripts, vous devez utiliser les balises HTML de commentaire (`<!--` et `-->`) pour empêcher ces navigateurs de rencontrer du code de script qu'ils ne doivent pas voir. L'exemple 13-2 montre comment les ajouter à un code de script.

Exemple 13-2. Exemple de « Bonjour tout le monde » modifié pour les navigateurs sans JavaScript

```
<html>
  <head><title>Bonjour tout le monde</title></head>
  <body>
    <script type="text/javascript"><!--
      document.write("Bonjour tout le monde")
    // --></script>
  </body>
</html>
```

Ici, une balise d'ouverture de commentaire (`<!--`) vient s'ajouter directement après la balise ouvrante `<script>`, et une balise de fermeture de commentaire (`// -->`) juste avant la fermeture du script avec `</script>`.

La double barre oblique (`//`) sert à indiquer en JavaScript que le reste de la ligne est du commentaire. Par conséquent, les navigateurs qui prennent JavaScript en charge ignorent le `-->` suivant, tandis que les navigateurs qui ne prennent pas JavaScript en charge ignorent les `//` qui précèdent le `-->` car, pour eux, ils font partie d'un commentaire HTML.

Si cette solution semble un peu alambiquée, tout ce que vous devez retenir, c'est d'utiliser les deux lignes suivantes pour entourer votre code JavaScript lorsque vous préférez tenir compte des navigateurs anciens ou hors normes :

```
<script type="text/javascript"><!—
  (Votre code JavaScript vient ici...)
// --></script>
```

En revanche, l'usage de ces commentaires devient inutile pour tout navigateur édité ces dernières années.

Vous devez savoir que deux autres langages de script existent. Parmi ceux-ci, il y a VBScript de Microsoft, fondé sur le langage de programmation Visual Basic, ainsi que Tcl, un langage de prototypage rapide. Leur appel se fait d'une manière semblable à celui de JavaScript, sauf qu'ils nécessitent les types d'utilisation `text/vbscript` et `text/tcl`, respectivement. VBScript ne fonctionne que dans Internet Explorer et son utilisation dans d'autres navigateurs nécessite un module d'extension (*plug-in*). Tcl nécessite un module d'extension dans tous les cas. Nous pouvons donc les considérer comme hors standard et ce livre n'examine aucun des deux.

Inclure des fichiers JavaScript

Au lieu d'écrire du code JavaScript directement dans des documents HTML, vous pouvez inclure des fichiers de code JavaScript à partir de votre site web ou de n'importe où sur l'internet. La syntaxe est la suivante :

```
<script type="text/javascript" src="script.js"></script>
```


Ou, pour inclure un fichier de la Toile, utilisez ceci :

```
<script type="text/javascript" src="http://unserveur.com/script.js">
</script>
```

Comme pour les fichiers de scripts eux-mêmes, ils *ne* doivent *pas* inclure de balises `<script>` et `</script>`, car elles ne sont pas nécessaires : le navigateur sait automatiquement qu'il charge un fichier JavaScript. Placer ces balises dans un fichier JavaScript d'extension *.js* provoque une erreur.

L'inclusion de fichiers de script dans vos pages constitue le moyen de choix pour exploiter des fichiers JavaScript de parties tierces dans votre site web.

Vous pouvez éluder les paramètres `type="text/javascript"`, car tous les navigateurs modernes supposent que ces scripts contiennent du JavaScript.

Déboguer les erreurs en JavaScript

Lors de l'apprentissage de JavaScript, il est important de pouvoir suivre les erreurs de frappe ou de programmation. Au contraire de PHP, qui affiche des messages d'erreur dans le navigateur, JavaScript gère les messages d'erreur d'une façon qui varie en fonction du navigateur utilisé. Le tableau 13-1 détaille la manière d'accéder aux messages d'erreur de JavaScript dans cinq des navigateurs les plus courants.

Tableau 13-1. Accès aux messages d'erreur JavaScript dans différents navigateurs

Browser	Comment accéder aux messages d'erreur JavaScript
Apple Safari	Safari n'active pas de console d'erreur par défaut, mais vous pouvez l'activer à partir du bouton « Afficher un menu des réglages généraux de Safari » ou du menu Édition → Préférences → Avancées → « Afficher le menu Développement dans la barre de menu ». Vous pouvez également utiliser le module Firebug Lite JavaScript que nombre de personnes préfèrent.
Google Chrome	Cliquez sur l'icône de personnalisation et de contrôle de Google Chrome, puis sélectionnez Plus d'outils → Console JavaScript. Vous disposez aussi du raccourci clavier `Maj.+Ctrl+J` sur PC ou `Commande+Maj.+J` sur un Mac.
Microsoft Internet Explorer	Sélectionnez Outils → Options Internet → Avancé, puis sous la rubrique Navigation, ôtez la coche de la case Désactiver le débogage des scripts (Internet Explorer) et cochez la case Afficher une notification de chaque erreur de script.
Mozilla Firefox	Affichez la barre de menus, puis sélectionnez Outils → Développement Web → Console du navigateur, ou utilisez le raccourci clavier `Maj.+Ctrl+J` sur PC ou `Commande+Maj.+J` sur un Mac.
Opera	Cliquez sur Opera → Plus d'outils → Activer les outils de développement, puis cliquez sur Opera → Outils de développement → Inspecteur web. Cliquez sur l'onglet Console si nécessaire.

Aux utilisateurs d'OS X : bien que je vous aie montré comment créer une console d'erreur pour JavaScript dans Safari, vous pourriez préférer utiliser Google Chrome (pour OS X 10.5 ou plus, sur plateforme Intel).

Pour tester la console d'erreur, quelle qu'elle soit, que vous allez utiliser, créons un script avec une erreur mineure. L'exemple 13-3 est quasi identique à l'exemple 13-1, sauf qu'il manque un guillemet à la fin de la chaine `Bonjour tout le monde`, ce qui correspond à une erreur de syntaxe assez habituelle.

Exemple 13-3. Script JavaScript « Bonjour tout le monde » avec une erreur

```
<html>
  <head><title>Bonjour tout le monde</title></head>
  <body>
    <script type="text/javascript">
      document.write("Bonjour tout le monde)
    </script>
  </body>
</html>
```

Entrez cet exemple et enregistrez-le sous le nom de fichier *test.html*, puis appelez-le dans le navigateur. Il doit réussir à afficher le titre mais rien d'autre dans la fenêtre du navigateur. Ouvrez à présent la console d'erreur du navigateur et un message doit s'y afficher comme celui de l'exemple 13-4. Un lien situé à côté du message permet d'accéder à la source et, au clic sur le lien, la ligne fautive est surlignée (sans indiquer précisément l'endroit où se trouve l'erreur).

Exemple 13-4. Un message de la console d'erreur de Mozilla Firefox

```
SyntaxError: unterminated string literal
```

Dans Microsoft Internet Explorer, le message prend l'allure de l'exemple 13-5 et aucune flèche de lien n'apparait, mais vous obtenez la ligne et l'emplacement.

Exemple 13-5. Un message de la console d'erreur de Microsoft Internet Explorer

```
unterminated string constant
```

Google Chrome et Opera donnent le message de l'exemple 13-6. Chez eux aussi, vous obtenez le numéro de la ligne mais pas l'emplacement exact de l'erreur.

Exemple 13-6. Un message de la console d'erreur de Google Chrome et Opera

```
Uncaught SyntaxError: Unexpected token ILLEGAL
```

Enfin, Apple Safari fournit le message de l'exemple 13-7, avec à droite un lien vers la source qui indique le numéro de ligne de l'erreur. Cliquez sur le lien pour mettre la ligne fautive en évidence mais pas l'emplacement exact de l'erreur.

Exemple 13-7. Un message de la console d'erreur d'Apple Safari

```
SyntaxError: Unexpected EOF
```

Si vous trouvez ce soutien quelque peu décevant, le module d'extension Firebug pour Firefox (et actuellement aussi pour Chrome), disponible sur *http://getfirebug.com*, bénéficie d'une grande popularité parmi les développeurs JavaScript et vaut vraiment le détour.

Lorsque vous entrez les petits extraits de code suivants pour les tester, n'oubliez pas de les entourer des balises `<script>` et `</script>`.

Commentaires

Leur héritage partagé du langage de programmation C explique que PHP et JavaScript affichent de nombreuses similitudes, dont l'une se situe au niveau des commentaires. D'abord, il y a le commentaire sur une seule ligne, comme suit :

```
// Voici un commentaire
```

Ce style emprunte les deux barres obliques (//) pour informer JavaScript que tout ce qui les suit est à ignorer. Il y a aussi le bloc de commentaire sur plusieurs lignes :

```
/* Voici une section
   ou un bloc de commentaire
   de plusieurs lignes
   à ne pas interpréter */
```

Ici, le bloc de commentaire débute par la séquence /* et se termine par */. Retenez toutefois que vous ne pouvez pas imbriquer les blocs de commentaires sur plusieurs lignes, donc vérifiez que vous ne commentez pas de longues sections de code qui contiennent déjà des commentaires sur plusieurs lignes.

Points-virgules

Au contraire de PHP, JavaScript ne nécessite généralement pas de points-virgules si les instructions figurent seules par ligne. Par conséquent, ce qui suit est correct :

```
x += 10
```

En revanche, lorsque vous placez plus d'une instruction sur une même ligne, vous devez les séparer par des points-virgules, comme suit :

```
x += 10; y -= 5; z = 0
```

Normalement, vous pouvez omettre le point-virgule qui termine l'instruction finale.

La règle du point-virgule connait quelques exceptions. Si vous écrivez des minisignets (*bookmarklets*) JavaScript ou terminez une ligne avec une référence à une variable ou une fonction, alors que le premier caractère de la ligne qui suit est une parenthèse ouvrante ou un crochet gauche, vous devez alors ajouter un point-virgule à la fin de la première ligne, sinon le script affiche une erreur. Donc, dans le doute, mettez un point-virgule.

Variables

Aucun caractère particulier n'identifie spécialement une variable en JavaScript (comme le signe dollar en PHP). Les variables suivent les règles de nommage suivantes :

- Le nom d'une variable ne peut contenir que les lettres `a-z`, `A-Z`, `0-9`, le symbole `$` et le soulignement (`_`).
- Aucun autre caractère, espace ou de ponctuation n'est autorisé dans un nom de variable.
- Le premier caractère d'un nom de variable ne peut être qu'une lettre `a-z`, `A-Z`, le symbole `$` ou le soulignement (`_`), mais pas un chiffre.
- Les noms de variables sont sensibles à la casse, donc `Compteur`, `compteur` et `COMPTEUR` désignent des variables différentes.
- Aucune limite n'est définie au niveau de la longueur des noms de variables.

Oui, vous avez bien lu : le `$` figure bien dans cette liste. Il *est autorisé* en JavaScript et *peut* être le premier caractère d'un nom de variable ou de fonction. Mais bien que je vous déconseille de conserver les symboles `$`, sachez que si vous les conservez, vous pourrez ainsi porter plus rapidement de grands segments de code vers JavaScript.

Variables de chaines

En JavaScript, les variables de chaines de caractères sont entourées de guillemets verticaux ou d'apostrophes verticales :

```
salutation     = "Bonjour"
avertissement = 'Soyez prudent'
```

L'apostrophe est permise dans une chaine entourée de guillemets et le guillemet est permis dans une chaine entourée d'apostrophes. En revanche, vous devez placer une séquence d'échappement, c'est-à-dire une barre oblique inverse (`\`), devant un guillemet ou une apostrophe si la chaine est entourée de caractères du même type, comme suit :

```
salutation     = "\"Bonjour\" est une salutation"
avertissement = '\'Soyez prudent\' est un avertissement'
```

L'affectation d'une variable de chaine à une autre prend l'allure suivante :

```
nouvellechaine = anciennechaine
```

Vous pouvez aussi l'utiliser dans une fonction, comme suit :

```
statut = "Tous les systèmes sont opérationnels"
document.write(statut)
```


Variables numériques

Pour créer une variable numérique, affectez-lui une valeur de type nombre, comme suit :

```
compteur    = 42
temperature = 22.5
```

À l'instar des variables de chaines, les variables numériques peuvent intervenir dans des expressions, des fonctions et peuvent en recevoir les résultats.

Tableaux

Les tableaux en JavaScript se comportent de manière semblable à ceux de PHP, car ils peuvent contenir des chaines, des données numériques et d'autres tableaux. Pour attribuer des valeurs à un tableau, utilisez la syntaxe suivante (qui ici crée un tableau de chaines) :

```
jouets = ['raquette', 'balle', 'sifflet', 'puzzle', 'poupée']
```

Pour créer un tableau à plusieurs dimensions, imbriquez de plus petits tableaux dans un plus grand. Ainsi, pour constituer un tableau à deux dimensions contenant les couleurs d'une face d'un Cube (de) Rubik mélangé, où les couleurs rouge, vert, orange, jaune, bleu et blanc (W comme *white*) sont représentées par leur première lettre capitale, écrivez par exemple :

```
face =
[
  ['R', 'V', 'J'],
  ['W', 'R', 'O'],
  ['J', 'W', 'V']
]
```

Cet exemple est présenté sous une forme triviale, facile à comprendre et qui illustre bien comment les choses se passent, mais vous auriez pu l'écrire comme suit :

```
face = [['R', 'V', 'J'], ['W', 'R', 'O'], ['J', 'W', 'V']]
```

Ou même, comme ceci :

```
haut   = ['R', 'V', 'J'],
centre = ['W', 'R', 'O'],
bas    = ['J', 'W', 'V']

face = [haut, centre, bas]
```

Ensuite, pour accéder à l'élément situé à la deuxième ligne et à la troisième colonne de cette matrice, il suffit d'écrire (car les indices des éléments des tableaux commencent à 0) :

```
document.write(face[1][2])
```

Cette instruction affiche la lettre `O`, pour *orange*.

Les tableaux en JavaScript forment des structures extrêmement puissantes, qui méritent une étude spécifique. Le chapitre 15 les examine plus en profondeur.

Opérateurs

Tout comme ceux de PHP, les opérateurs en JavaScript portent sur les mathématiques, la conversion en chaines, ainsi que les opérations de comparaison et logiques (`and`, `or`, etc.). Les opérateurs mathématiques ressemblent à l'arithmétique classique ; ainsi, l'instruction suivante affiche `15` :

```
document.write(13 + 2)
```

Opérateurs arithmétiques

Les *opérateurs arithmétiques* permettent d'effectuer des opérations algébriques qui portent sur les quatre opérations principales (addition, soustraction, multiplication et division), plus le modulo (ou reste de la division entière), ainsi que l'incrémentation et la décrémentation d'une valeur (tableau 13-2).

Tableau 13-2. Opérateurs arithmétiques

Opérateur	Description	Exemple
+	Addition	j + 12
-	Soustraction	j - 22
*	Multiplication	j * 7
/	Division	j / 3.13
%	Modulo (reste de la division entière)	j % 6
++	Incrémentation	++j
--	Décrémentation	--j

Opérateurs d'affectation

Les *opérateurs d'affectation* servent à attribuer des valeurs à des variables. Ils vont du simple =, jusqu'aux `+=`, `-=` et ainsi de suite. L'opérateur `+=` ajoute la valeur située à droite de l'opérateur à celle de gauche, au lieu de totalement remplacer la valeur de gauche. Ainsi, si `compteur` débute avec la valeur `6`, l'instruction

```
compteur += 1;
```

met `compteur` à `7`, exactement comme l'instruction plus familière suivante :

```
compteur = compteur + 1;
```


Le tableau 13-3 énumère les opérateurs d'affectation disponibles.

Tableau 13-3. Opérateurs d'affectation

Opérateur	Exemple	Équivaut à
=	j = 99	j = 99
+=	j += 2	j = j + 2
+=	j += 'chaine'	j = j + 'chaine'
-=	j -= 12	j = j - 12
*=	j *= 2	j = j * 2
/=	j /= 6	j = j / 6
%=	j %= 7	j = j % 7

Opérateurs de comparaison

Les *opérateurs de comparaison* servent généralement dans une construction telle qu'une instruction `if` où il faut comparer deux éléments, par exemple, pour savoir si une variable que vous avez incrémentée a atteint une valeur donnée, ou si une autre variable est inférieure à une valeur définie, et ainsi de suite (tableau 13-4).

Tableau 13-4. Opérateurs de comparaison

Opérateur	Description	Exemple
==	est égal à	j == 42
!=	est différent de	j != 17
>	est plus grand que	j > 0
<	est plus petit que	j < 100
>=	est plus grand ou égal à	j >= 23
<=	est plus petit ou égal à	j <= 13
===	est égal à (et de même type que)	j === 56
!==	est différent de (ou de type différent de)	j !== '1'

Opérateurs logiques

Si comme PHP, JavaScript possède les *opérateurs logiques* `&&` et `||`, au contraire de PHP, il ne possède pas les opérateurs logiques `and` et `or`, ni l'opérateur `xor` (tableau 13-5).

Tableau 13-5. Opérateurs logiques

Opérateur	Description	Exemple
&&	Et	j == 1 && k == 2
||	Ou	j < 100 || j > 0
!	Non (ou pas)	! (j == k)

Incrémentation et décrémentation de variables

Les formes de pré et de post-incrémentation et décrémentation que vous avez vues en PHP sont également prises en charge en JavaScript :

```
++x
--y
x += 22
y -= 3
```

Concaténation de chaines

JavaScript gère la concaténation de chaines de manière légèrement différentes par rapport à PHP. Au lieu de l'opérateur . (point) de PHP, JavaScript utilise le signe + (plus), comme suit :

```
document.write("Vous avez " + messages + " messages.")
```

En supposant que la variable `messages` contienne la valeur 3, le résultat de cette instruction est le suivant :

```
Vous avez 3 messages.
```

De même que vous pouvez ajouter une valeur à une variable numérique avec l'opérateur +=, vous pouvez ajouter (*concaténer*) une chaine à la fin d'une autre :

```
acteur =  "Janes"
acteur += " Dean"
```

Séquences d'échappement

Les *séquences d'échappement*, que vous avez déjà utilisées pour insérer des guillemets et des apostrophes dans des chaines, permettent aussi d'insérer d'autres caractères spéciaux, tels que les tabulations, les nouvelles lignes et les retours à la ligne. Voici un exemple d'insertion de tabulations pour mettre un entête de tableau en forme ; il sert surtout ici à illustrer les séquences d'échappement, car les pages web disposent de meilleurs moyens pour ce faire :

```
entete = "Nom\tÂge\tProfession"
```

Le tableau 13-6 énumère les séquences d'échappement disponibles.

Tableau 13-6. Séquences d'échappement de JavaScript

Séquence	Signification
\b	Retour arrière
\f	Saut de page
\n	Nouvelle ligne
\r	Retour à la ligne
\t	Tabulation
\ '	Apostrophe verticale
\ "	Guillemet vertical
\\	Barre oblique inverse
XXX	Un nombre octal compris entre `000` et `377`, qui représente le caractère équivalent en `Latin-1` (comme `\251` pour le symbole ©)
\x*XX*	Un nombre hexadécimal compris entre `00` et `FF`, qui représente le caractère équivalent en `Latin-1` (comme `\xA9` pour le symbole ©)
\u*XXXX*	Un nombre hexadécimal compris entre `0000` et `FFFF`, qui représente le caractère équivalent en Unicode (comme `\u00A9` pour le symbole ©)

Typage des variables

Comme PHP, JavaScript est un langage à *typage faible*, souple. Le *type* d'une variable est défini seulement lorsqu'elle reçoit une valeur et peut évoluer si la variable apparait dans des contextes différents. Il n'est généralement pas nécessaire de se préoccuper du type, car JavaScript devine ce que vous voulez faire et le fait, tout simplement.

Examinez l'exemple 13-8, où :

1. La variable `n` reçoit la valeur de chaine `838102050`, la ligne suivante affiche sa valeur et l'opérateur `typeof` permet d'en déterminer le type.
2. La variable `n` reçoit ensuite la valeur renvoyée par la multiplication des nombres 12345 et 67890. La valeur est aussi de `838102050`, mais sous forme de nombre et non plus de chaine. Le type de la variable est inspecté et affiché.
3. Nous ajoutons un peu de texte au nombre contenu dans `n`, puis nous affichons le résultat.

Exemple 13-8. Réglage du type d'une variable par affectation

```
<script>
  n = '838102050'         // Définit 'n' comme une chaine
  document.write('n = ' + n + ' et est une ' + typeof n + '<br>')

  n = 12345 * 67890;      // Définit 'n' comme un nombre
  document.write('n = ' + n + ' et est un ' + typeof n + '<br>')
```

```
  n += ' plus du texte'  // Change le nombre 'n' en une chaine
  document.write('n = ' + n + ' et est une ' + typeof n + '<br>')
</script>
```

Les résultats de ce script prennent l'allure suivante (`string` et `number` sont issus de `typeof`) :

```
n = 838102050 et est une string
n = 838102050 et est un number
n = 838102050 plus du texte et est une string
```

S'il y a le moindre doute à propos du type d'une variable ou si vous devez garantir qu'une variable soit d'un type déterminé, vous pouvez lui imposer d'adopter ce type à l'aide d'instructions telles que les suivantes qui convertissent respectivement une chaine en nombre et inversement :

```
n = "123"
n *= 1     // Convertit 'n' en un nombre

n = 123
n += ""    // Convertit 'n' en une chaine
```

Bien entendu, vous disposez toujours de l'opérateur `typeof` qui permet de déterminer le type de la variable.

Fonctions

Comme en PHP, les fonctions en JavaScript permettent d'isoler des portions de code qui exécutent des tâches spécifiques, puis de les appeler selon les nécessités. Pour créer une fonction, déclarez-la comme dans l'exemple 13-9.

Exemple 13-9. Déclaration de fonction simple

```
<script>
  function produit(a, b)
  {
    return a*b
  }
</script>
```

Cette fonction demande deux paramètres et les multiplie pour en renvoyer le produit.

Variables globales

Les *variables globales* sont celles définies en dehors de toute fonction (ou au sein de fonctions mais définies sans le mot clé `var`). Deux manières existent pour les définir :


```
    a = 123              // Portée globale
  var b = 456            // Portée globale
  if (a == 123) var c = 789 // Portée globale
```

Que vous mentionniez ou non le mot clé `var`, tant que la définition d'une variable ait lieu en dehors d'une fonction, sa portée est globale. Cela signifie que la variable est accessible de partout dans un script.

Variables locales

Les paramètres transmis à une fonction ont d'emblée une portée locale, c'est-à-dire qu'il n'est possible d'y faire référence qu'au sein de cette fonction. Subsiste toutefois une exception au niveau des tableaux. Les tableaux sont passés par référence, donc si vous modifiez le moindre élément d'un tableau en paramètre d'une fonction, alors l'élément correspondant du tableau original est également modifié.

Pour définir une variable locale dont la portée se limite à la fonction courante qui ne soit pas passée en paramètre de celle-ci, utilisez le mot clé `var` pour la définir. L'exemple 13-10 illustre une fonction qui crée une variable de portée globale et deux autres de portée locale.

Exemple 13-10. Création de variables de portées globale et locale dans une fonction

```
<script>
  function test()
  {
      a = 123              // Portée globale
    var b = 456            // Portée locale
    if (a == 123) var c = 789 // Portée locale
  }
</script>
```

Pour tester le fonctionnement de la portée en PHP, nous utilisions la fonction `isset` mais aucun équivalent n'existe en JavaScript, donc l'exemple 13-11 exploite l'opérateur `typeof` qui renvoie la chaine `undefined` quand une variable n'est pas définie.

Exemple 13-11. Vérification de la portée des variables de la fonction test

```
<script>
  test()

  if (typeof a != 'undefined') document.write('a = "' + a + '"<br />')
  if (typeof b != 'undefined') document.write('b = "' + b + '"<br />')
  if (typeof c != 'undefined') document.write('c = "' + c + '"<br />')

  function test()
  {
     a     = 123
    var b = 456
```

```
    if (a == 123) var c = 789
  }
</script>
```

Le résultat affiché par ce script n'offre que la ligne suivante :

```
a = "123"
```

Ceci indique que seule la variable `a` possède une portée globale, comme nous pouvions nous y attendre, puisque les variables `b` et `c` ont été préfacées du mot clé `var`, qui les rend locales.

Si votre navigateur émet un avertissement indiquant que `b` n'est pas défini, cet avertissement est correct et vous pouvez l'ignorer.

Le modèle DOM : Document object model

Les concepteurs de JavaScript ont été particulièrement bien avisés. Au lieu de créer encore un autre langage de script (qui constituait déjà en son temps une amélioration plutôt bienvenue), ils ont pressenti l'importance de l'édifier autour du modèle objet de document *DOM* (*Document object model*). Celui-ci décompose les portions d'un document HTML en objets discrets, qui possèdent chacun leurs propres *propriétés et méthodes*, dont le contrôle devient ainsi à la portée de JavaScript.

JavaScript distingue les objets, propriétés et méthodes à l'aide d'un point (ce qui explique que le symbole `+` sert d'opérateur de concaténation au lieu du point qu'utilise PHP). Pour comprendre comment cela fonctionne, prenons l'exemple d'une carte de visite, que nous considérons comme un objet nommé `carte`. Cet objet possède des propriétés, telles que le nom, l'adresse, le numéro de téléphone et ainsi de suite. Selon la syntaxe de JavaScript, pour accéder à ces propriétés, nous écrivons :

```
carte.nom
carte.telephone
carte.adresse
```

Les méthodes de l'objet sont les fonctions qui permettent de lire, de modifier et, d'une manière générale, d'agir sur les propriétés. Ainsi, pour appeler une méthode qui affiche les propriétés de l'objet `carte`, vous utiliseriez la syntaxe suivante :

```
carte.affiche()
```

Retournez brièvement aux premiers exemples de ce chapitre et examinez l'utilisation de l'instruction `document.write`. Maintenant que vous savez que JavaScript se base sur des objets, vous comprenez que `write` est en fait une méthode de l'objet `document`.

Au sein de JavaScript existe une hiérarchie d'objets parents et enfants, que l'on appelle le *Document Object Model* (figure 13-3).

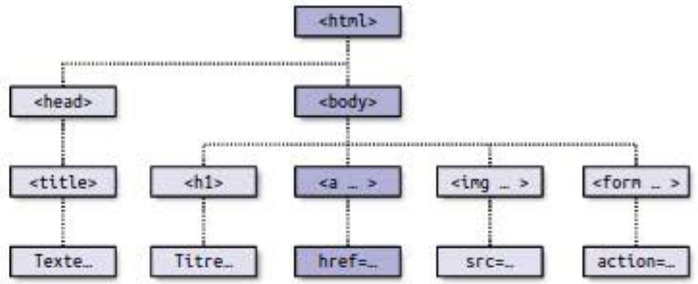


Figure 13-3. Exemple de hiérarchie objet DOM

Cette figure reprend les balises HTML que vous avez déjà rencontrées pour illustrer la relation parent-enfant entre les différents objets d'un document. Par exemple, une URL dans un lien (*link*) fait partie du corps (*body*) d'un document HTML. Pour y accéder en JavaScript, vous pourriez écrire :

```
url = document.links.monlien.href
```

Remarquez que cette expression suit la colonne centrale vers le bas. La première partie, `document`, fait référence aux balises `<html>` et `<body>`; `links.monlien` à la balise `<a>` et `href` à l'attribut `href` de celle-ci.

Passons à un peu de HTML et à un script pour lire les propriétés d'un lien. Entrez l'exemple 13-12 et enregistrez-le sous le nom de fichier *linktest.html*, puis appelez celui-ci dans votre navigateur.

Si vous utilisez Microsoft Internet Explorer comme navigateur de développement principal, lisez seulement cette section, puis lisez celle intitulée « Ce n'est pas aussi simple que ça ! » à la page 326 et, enfin, revenez à cette section pour essayer l'exemple avec la modification du `getElementById` expliquée là. Sans celle-ci, l'exemple suivant ne fonctionne pas.

Exemple 13-12. Lecture de l'URL d'un lien en JavaScript

```
<html>
  <head>
    <title>Test de lien</title>
  </head>
  <body>
    <a id="monlien" href="http://monsite.com">Cliquez ici</a><br />
    <script>
      url = document.links.monlien.href
      document.write('L\'URL est : ' + url)
    </script>
  </body>
</html>
```

Remarquez la forme réduite des balises <script>, où j'ai éludé le paramètre type="text/JavaScript" pour simplifier la frappe au clavier. Si vous le voulez, juste pour le test de cet exemple (et des suivants), vous pouvez éluder tout, en dehors des balises <script> et </script>. Le résultat de cet exemple affiche ce qui suit :

```
Cliquez ici
L'URL est : http://monsite.com
```

La deuxième ligne de la sortie provient de la méthode document.write. Notez au passage que le code suit l'arborescence du document, de haut en bas, depuis document vers links, puis monlien (l'id donné au lien), puis href (la valeur de l'URL de destination).

Une autre forme abrégée fonctionne aussi, qui part de la valeur de l'attribut id du lien : monlien.href. Donc, vous pouvez remplacer ceci :

```
url = document.links.monlien.href
```

par cela :

```
url = monlien.href
```

Ce n'est pas aussi simple que ça !

Que vous essayiez l'exemple 13-12 dans Safari, Firefox, Opera ou Chrome, tout devrait bien se passer en principe. Mais avec Internet Explorer, rien ne va plus, parce que l'implantation de JavaScript de Microsoft, appelée *JScript*, présente de nombreuses différences subtiles par rapport aux standards établis. Bienvenue dans le monde du développement web avancé!

Alors, que faire dans ce cas-ci ? Eh bien, au lieu d'utiliser l'objet enfant links de l'objet parent document, sur lequel Internet Explorer trébuche, vous devez le remplacer par une méthode qui permet de retrouver un élément par son id. En conséquence, la ligne suivante,

```
url = document.links.monlien.href
```

doit être remplacée par celle-ci :

```
url = document.getElementById('monlien').href
```

À ce stade, ce script fonctionne dans tous les principaux navigateurs. En corolaire, lorsque vous n'êtes pas obligé de rechercher l'élément par son id, la forme abrégée qui suit fonctionne encore dans Internet Explorer, ainsi que dans les autres navigateurs :

```
url = monlien.href
```


Autre utilisation du symbole $

Comme indiqué précédemment, JavaScript autorise l'utilisation de $ dans les noms de variables et de fonctions. De ce fait, vous rencontrerez parfois du code avec des formulations étranges, telles que la suivante :

```
url = $('monlien').href
```

Certains programmeurs particulièrement entreprenants ont décidé que la fonction getElementById était tellement incontournable en JavaScript qu'ils ont écrit une fonction pour la remplacer et l'ont appelée $, comme dans jQuery (notez que jQuery utilise le $ pour bien plus que cela, comme indiqué au chapitre 21). L'exemple 13-13 illustre ce remplacement.

Exemple 13-13. Fonction de remplacement de la méthode getElementById

```
<script>
  function $(id)
  {
    return document.getElementById(id)
  }
</script>
```

Ainsi, si vous insérez la définition de la fonction $ dans votre code, la syntaxe suivante :

```
$('monlien').href
```

remplace automatiquement un code tel que le suivant :

```
document.getElementById('monlien').href
```

Utiliser le DOM

L'élément links est en pratique un tableau contenant des URL (une collection). En conséquence, la référence à l'URL monlien de l'exemple 13-12 est possible en toute sécurité dans tous les navigateurs par l'entremise de l'expression suivante (parce que c'est le premier et seul lien du corps de cette page) :

```
url = document.links[0].href
```

Pour connaitre le nombre de liens disponible dans la totalité d'un document, examinez la propriété length (longueur) de l'objet links, comme suit :

```
nbreliens = document.links.length
```

Puis, pour extraire et afficher tous les liens présents dans un document, écrivez ceci :

```
for (j=0 ; j < document.links.length ; ++j)
  document.write(document.links[j].href + '<br>')
```

Par convention et simple intuition, length est une propriété de tout tableau, de même qu'une propriété ou une méthode de bien d'autres objets. Ainsi, pour connaitre le nombre d'éléments présents dans l'historique de navigation de votre navigateur web, vous pouvez écrire quelque chose du genre :

```
document.write(history.length)
```

Cela dit, comme ce n'est jamais bon de voir un site web fouiller dans l'historique de votre navigateur, l'objet `history` ne mémorise que le nombre de sites dans le tableau : vous ne pouvez ni lire ni écrire dans ces valeurs. En revanche, vous pouvez remplacer la page courante par une de celles de l'historique, à condition de connaitre son emplacement dans l'historique. Ceci peut s'avérer très utile lorsque vous savez que certaines pages de l'historique proviennent de votre site, ou si vous souhaitez simplement renvoyer le navigateur parmi une ou plusieurs pages en arrière. La méthode `go` de l'objet `history` est conçue pour cela. Ainsi, pour renvoyer le navigateur trois pages en arrière, envoyez la commande suivante :

```
history.go(-3)
```

Les deux méthodes suivantes permettent respectivement d'avancer ou de reculer d'une page à la fois :

```
history.back()
history.forward()
```

De la même manière, vous pouvez remplacer l'URL chargée actuellement par une autre de votre choix, avec :

```
document.location.href = 'https://www.google.fr'
```

Vous imaginez bien que le DOM offre bien plus que simplement lire et modifier des liens. À mesure que vous avancerez dans les chapitres suivants sur JavaScript, vous vous familiariserez avec le DOM et les manières d'y accéder.

À propos de document.write

Lorsqu'il s'agit d'apprendre la programmation, il s'avère utile de disposer d'un moyen pour afficher facilement et rapidement le résultat d'une expression. En PHP par exemple, les instructions `echo` et `print` suffisent pour envoyer du texte au navigateur, donc c'est simple. En JavaScript, en revanche, il reste les alternatives suivantes.

Utiliser console.log

La fonction `console.log` permet d'envoyer le résultat de n'importe quelle valeur ou expression qui lui est envoyée dans la console du navigateur courant. Il s'agit alors de mettre en place un mode spécial avec un volet distinct dans lequel les erreurs et autres messages d'avertissement s'affichent. Si cela comble les programmeurs expérimentés, ce n'est pas l'idéal pour les débutants, parce que l'ouverture de cette console diffère selon les navigateurs et les affichages ne sont pas à proximité du contenu web dans le navigateur.

Utiliser alert

La fonction `alert` permet d'afficher les valeurs et expressions qui lui sont données dans une fenêtre contextuelle (*pop-up window*), qui exige de cliquer sur un bouton pour la clôturer. Très rapidement, ce genre d'interférence irrite et présente l'inconvénient qu'une telle boite de dialogue n'affiche que le dernier message, tandis que les précédents sont perdus si vous ne les avez pas notés à la main.

Écrire dans les éléments

Il est possible d'écrire directement dans le texte d'un élément HTML, ce qui constitue une solution élégante, en particulier pour les sites web de production. L'ennui, c'est que dans le contexte de ce livre, il faudrait créer au préalable chacun de ces éléments et rédiger quelques lignes de code JavaScript pour y accéder. Donc, pour expliquer des exemples de base, le code deviendrait rapidement lourd et déroutant.

Utiliser document.write

Donc, reste `document.write`. Cette fonction écrit une valeur ou le résultat d'une expression à l'emplacement courant dans le navigateur et fournit une technique simple et efficace pour afficher des résultats, parce qu'elle conserve aux exemples leur simplicité et leur concision, en plaçant ses sorties dans le navigateur à l'endroit le plus proche du contenu web et du code exposés.

Vous avez peut-être déjà entendu dire que cette fonction est considérée comme non sécuritaire par certains développeurs parce qu'elle écrase le document courant quand vous l'appelez après le chargement complet d'une page. Si c'est fondamentalement exact, le problème n'affecte aucun des exemples de ce livre, car ils empruntent `document.write` de la manière dont la fonction a été conçue à l'origine, en tant qu'élément participatif de la construction d'une page, donc par un appel *avant* que la page soit complètement chargée et affichée.

Soyons francs : si j'utilise `document.write` de cette manière pour des exemples simples, je ne l'utilise en principe jamais dans du code de production (sauf circonstances exceptionnelles, où elle est indispensable). Je préfère écrire dans un élément préparé spécialement à cet effet, comme dans les exemples beaucoup plus complexes du chapitre 17 (qui accèdent à la propriété `innerHTML` d'éléments pour les affichages des programmes).

Par conséquent, lorsque vous voyez un appel à `document.write` dans ce livre, considérez qu'il ne s'agit que d'un moyen de simplifier les exemples et d'obtenir rapidement des résultats de tests. N'utilisez cette fonction que de cette manière et dans ce contexte.

Forts de ces explications et mises en garde, nous sommes prêts à poursuivre notre exploration de JavaScript au chapitre suivant, qui examine le contrôle de flux d'exécution des scripts et la rédaction d'expressions.

Questions

1. Quelles sont les balises utilisées pour entourer du code JavaScript ?
2. Par défaut, dans quelle partie d'un document le code JavaScript envoie-t-il ses sorties ?
3. Comment fait-on pour inclure du code JavaScript d'une autre source dans des documents ?
4. Quelle est la fonction JavaScript équivalente à `echo` ou `print` en PHP ?
5. Comment crée-t-on des commentaires en JavaScript ?

6. Quel est l'opérateur de concaténation de chaine en JavaScript ?
7. Quel mot clé utilise-t-on dans une fonction JavaScript pour définir une variable à portée locale ?
8. Donnez deux méthodes valables sur plusieurs navigateurs pour afficher l'URL affectée au lien d'identifiant `celien`.
9. Quelles sont les deux commandes JavaScript qui permettent d'indiquer au navigateur de charger la page précédente de l'historique de navigation ?
10. Quelle commande JavaScript utiliseriez-vous pour remplacer le document courant d'un navigateur par la page principale de *goulet.ca* ?

Retrouvez les réponses du chapitre 13 dans l'annexe A.

CHAPITRE 14

Expressions et contrôle du flux d'exécution en JavaScript

Le chapitre précédent vous a présenté les bases de JavaScript et du modèle DOM. Ce chapitre aborde la construction d'expressions complexes en JavaScript et le contrôle du flux d'exécution des scripts à l'aide d'instructions conditionnelles.

Expressions

En JavaScript, les expressions ressemblent fortement à celles en PHP. Le chapitre 4 expliquait qu'une expression est une combinaison de valeurs, de variables, d'opérateurs et de fonctions qui renvoie une valeur; ce résultat peut être un nombre, une chaine ou une valeur booléenne (qui s'évalue à `true` ou `false`).

L'exemple 14-1 illustre quelques expressions simples. Chaque ligne affiche une lettre comprise entre `a` et `d`, suivie de deux points et du résultat d'une expression. La balise `<br>` permet de renvoyer la suite à la ligne suivante et d'étaler les sorties sur quatre lignes (rappelez-vous que HTML5 accepte indifféremment `<br>` et `<br />`, donc choisissez la formulation la plus courte).

Exemple 14-1. Quatre expressions booléennes simples

```
<script>
  document.write("a : " + (42 > 3) + "<br>")
  document.write("b : " + (91 < 4) + "<br>")
  document.write("c : " + (8 == 2) + "<br>")
  document.write("d : " + (4 < 17) + "<br>")
</script>
```

Les résultats de ce script sont les suivants :

```
a : true
b : false
c : false
d : true
```

Les deux expressions `a` et `d` s'évaluent à `true`, tandis que les expressions `b` et `c` s'évaluent à `false`. Remarquez que PHP renverrait le nombre `1` à la place de `true` et rien du tout à la place de `false`. Et remarquez aussi que ce sont les chaines `true` et `false` qui s'affichent ici.

En JavaScript, lorsque vous vérifiez si une valeur est vraie ou fausse, toutes les valeurs s'évaluent à `true`, sauf les suivantes, qui s'évaluent à `false` : la chaine `false` elle-même, `0`, `-0`, la chaine vide, `null`, `undefined` et `NaN` (*not a number*, un concept du génie logiciel qui correspond à une opération illégale sur des nombres à virgule flottante, comme la division par zéro).

Remarquez également qu'en JavaScript, `true` et `false` s'écrivent obligatoirement en bas de casse (lettres minuscules), au contraire de ce qui se passe en PHP. Ainsi, seule la première des deux instructions suivantes affichera son résultat, le mot `true` en bas de casse, parce que la seconde provoque l'affichage du message d'erreur `'TRUE' is not defined` :

```
if (1 == true) document.write('true') // Vrai
if (1 == TRUE) document.write('TRUE') // Provoque une erreur
```

Si vous souhaitez tester ces bribes de code, n'oubliez pas de les entourer des balises `<script>` et `</script>` dans un document HTML.

Valeurs littérales et variables

La forme la plus simple d'expression se situe dans le *littéral*, ou *valeur littérale*, qui désigne quelque chose qui s'évalue à lui-même, comme le nombre `22` ou la chaine `'Appuyez sur Entrée'`. Une expression peut aussi être une variable, dont l'évaluation donne la valeur qui lui a été affectée. Ce sont là deux types d'expressions parce qu'elles renvoient une valeur.

L'exemple 14-2 illustre différentes valeurs littérales et deux variables, qui renvoient toutes des valeurs, même si elles sont de types différents.

Exemple 14-2. Cinq types de littéraux

```
<script>
  monnom = "Pierre"
  monage = 24
  document.write("a : " + 42     + "<br>") // Littéral numérique
  document.write("b : " + "Allo" + "<br>") // Littéral chaine
  document.write("c : " + true   + "<br>") // Littéral constant
  document.write("d : " + monnom + "<br>") // Variable chaine
  document.write("e : " + monage + "<br>") // Variable numérique
</script>
```


Et, vous vous en doutez, vous voyez une valeur renvoyée par toutes ces instructions, comme suit :

```
a : 42
b : Allo
c : true
d : Pierre
e : 24
```

Les opérateurs permettent de créer des expressions plus complexes, qui s'évaluent à des résultats plus utiles. Lorsque vous combinez l'affectation ou des constructions de contrôle de flux de programme avec des expressions, vous obtenez une *instruction*.

L'exemple 14-3 en montre une d'affectation et une de contrôle de flux. La première affecte le résultat de l'expression `366 - numero_du_jour` à la variable `jours_avant_nouvel_an` et la seconde sort un message convivial seulement si l'expression `jours_avant_nouvel_an < 30` s'évalue à `true`.

Exemple 14-3. Deux instructions simples en JavaScript

```
<script>
  jours_avant_nouvel_an = 366 - numero_du_jour;
  if (jours_avant_nouvel_an < 30) document.write("Le Nouvel An est pour bientôt")
</script>
```

Opérateurs

JavaScript offre de nombreux opérateurs puissants, qui vont des opérateurs arithmétiques, de chaines et logiques, jusqu'aux opérateurs d'affectation et de comparaison, entre autres (tableau 14-1).

Tableau 14-1. Types d'opérateurs de JavaScript

Opérateur	Description	Exemple
Arithmétique	Mathématiques de base	`a + b`
Tableau	Manipulation de tableaux	`a + b`
Affectation	Affectation de valeurs	`a = b + 23`
Niveau des bits	Manipuler les bits dans des octets	`12 ^ 9`
Comparaison	Comparaison de deux valeurs	`a < b`
Incrémentation/décrémentation	Ajouter ou soustraire un	`a++`
Logique	Booléen	`a && b`
Chaine	Concaténation	`a + 'chaine'`

Chaque opérateur prend un nombre d'opérandes différent :

- Les opérateurs *unaires*, comme l'incrémentation (`a++`) ou la négation (`-a`), ne prennent qu'un seul opérande.

- Les opérateurs *binaires*, qui représentent l'essentiel des opérateurs de JavaScript, dont l'addition, la soustraction, la multiplication et la division, attendent deux opérandes.
- Un seul opérateur *ternaire*, qui adopte la forme `test ? si_vrai : si_faux`. Il s'agit d'une instruction `if` écrite en une seule ligne, qui effectue un choix entre les portions `si_vrai` et `si_faux`, en fonction de l'état de l'évaluation du troisième, `test`.

Préséance des opérateurs

Comme PHP, JavaScript impose une préséance aux opérateurs, où certaines opérations d'une expression sont considérées comme plus importantes que d'autres et sont donc évaluées en premier. Le tableau 14-2 énumère les opérateurs de JavaScript et leur préséance.

Tableau 14-2. Préséance (décroissante) des types d'opérateurs de JavaScript

Opérateurs	Type
`() [] .`	Parenthèses, appel et membre
`++ --`	Incrémentation/décrémentation
`+ - ~ !`	Unaire, niveau des bits et logique
`* / %`	Arithmétique
`+ -`	Arithmétique et chaine
`<< >> >>>`	Niveau des bits
`< > <= >=`	Comparaison
`== != === !==`	Comparaison
`& ^ |`	Niveau des bits
`&&`	Logique
`||`	Logique
`? :`	Ternaire
`= += -= *= /= %=`	Affectation
`<<= >>= >>>= &= ^= |=`	Affectation
`,`	Séparateur

Associativité

La plupart des opérateurs JavaScript sont évalués de la gauche vers la droite dans une équation, mais certains opérateurs nécessitent un traitement de droite à gauche. Le sens du traitement s'appelle l'*associativité* de l'opérateur.

Cette associativité devient particulièrement importante dans les cas où vous n'imposez pas la préséance. Ainsi, examinez les opérateurs d'affectation dans l'instruction suivante, qui initialise trois variables à la valeur 0 :

```
niveau = score = duree = 0
```


Cette affectation multiple n'est possible que parce que la partie la plus à droite est évaluée en premier lieu, puis le traitement évolue progressivement vers la gauche. Le tableau 14-3 énumère les opérateurs et leur associativité.

Tableau 14-3. Associativité des opérateurs

Opérateur	Description	Associativité
`++ --`	Incrémentation, décrémentation	Aucune
`new`	Création d'un nouvel objet	Droite
`+ - ~ !`	Unaire et au niveau des bits	Droite
`?:`	Opérateur ternaire	Droite
`= *= /= %= += -=`	Affectation	Droite
`<<= >>= >>>= &= ^= |=`	Affectation	Droite
`,`	Séparateur (virgule)	Gauche
`+ - * / %`	Arithmétique	Gauche
`<< >> >>>`	Opérations au niveau des bits	Gauche
`< <= > >= == != === !==`	Relationnels	Gauche

Opérateurs relationnels

Les *opérateurs relationnels* testent deux opérandes et renvoient un résultat booléen soit `true` (vrai), soit `false` (faux). Trois types d'opérateurs relationnels existent : d'égalité, de comparaison et logiques.

Opérateur d'égalité

L'opérateur d'égalité est `==`, qu'il ne faut surtout pas confondre avec `=`, le symbole d'affectation. Dans l'exemple 14-4, la première instruction affecte une valeur, tandis que la seconde teste son égalité avec une autre valeur. Tel qu'il est écrit, le script n'affiche rien, puisque la variable `mois` reçoit la valeur `juillet` et que la vérification si le mois est égal à `octobre` échoue.

Exemple 14-4. Affectation d'une valeur et test d'égalité

```
<script>
  mois = "juillet"
  if (mois == "octobre") document.write("C'est l'automne")
</script>
```

Si les types des deux opérandes d'une expression d'égalité diffèrent, JavaScript les convertit au type qui lui semble de plus approprié. Ainsi, toute chaine qui ne contient que des chiffres est convertie automatiquement en nombre, chaque fois qu'elle est comparée à un nombre. Dans l'exemple 14-5, `a` et `b` ont deux valeurs différentes (l'une avec un nombre et l'autre avec une chaine), donc nous pourrions nous attendre à ce qu'aucune des deux instructions `if` n'affiche de résultat.

Exemple 14-5. Les opérateurs d'égalité et d'identité

```
<script>
  a = 3.1415927
  b = "3.1415927"
  if (a == b)  document.write("1")
  if (a === b) document.write("2")
</script>
```

Pourtant, lorsque vous exécutez cet exemple, vous constatez qu'il affiche `1`, ce qui signifie que l'expression de la première instruction `if` s'est évaluée à `true`. En effet, la valeur de chaine de `b` a été temporairement convertie en un nombre et donc les deux opérandes de l'égalité avaient la valeur numérique `3.1415927`.

À titre de comparaison, la seconde instruction `if` emploie l'opérateur d'identité, constitué de trois signes = successifs, qui empêche JavaScript de convertir automatiquement les types. Cela signifie que, comme `a` et `b` ne sont pas de mêmes types, ils sont différents et rien ne s'affiche.

Lorsque vous doutez de la manière dont JavaScript convertira les types d'opérandes, utilisez l'opérateur d'identité pour désactiver ce comportement.

Opérateurs de comparaison

À l'aide des opérateurs de comparaison, vous pouvez vous livrer à d'autres tests que simplement l'égalité ou l'inégalité. JavaScript fournit aussi les opérateurs `>` (plus grand que), `<` (plus petit que), `>=` (plus grand ou égal à) et `<=` (plus petit ou égal à), qui autorisent des tests plus variés. L'exemple 14-6 les met en œuvre.

Exemple 14-6. Les quatre opérateurs de comparaison

```
<script>
  a = 7; b = 11
  if (a > b)  document.write("a est plus grand que b<br>")
  if (a < b)  document.write("a est plus petit que b<br>")
  if (a >= b) document.write("a est plus grand ou égal à b<br>")
  if (a <= b) document.write("a est plus petit ou égal à b<br>")
</script>
```

Dans cet exemple, où `a` vaut 7 et `b` vaut 11, les résultats suivants sont générés (parce que 7 est plus petit que 11 et qu'il est aussi plus petit ou égal à 11) :

```
a est plus petit que b
a est plus petit ou égal à b
```

Opérateurs logiques

Les opérateurs logiques produisent des résultats soit vrais, soit faux, et sont de ce fait connus aussi sous le nom d'opérateurs booléens. JavaScript en comporte trois, énumérés au tableau 14-4.

Tableau 14-4. Les opérateurs logiques de JavaScript

Opérateur logique	Description
&&	*(Et)* `true` si les deux opérandes sont à `true`
||	*(Ou)* `true` si au moins un des deux opérandes vaut `true`
!	*(non)* `true` si l'opérande vaut `false`, `false` si l'opérande vaut `true`

L'exemple 14-7 illustre leur utilisation et affiche `0`, `1` et `true`.

Exemple 14-7. Les opérateurs logiques en action

```
<script>
  a = 1; b = 0
  document.write((a && b) + "<br>")
  document.write((a || b) + "<br>")
  document.write((  !b  ) + "<br>")
</script>
```

L'opérateur `&&` (et) exige que ses deux opérandes valent `true` pour renvoyer `true`. L'opérateur `||` (ou) renvoie `true` si au moins un de ses opérandes vaut `true`, tandis que l'opérateur `!` effectue un `NON` sur la valeur de `b` et retourne `true` lorsque `b` vaut `0`.

L'opérateur `||` peut provoquer des problèmes inattendus, parce que le deuxième opérande n'est pas évalué si le premier est évalué à `true`. Ainsi, dans l'exemple 14-8, la fonction `getnext` n'est jamais appelée si la valeur de `termine` vaut `1`.

Exemple 14-8. Utilisation de l'opérateur || dans une instruction

```
<script>
  if (termine == 1 || getnext() == 1) fini = 1
</script>
```

Si vous devez absolument appeler `getnext` dans l'instruction `if`, quelle que soit la valeur de `termine`, il vaut mieux récrire ce code comme dans l'exemple 14-9.

Exemple 14-9. Récriture de l'instruction if...OR pour garantir l'appel de getnext

```
<script>
  gn = getnext()
  if (termine == 1 OR gn == 1) fini = 1;
</script>
```

Dans cette version-ci, le code de la fonction `getnext` est exécuté avant le `if` et renvoie une valeur mémorisée dans `gn`, utilisée ensuite dans l'instruction `if`.

Le tableau 14-5 illustre toutes les variations des résultats des opérateurs logiques. Notez que `!true` donne `false` et que `!false` donne `true`.

Tableau 14-5. Toutes les expressions logiques possibles en JavaScript

Entrées		Opérateur et résultats	
A	**b**	**&&**	**||**
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

Instruction with

L'instruction `with` (avec) n'a jamais été vue dans les chapitres précédents sur PHP parce qu'elle est exclusive à JavaScript. Avec elle, vous pouvez simplifier certains types d'instructions JavaScript par une réduction de nombreuses références à un objet, à une seule référence à cet objet. Les références aux propriétés et méthodes au sein du bloc `with` se rapportent à cet objet.

Ainsi, dans le code de l'exemple 14-10, la fonction `document.write` ne fait aucune référence explicite au nom de la variable `chaine`.

Exemple 14-10. Utilisation de l'instruction with

```
<script>
  chaine = "Voix ambigüe d'un coeur qui au zéphyr préfère les jattes de kiwis"

  with (chaine)
  {
    document.write("La chaine a " + length + " caractères<br>")
    document.write("En capitales : " + toUpperCase())
  }
</script>
```

Même si le nom de `chaine` n'est jamais référencé directement dans `document.write`, ce code parvient néanmoins à afficher ce qui suit :

```
La chaine a 65 caractères
En capitales : VOIX AMBIGÜE D'UN COEUR QUI AU ZÉPHYR PRÉFÈRE LES JATTES DE KIWIS
```

Voici comment cela fonctionne : l'interpréteur JavaScript reconnaît que la propriété `length` et la méthode `toUpperCase()` s'appliquent à un objet. Or, comme elles figurent seules, sans cet objet, l'interpréteur suppose qu'elles s'appliquent à l'objet `chaine` que vous avez indiqué dans l'instruction `with`.

Utiliser onerror

D'autres constructions existent en JavaScript qui n'existent pas en PHP. À l'aide de l'évènement `onerror` ou d'une combinaison des mots clés `try` et `catch`, vous pouvez capturer des erreurs en JavaScript et les traiter vous-même.

Les évènements sont des actions détectées par JavaScript. Chaque élément sur une page web possède certains évènements qui peuvent déclencher des fonctions JavaScript. Par exemple, vous pouvez définir l'évènement `onclick` d'un élément de type bouton pour appeler une fonction et l'exécuter chaque fois que l'utilisateur clique sur ce bouton.

L'exemple 14-11 illustre l'utilisation de l'évènement `onerror`.

Exemple 14-11. Un script d'utilisation de l'évènement onerror

```
<script>
  onerror = errorHandler
  document.writ("Bienvenue sur ce site Web") // Erreur intentionnelle

  function errorHandler(message, url, ligne)
  {
    out  = "Désolé, une erreur s'est produite.\n\n";
    out += "Erreur : " + message + "\n";
    out += "URL : "    + url     + "\n";
    out += "Ligne : "  + ligne   + "\n\n";
    out += "Cliquez sur OK pour continuer.\n\n";
    alert(out);
    return true;
  }
</script>
```

La première ligne de ce script indique à l'évènement d'erreur d'utiliser la nouvelle fonction `errorHandler` à partir de là. Cette fonction demande trois paramètres : un `message`, une `url` et un numéro de `ligne`. Il est donc facile d'afficher ces trois informations dans une fenêtre incrustée d'alerte.

Ensuite, pour tester cette fonction, nous laissons délibérément une erreur de syntaxe dans le code, avec un appel de `document.writ`, où le `e` final manque. La figure 14-1 illustre les résultats de l'exécution de ce script dans un navigateur. L'utilisation de `onerror` de cette manière peut s'avérer également très utile en phase de débogage.

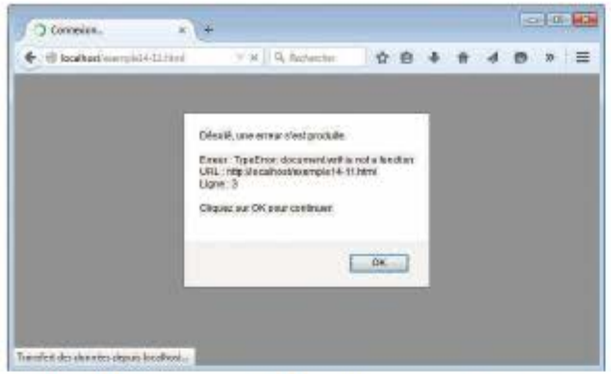


Figure 14-1. Utilisation de l'évènement onerror avec une alerte incrustée

Utiliser try...catch

Les mots clés `try` (essayer) et `catch` (attraper) représentent une technique plus formelle et souple que celle de `onerror` vue dans la section précédente. Ces mots clés permettent de capturer des erreurs dans une section de code bien déterminée, au lieu de parmi tous les scripts d'un document. Cependant, ils ne capturent pas les erreurs de syntaxe, pour lesquelles vous devez encore utiliser `onerror`.

La construction `try...catch` est prise en charge par tous les navigateurs principaux et s'avère pratique lorsqu'il s'agit de capturer une certaine condition dont vous êtes conscient de la possibilité qu'elle se produise dans une portion délimitée de votre programme.

Par exemple, au chapitre 17, nous explorons des techniques Ajax qui font appel à l'objet `XMLHttpRequest`. Malheureusement, celui-ci n'est pas disponible dans le navigateur Internet Explorer (alors qu'il l'est dans tous les autres navigateurs principaux). Par conséquent, nous pouvons utiliser un `try` et `catch` pour intercepter ce cas et faire autre chose si la fonction n'est pas disponible. L'exemple 14-12 montre comment faire.

Exemple 14-12. Capture d'une erreur avec try et catch

```
<script>
  try
  {
    requete = new XMLHTTPRequest()
  }
  catch(err)
  {
    // Utiliser une méthode différente pour créer un objet XML HTTP Request
  }
</script>
```


À ce stade, nous n'entrons pas dans les détails de la manière de contourner l'objet absent dans Internet Explorer car l'important est que vous voyiez comment fonctionne ce système. Un autre mot clé est associé à `try` et `catch`, appelé `finally`. Le bloc de code qu'il délimite est toujours exécuté, qu'il y a ait eu une erreur ou non dans la clause `try`. Pour l'utiliser, ajoutez simplement un bloc d'instructions comme le suivant à la suite de l'instruction `catch` :

```
finally
{
  alert("La clause \"try\" a été rencontrée")
}
```

Conditions

Les instructions conditionnelles modifient le flux d'exécution d'un programme. Elles vous permettent de poser des questions à propos de certaines choses et de réagir de manières différentes selon les réponses que vous obtenez. Trois instructions conditionnelles existent en dehors des boucles : l'instruction `if`, l'instruction `switch` et l'opérateur ternaire `?:`.

Instruction if

Plusieurs exemples de ce chapitre ont déjà fait usage d'instructions `if`. Le code d'une telle instruction n'est exécuté que si l'expression donnée s'évalue à `true`. Les conditions `if` qui ont plusieurs instructions à exécuter si la condition est évaluée à `true` ou `false` sur plusieurs lignes nécessitent des accolades pour les entourer mais, comme en PHP, vous pouvez les omettre pour les instructions uniques. Ainsi, les instructions suivantes sont valides :

```
if (a > 100)
{
  b=2
  document.write("a est plus grand que 100")
}

if (b == 10) document.write("b est égal à 10")
```

Instruction else

Quand une condition n'est pas remplie, vous exécutez une voie alternative à l'aide d'une instruction `else`, comme suit :

```
if (a > 100)
{
  document.write("a est plus grand que 100")
}
else
{
  document.write("a est plus petit ou égal à 100")
}
```

Contrairement à PHP, JavaScript ne possède pas d'instruction `elseif`, qui se traduit par « sinon, si », mais ce n'est pas un problème, puisque vous pouvez toujours utiliser `else`, « sinon », suivie d'une autre instruction `if`, « si », ce qui équivaut à l'instruction `elseif`, comme suit :

```
if (a > 100)
{
  document.write("a est plus grand que 100")
}
else if(a < 100)
{
  document.write("a est plus petit que 100")
}
else
{
  document.write("a est égal à 100")
}
```

Comme vous le constatez, vous pouvez enchainer un autre `else` après le nouveau `if`, qui pourrait être aussi bien suivi d'une autre instruction `if`, et ainsi de suite. Si j'ai tenu à afficher des accolades dans les instructions qui ne forment qu'une seule ligne, comme celles-ci ne les exigent pas, l'exemple aurait pu s'écrire comme suit :

```
if     (a > 100) document.write("a est plus grand que 100")
else if(a < 100) document.write("a est plus petit que 100")
else             document.write("a est égal à 100")
```

Instruction switch

L'instruction `switch` s'avère la plus utile lorsqu'une variable ou le résultat d'une expression peut endosser plusieurs valeurs possibles, et que vous voulez exécuter une fonction différente pour chaque valeur.

Par exemple, le code suivant reprend le système de menu PHP que nous avons évoqué au chapitre 4, pour le convertir en JavaScript. Il fonctionne en passant une simple chaine au code principal du menu, selon la demande de l'utilisateur. Supposons que les options soient Accueil, À propos de, Bulletin, Identification et Liens, et que nous définissions la variable `page` à l'une de celles-ci, en fonction de l'entrée de l'utilisateur.

Le code écrit à l'aide d'une construction `if...else if...` peut prendre l'allure de l'exemple 14-13.

Exemple 14-13. Une instruction if...else if... sur plusieurs lignes

```
<script>
  if      (page == "Accueil")       document.write("Vous avez sélectionné Accueil")
  else if (page == "À propos de")   document.write("Vous avez sélectionné À propos de")
  else if (page == "Bulletin")      document.write("Vous avez sélectionné Bulletin")
  else if (page == "Identification") document.write("Vous avez sélectionné Identification")
  else if (page == "Liens")         document.write("Vous avez sélectionné Liens")
</script>
```


Mais cette même construction à l'aide de switch prend l'allure de l'exemple 14-14.

Exemple 14-14. Une construction à l'aide switch

```
<script>
  switch (page)
  {
    case "Accueil":
      document.write("Vous avez sélectionné Accueil")
      break
    case "À propos de":
      document.write("Vous avez sélectionné À propos de")
      break
    case "Bulletin":
      document.write("Vous avez sélectionné Bulletin")
      break
    case "Identification":
      document.write("Vous avez sélectionné Identification")
      break
    case "Liens":
      document.write("Vous avez sélectionné Liens")
      break
  }
</script>
```

La variable `page` n'est mentionnée qu'une seule fois au début de l'instruction `switch`. Ensuite, la commande `case` vérifie chaque correspondance. Lorsqu'une d'elle se produit, l'instruction conditionnelle correspondante est exécutée. Bien entendu, un programme réel contiendrait du code pour afficher ou assurer le lien vers une page, au lieu de simplement dire à l'utilisateur qu'il a fait ceci ou cela.

Rupture

L'exemple 14-14 montre qu'à l'instar de PHP, la commande `break` permet de sortir de la construction `switch` lorsqu'une condition est satisfaite. Rappelez-vous de toujours inclure le `break` à moins que vous ne souhaitiez poursuivre l'exécution aux instructions du `case` suivant.

Action par défaut

Lorsqu'aucune condition n'est satisfaite, vous pouvez définir une action par défaut dans une instruction `switch` à l'aide du mot clé `default`. L'exemple 14-15 montre un extrait de code que vous pouvez insérer dans l'exemple 14-14.

Exemple 14-15. Une instruction par défaut à insérer dans l'exemple 14-14

```
default:
  document.write("Sélection non reconnue")
  break
```

L'opérateur ternaire ?

En combinaison avec le caractère :, l'opérateur ternaire ? fournit un moyen rapide de réaliser rapidement des tests if...else. Il permet d'écrire une expression à évaluer, suivie d'un symbole ? et du code à exécuter si l'expression s'évalue à true. À la suite, insérez un :, puis le code à exécuter si l'expression s'évalue à false.

L'exemple 14-16 montre l'utilisation de l'opérateur ternaire pour afficher si la variable a est inférieure ou égale à 5, ou afficher autre chose si ce n'est le cas.

Exemple 14-16. Utilisation de l'opérateur ternaire

```
<script>
  document.write(
    a <= 5 ?
    "a est plus petit ou égal à 5" :
    "a est plus grand que 5"
  )
</script>
```

Pour les besoins de l'explication, l'instruction est éclatée ici sur plusieurs lignes, mais vous l'écririez plus vraisemblablement sur une seule ligne, dans le genre de ce qui suit :

```
taille = a <= 5 ? "courte" : "longue"
```

Boucles

Dans ce contexte aussi, vous découvrirez de proches ressemblances entre JavaScript et PHP, lorsqu'il s'agit de parler de boucles. Les deux langages prennent en charge les boucles while, do...while et for.

Boucles while

La boucle while (tant que) en JavaScript vérifie d'abord la valeur d'une expression et commence l'exécution des instructions dans la boucle seulement si cette expression est vraie. Si la condition est fausse, l'exécution passe à l'instruction JavaScript suivante (s'il y en a).

Avant d'aborder une itération de la boucle, l'expression est de nouveau testée pour vérifier si elle est vraie, et le processus se poursuit ainsi, jusqu'à ce que l'expression s'évalue à faux ou que l'exécution soit interrompue autrement. L'exemple 14-17 illustre une telle boucle.

Exemple 14-17. Une boucle while

```
<script>
  compteur=0

  while (compteur < 5)
  {
    document.write("compteur : " + compteur + "<br>")
    ++compteur
  }
</script>
```

Le script affiche ce qui suit :

```
Compteur : 0
Compteur : 1
Compteur : 2
Compteur : 3
Compteur : 4
```

Si vous oubliez d'incrémenter la variable compteur dans la boucle, il se peut que certains navigateurs deviennent incontrôlables à cause d'une boucle infinie et que vous ne puissiez pas interrompre la page à l'aide de la touche *Échap.* ou le bouton de fermeture. Donc soyez prudent avec vos boucles en JavaScript.

Boucles do...while

Lorsqu'une boucle doit itérer au moins une fois avant d'effectuer le premier test, utilisez plutôt la boucle do...while, semblable à la boucle while, à l'exception du fait que l'expression de test n'est évaluée qu'à la fin des itérations de la boucle. Ainsi, pour sortir les sept premiers résultats de la table de multiplication de 7, utilisez un code comme celui de l'exemple 14-18.

Exemple 14-18. Une boucle do...while

```
<script>
  compteur = 1

  do
  {
    document.write(compteur + " fois 7 égale " + compteur * 7 + "<br>")
  } while (++compteur <= 7)
</script>
```

Comme vous pouvez vous y attendre, la boucle génère ce qui suit :

```
1 fois 7 égale 7
2 fois 7 égale 14
3 fois 7 égale 21
4 fois 7 égale 28
5 fois 7 égale 35
6 fois 7 égale 42
7 fois 7 égale 49
```

Boucles for

La boucle `for` conjugue le meilleur de tous les mondes dans une construction de boucle simple qui prend trois paramètres dans son instruction :

- Une expression d'initialisation (`expr1`)
- Une expression de condition (`expr2`)
- Une expression de modification (`expr3`)

Des points-virgules séparent ces expressions : `for(expr1 ; expr2 ; expr3)`. Au début, avant toute itération de la boucle, l'expression d'initialisation est exécutée. Dans le cas du script de la table de multiplication de 7, le `compteur` est initialisé à la valeur 1. Ensuite, chaque itération de la boucle entraine le test de l'expression de condition, `compteur <= 7`, et l'exécution du corps de la boucle n'a lieu que si la condition est vraie. Enfin, l'expression de modification est exécutée à la fin de chaque itération. Dans le cas de la table de multiplication de 7, la variable `compteur` est incrémentée. L'exemple 14-19 montre ce que devient le code.

Exemple 14-19. Une boucle for

```
<script>
  for (compteur = 1 ; compteur <= 7 ; ++compteur)
  {
    document.write(compteur + " fois 7 égale " + compteur * 7 + "<br>");
  }
</script>
```

Comme en PHP, il est permis d'affecter plusieurs variables au premier paramètre d'une boucle `for`, à condition de les séparer par des virgules, comme suit :

```
for (i = 1, j = 1 ; i < 10 ; i++)
```

De la même manière, vous pouvez effectuer plusieurs modifications dans le dernier paramètre, comme ceci :

```
for (i = 1 ; i < 10 ; i++, --j)
```

Ou en combinant les deux possibilités :

```
for (i = 1, j = 1 ; i < 10 ; i++, --j)
```

Rupture d'une boucle

La commande `break`, dont vous vous rappelez l'importance dans une instruction `switch`, est également utilisable dans une boucle `for`. Vous l'utilisez par exemple pour rechercher une correspondance d'un genre ou l'autre. Dès que vous constatez la correspondance, vous évitez de poursuivre inutilement la recherche et vous gagnez du temps en sortant immédiatement de la boucle. L'exemple 14-20 illustre un cas d'application de la commande `break`.

Exemple 14-20. Utilisation de la commande break dans une boucle for

```
<script>
  meule_de_foin     = new Array()
  meule_de_foin[17] = "Aiguille"

  for (j = 0 ; j < 20 ; ++j)
  {
    if (meule_de_foin[j] == "Aiguille")
    {
      document.write("<br>- Trouvée à l'enplacement " + j)
      break
    }
    else document.write(j + ", ")
  }
</script>
```

Le script affiche ce qui suit :

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
- Trouvée à l'emplacement 17
```

Instruction continue

Il est parfois nécessaire de ne pas sortir complètement d'une boucle, mais d'éluder le reste des instructions du corps de cette boucle pour l'itération courante et passer directement à la suivante. Dans de tels cas, utilisez la commande `continue`. L'exemple 14-21 en illustre l'utilisation.

Exemple 14-21. Utilisation de la commande continue dans une boucle for

```
<script>
  meule_de_foin     = new Array()
  meule_de_foin[4]  = "Aiguille"
  meule_de_foin[11] = "Aiguille"
  meule_de_foin[17] = "Aiguille"

  for (j = 0 ; j < 20 ; ++j)
  {
    if (meule_de_foin[j] == "Aiguille")
    {
      document.write("<br>- Trouvée à l'enplacement " + j + "<br>")
      continue
    }

    document.write(j + ", ")
  }
</script>
```

Remarquez qu'ici, le deuxième appel de `document.write` n'est plus enfermé dans une instruction `else`, contrairement à l'exemple précédent, car la commande `continue` l'ignore si une correspondance a été trouvée. Les résultats de ce script sont les suivants :

```
0, 1, 2, 3,
- Trouvée à l'emplacement 4
5, 6, 7, 8, 9, 10,
- Trouvée à l'emplacement 11
12, 13, 14, 15, 16,
- Trouvée à l'emplacement 17
18, 19,
```

Conversion de type (cast) explicite

Au contraire de PHP, JavaScript ne possède pas de conversion de type explicite, telle que `(int)` ou `(float)`. Lorsque vous avez besoin d'une valeur d'un type déterminé, utilisez plutôt une des fonctions intégrées de JavaScript énumérées dans le tableau 14-6.

Tableau 14-6. Les fonctions JavaScript de changement de type

Conversion en type	Fonction à utiliser
Int, Integer	`parseInt()`
Bool, Boolean	`Boolean()`
Float, Double, Real	`parseFloat()`
String	`String()`
Array	`split()`

Par exemple, pour convertir un nombre à virgule flottante (`float`) en un entier (`int`), utilisez des instructions du genre suivant (qui affichent la valeur 3) :

```
n = 3.1415927
i = parseInt(n)
document.write(i)
```

Ou, si vous préférez, il y a aussi la version condensée :

```
document.write(parseInt(3.1415927))
```

Voilà qui complète le contrôle de flux d'exécution et les expressions. Le chapitre suivant explore l'utilisation des fonctions, des objets et des tableaux en JavaScript.

Questions

1. Quelle est la différence de traitement des valeurs booléennes entre PHP et JavaScript ?
2. Quels sont les caractères utilisés pour définir les noms de variables en JavaScript ?
3. Quelle est la différence entre les opérateurs unaires, binaires et ternaire ?
4. Quel est le meilleur moyen d'imposer votre propre préséance aux opérateurs ?
5. Quand utilisez-vous l'opérateur d'identité `===` ?
6. Quelles sont les deux formes d'expressions les plus simples ?
7. Nommez les trois types d'instructions conditionnelles.
8. Comment font les instructions `if` et `while` pour interpréter des expressions conditionnelles qui contiennent des types de données différents ?
9. Pourquoi la boucle `for` est-elle plus puissante qu'une boucle `while` ?
10. Quel est le but de l'instruction `with` ?

Retrouvez les réponses du chapitre 14 dans l'annexe A.

CHAPITRE 15

Fonctions, objets et tableaux en JavaScript

Comme PHP, JavaScript offre la possibilité de manipuler des fonctions et des objets. En fait, JavaScript est fondamentalement basé sur des objets car, comme vous l'avez vu, il doit accéder au DOM qui permet de disposer de chaque élément d'un document HTML en tant qu'objet.

L'utilisation et la syntaxe ressemblent fort à celles de PHP donc vous vous sentirez vite en monde connu lors de l'utilisation des fonctions et objets en JavaScript, puis de l'exploration approfondie de la manipulation des tableaux.

Fonctions en JavaScript

En plus de pouvoir accéder à des dizaines de fonctions (et méthodes) intégrées, telles que `write`, que vous connaissez désormais, associée à `document` dans `document.write`, vous pouvez créer vos propres fonctions. Chaque fois que vous avez une portion de code complexe susceptible d'être réutilisée, vous avez un candidat à la définition d'une fonction.

Définir une fonction

Pour définir une fonction, utilisez la syntaxe générale suivante :

```
function nom_fonction ([parametre [, ...]])
{
  instructions
}
```

La première ligne de cette syntaxe indique que :

- Une définition débute par le mot clé `function`.
- Un nom vient ensuite, qui doit commencer par une lettre ou un soulignement, suivi de n'importe quel nombre de lettres, chiffres, symboles dollar ou soulignements.
- Les parenthèses sont obligatoires.
- Un ou plusieurs paramètres, séparés par des virgules, viennent s'insérer de manière facultative entre les parenthèses (les crochets ne font pas partie de la syntaxe mais indiquent que ces paramètres sont facultatifs).

Les noms de fonctions sont sensibles à la casse, donc les noms suivants correspondent à des fonctions différentes : `getInput`, `GETINPUT` et `getinput`.

JavaScript applique une convention générale aux noms de fonctions : la toute première lettre de chaque mot formant le nom s'écrit en capitale, sauf la toute première, en bas de casse. Par conséquent, parmi les exemples ci-dessus, la plupart des programmeurs préfèrent écrire `getInput`. Cette convention porte de jolis noms, comme *camelCase* (casse en forme de chameau), WikiMot ou encore ChatMot (en guise de jeu de mots : *camel* signifie chameau).

L'accolade ouvrante marque le début du corps de la fonction et précède les instructions à exécuter lors de l'appel de la fonction. Une accolade fermante obligatoire marque la fin du corps de la fonction. Ces instructions peuvent compter une ou plusieurs instructions `return`, qui imposent à l'exécution de la fonction de s'arrêter et de retourner au code appelant. Si `return` est suivi d'une valeur, le code appelant peut la récupérer.

Le tableau arguments

Le tableau `arguments` est membre de toute fonction. Il permet de déterminer le nombre de variables passées à une fonction et ce qu'elles sont. Prenons l'exemple d'une fonction dénommée `afficheElements`. L'exemple 15-1 illustre une première manière de la rédiger.

Exemple 15-1. Définition d'une fonction

```
<script>
  afficheElements("Chien", "Chat", "Poney", "Hamster", "Tortue")

  function afficheElements(v1, v2, v3, v4, v5)
  {
    document.write(v1 + "<br>")
    document.write(v2 + "<br>")
    document.write(v3 + "<br>")
    document.write(v4 + "<br>")
    document.write(v5 + "<br>")
  }
</script>
```

Lorsque vous appelez ce script dans un navigateur, il affiche ce qui suit :

```
Chien
Chat
Poney
Hamster
Tortue
```


C'est bien joli tout ça, mais que se passe-t-il si vous décidez de passer plus de cinq arguments à la fonction ? De plus la réutilisation multiple de `document.write` au lieu de l'insérer dans une boucle constitue une pratique de programmation stérile. Heureusement, le tableau `arguments` apporte la souplesse nécessaire pour gérer un nombre variable d'arguments. L'exemple 15-2 illustre son utilisation pour récrire l'exemple d'une manière beaucoup plus efficace.

Exemple 15-2. Récriture de la fonction pour tirer parti du tableau arguments

```
<script>
  function afficheElements()
  {
    for (j = 0 ; j < afficheElements.arguments.length ; ++j)
      document.write(afficheElements.arguments[j] + "<br>")
  }
</script>
```

Remarquez l'utilisation de la propriété `length`, que vous avez déjà vue au chapitre précédent, ainsi que la manière de faire référence au tableau `afficheElements.arguments` à l'aide de la variable `j` pour assurer le décalage dans le tableau et accéder à ses éléments. Enfin, comme il n'y a qu'une seule instruction dans le corps de la boucle `for`, j'ai éludé les accolades qui entourent normalement ce corps.

Grâce à cette technique, vous disposez d'une fonction capable de recevoir autant (et aussi peu) d'arguments que vous le souhaitez et d'agir sur chacun de ces arguments à volonté.

Renvoyer une valeur

Les fonctions ne sont pas utiles qu'à l'affichage. Elles servent essentiellement à effectuer des calculs ou à manipuler des données et renvoyer un résultat. La fonction `nettoyerNoms` de l'exemple 15-3 utilise le tableau `arguments` vu précédemment pour prendre la suite de chaines de caractères qu'elle reçoit et les accoler pour les renvoyer sous la forme d'une seule chaine. La fonction ne se contente pas d'accoler les mots, mais elle en convertit aussi tous les caractères en bas de casse, sauf le premier caractère de chaque argument, mis en capitale.

Exemple 15-3. Nettoyage d'un nom complet

```
<script>
  document.write(nettoyerNoms("les", "CowBoys", "dE", "DALLAS"))

  function nettoyerNoms()
  {
    var s = ""

    for (j = 0 ; j < nettoyerNoms.arguments.length ; ++j)
      s += nettoyerNoms.arguments[j].charAt(0).toUpperCase() +
           nettoyerNoms.arguments[j].substr(1).toLowerCase() + " "
```

```
    return s.substr(0, s.length-1)
  }
</script>
```

Appelée avec les paramètres `les`, `CowBoys`, `dE` et `DALLAS`, par exemple, la fonction renvoie la chaine `Les Cowboys De Dallas`. Examinons le contenu de la fonction.

Elle initialise d'abord la variable temporaire et locale `s` à la chaine vide. La boucle qui suit itère parmi chaque paramètre reçu, isole le premier caractère du paramètre à l'aide de la méthode `charAt` et le convertit en capitale à l'aide de la méthode `toUpperCase`. Les différentes méthodes utilisées dans cet exemple font partie de JavaScript et sont disponibles par défaut.

Ensuite, la méthode `substr` sert à prendre le reste de chaque chaine, convertie en bas de casse à l'aide de la méthode `toLowerCase`. Une version plus complète de la méthode `substr` devrait normalement indiquer le nombre de caractères de la sous-chaine à extraire en second argument, comme suit :

```
substr(1, (arguments[j].length) - 1 )
```

En d'autres termes, cette méthode `substr` dit de « démarrer au caractère à l'emplacement 1 (le deuxième caractère) et renvoyer le reste de la chaine (la longueur moins un) ». Fort judicieusement, la méthode `substr` suppose que vous voulez le reste de la chaine lorsque vous omettez le deuxième argument.

Dès que l'argument est complètement converti à la casse souhaitée, un caractère espace vient s'ajouter à la fin du mot et le résultat est concaténé à la fin de la variable temporaire `s`.

Enfin, la méthode `substr` est utilisée une nouvelle fois pour renvoyer le contenu de la variable `s` mais sans l'espace de fin, non souhaitée. Pour supprimer celle-ci, nous utilisons `substr` pour renvoyer la chaine jusqu'à sa fin, moins le caractère final.

Cet exemple s'avère particulièrement intéressant car il illustre l'utilisation de plusieurs propriétés et méthodes dans une seule expression, par exemple :

```
nettoyerNoms.arguments[j].substr(1).toLowerCase()
```

Pour interpréter cette instruction, découpez-la mentalement en parties à chaque point. JavaScript évalue ces éléments de l'instruction de gauche à droite, comme suit :

1. Débuter avec le nom de la fonction, elle-même, `nettoyerNoms`.
2. Extraire l'élément `j` du tableau arguments qui représente les arguments de `nettoyerNoms`.
3. Appeler `substr` avec le paramètre `1` jusqu'à la fin de l'élément extrait du tableau. Ceci transmet tout sauf le premier caractère à la section suivante de l'expression.
4. Appliquer la méthode `toLowerCase` à la chaine générée à l'instant.

Cette pratique porte souvent le nom de *chainage de méthodes*. Ainsi, si la chaine `CASSEmélangée` est transmise à l'expression que nous venons de détailler, elle passe par les étapes suivantes de transformation :

```
CASSEmélangée
ASSEmélangée
assemélangée
```

Un dernier rappel : la variable `s` créée dans la fonction est locale, donc elle n'est pas accessible en dehors de la fonction. Par le renvoi de `s` par l'instruction `return`, nous rendons sa valeur disponible à l'appelant, qui peut ensuite la mémoriser ou l'utiliser pour autre chose. Mais `s` en elle-même disparait à la fin de l'exécution de la fonction. Bien que nous ayons pu faire en sorte que la fonction agisse sur des variables globales (c'est parfois nécessaire), il est préférable de retourner avec `return` les valeurs à conserver et de laisser JavaScript nettoyer toutes les autres variables utilisées dans la fonction et devenues inutiles.

Renvoyer un tableau

La fonction de l'exemple 15-3 ne retourne qu'un résultat, mais il arrive qu'il faille renvoyer plusieurs résultats. Pour ce faire, retournez un tableau, comme dans l'exemple 15-4.

Exemple 15-4. Renvoi d'un tableau de valeurs

```
<script>
  mots = nettoyerNoms("les", "CowBoys", "dE", "DALLAS")

  for (j = 0 ; j < mots.length ; ++j)
    docunent.write(nots[j] + "<br>")

  function nettoyerNoms()
  {
    var s = new Array()

    for (j = 0 ; j < nettoyerNoms.arguments.length ; ++j)
      s[j] = nettoyerNoms.arguments[j].charAt(0).toUpperCase() +
             nettoyerNoms.arguments[j].substr(1).toLowerCase()

    return s
  }
</script>
```

Ici, la variable `mots` est automatiquement définie comme un tableau et remplie avec le résultat de l'appel à la fonction `nettoyerNoms`. Une boucle parcourt toutes les cellules du tableau et en affiche chacun des membres.

La fonction `nettoyerNoms` est quasi identique à celle de l'exemple 15-3. Elle diffère en ceci : à présent, la variable `s` est un tableau et après chaque traitement d'un mot, le résultat est mémorisé dans un élément de ce tableau. Le tableau est retourné à l'appelant à l'aide de l'instruction `return`.

Cette fonction autorise l'extraction individuelle des informations à partir des valeurs qu'elle retourne, comme dans les instructions suivantes, qui affichent `Les Cowboys` :

```
mots = nettoyerNoms("les", "CowBoys", "dE", "DALLAS")
document.write(mots[0] + " " + mots[1])
```

Objets en JavaScript

En JavaScript, un *objet* représente une étape supérieure par rapport à une variable, qui peut ne contenir qu'une seule valeur au même moment, en ceci que les objets peuvent contenir plusieurs valeurs et même des fonctions. Un objet regroupe des données et les fonctions qui servent à les manipuler.

Déclarer une classe

Lors de la création d'un script destiné à gérer des objets, vous devez concevoir une combinaison de données et de code appelée *classe*. Chaque nouvel objet basé sur cette classe est appelé une *instance* (ou une *occurrence*) de cette classe. Vous avez déjà vu que les données associées à un objet s'appellent ses *propriétés*, tandis que les fonctions qu'il utilise sont appelées ses *méthodes*.

Voyons comment déclarer une classe dénommée `Utilisateur` qui, par l'entremise des objets qui en découleront, contiendra des informations à propos de l'utilisateur courant. Pour créer la classe, écrivez simplement une fonction du même nom que la classe. Cette fonction peut attendre des arguments (nous verrons plus loin comment y faire référence). Nous allons également créer des propriétés et des méthodes dans cette classe. Cette fonction s'appelle le *constructeur* de la classe.

L'exemple 15-5 illustre le constructeur de la classe `Utilisateur`, avec trois propriétés : `prenom`, `nomutilisateur` et `motdepasse`. La classe définit aussi la méthode `afficherUtilisateur`.

Exemple 15-5. Déclaration de la classe Utilisateur et de ses méthodes

```
<script>
  function Utilisateur(prenom, nomutilisateur, motdepasse)
  {
    this.prenom = prenom
    this.nomutilisateur = nomutilisateur
    this.motdepasse = motdepasse

    this.afficherUtilisateur = function()
    {
      document.write("Prénom : " + this.prenom + "<br>")
      document.write("Nom d'utilisateur : " + this.nomutilisateur + "<br>")
      document.write("Mot de passe : " + this.motdepasse + "<br>")
    }
  }
</script>
```


Cette fonction diffère des autres fonctions vues précédemment à deux niveaux :

- Elle fait référence à un objet nommé `this` (ceci). Lorsqu'un programme crée une instance d'`Utilisateur` par l'exécution de cette fonction, `this` se réfère à l'instance en cours de création. Vous pouvez appeler la fonction plusieurs fois avec des arguments différents et elle crée chaque fois un `Utilisateur` avec des valeurs différentes pour la propriété `prenom` et les suivantes.
- Une nouvelle fonction `afficherUtilisateur` est créée au sein de la fonction. Il s'agit d'une syntaxe nouvelle et assez complexe, mais son rôle est de lier `afficherUtilisateur` en tant que méthode de la classe `Utilisateur`.

Les conventions de nommage que j'adopte consistent à placer toutes les propriétés en bas de casse et d'utiliser au moins une lettre capitale dans les noms de méthodes, suivant la convention des WikiMots (*camelCase*) évoquée au début de ce chapitre.

L'exemple 15-5 respecte la manière recommandée d'écrire un constructeur de classe, qui consiste à insérer les méthodes dans la fonction du constructeur. Il est toutefois possible de faire référence à des fonctions définies en dehors du constructeur, comme dans l'exemple 15-6.

Exemple 15-6. Définitions distinctes d'une classe et d'une méthode

```
<script>
  function Utilisateur(prenom, nomutilisateur, motdepasse)
  {
    this.prenom = prenom
    this.nomutilisateur = nomutilisateur
    this.motdepasse = motdepasse
    this.afficherUtilisateur = afficherUtilisateur
  }

  function afficherUtilisateur()
  {
    document.write("Prénom : " + this.prenom + "<br>")
    document.write("Nom d'utilisateur : " + this.nomutilisateur + "<br>")
    document.write("Mot de passe : " + this.motdepasse + "<br>")
  }
</script>
```

Si je vous montre cette forme, c'est parce que tôt ou tard, vous la rencontrerez dans le code d'un autre programmeur.

Créer un objet

Pour créer une instance de la classe `Utilisateur`, utilisez une instruction du style de la suivante :

```
details = new Utilisateur("Wolfgang", "w.a.mozart", "compositeur")
```

Mais vous pouvez très bien créer un objet vide, comme suit :

```
details = new Utilisateur()
```

Et le remplir par la suite, comme ceci :

```
details.prenon         = "Wolfgang"
details.nomutilisateur = "w.a.mozart"
details.motdepasse     = "compositeur"
```

Vous pouvez aussi ajouter de nouvelles propriétés à un objet existant, comme suit :

```
details.salutation = "Bonjour"
```

Pour vérifier si la nouvelle propriété fonctionne, écrivez :

```
document.write(details.salutation)
```

Accéder aux objets

Pour accéder à un objet, faites référence à ses propriétés, comme dans les deux instructions suivantes, sans aucun rapport entre elles :

```
nom = details.prenon
if (details.nomutilisateur == "Admin") connecterComeAdmin()
```

Ainsi, pour accéder à la méthode `afficherUtilisateur` d'un objet de la classe `Utilisateur`, empruntez la syntaxe suivante, où l'objet `details` est supposé déjà créé et rempli de données :

```
details.afficherUtilisateur()
```

En admettant que les données soient celles déjà décrites, le code affiche ce qui suit :

```
Prénom : Wolfgang
Nom d'utilisateur : w.a.mozart
Mot de passe : compositeur
```

Mot clé prototype

Le mot clé `prototype` permet parfois d'économiser beaucoup de mémoire. Dans la classe `Utilisateur`, chaque instance contient les trois propriétés et la méthode. Par conséquent, si vous gérez 1 000 exemplaires de cet objet en mémoire, la méthode `afficherUtilisateur` est également répliquée 1 000 fois. Or, comme cette méthode est identique dans chaque instance, vous pouvez indiquer à chaque nouvel objet qu'il doit faire référence à une seule instance de la méthode au lieu d'en créer une copie. Pour cela, au lieu d'utiliser ce qui suit dans un constructeur de classe,

```
this.afficherUtilisateur = function()
```

remplacez-le par ceci :

```
Utilisateur.prototype.afficherUtilisateur = function()
```

L'exemple 15-7 illustre l'allure du nouveau constructeur.

Exemple 15-7. Déclaration d'une classe avec le mot clé prototype pour une méthode

```
<script>
  function Utilisateur(prenom, nomutilisateur, motdepasse)
  {
    this.prenom = prenom
    this.nomutilisateur = nomutilisateur
    this.motdepasse = motdepasse

    Utilisateur.prototype.afficherUtilisateur = function()
    {
      document.write("Prénom : " + this.prenon + "<br>")
      document.write("Nom d'utilisateur : " + this.nomutilisateur + "<br>")
      document.write("Mot de passe : " + this.motdepasse + "<br>")
    }
  }
</script>
```

Ceci fonctionne parce que toutes les fonctions possèdent la propriété `prototype`, conçue pour conserver des propriétés et des méthodes à ne répliquer dans aucun des objets créés à partir d'une classe. À la place, elles sont transmises à ses objets par référence.

Cela signifie que vous pouvez ajouter une propriété ou une méthode `prototype` à tout moment et tous les objets (même ceux déjà créés) en héritent, comme l'illustrent les instructions suivantes :

```
Utilisateur.prototype.salutation = "Bonjour"
document.write(details.salutation)
```

La première instruction ajoute à la classe `Utilisateur` la propriété `salutation`, sous forme de `prototype`, avec la valeur `Bonjour`. À la seconde ligne, l'objet `details`, déjà créé, affiche correctement cette nouvelle propriété.

Vous pouvez aussi ajouter et modifier des méthodes dans une classe, comme l'illustrent les instructions suivantes :

```
Utilisateur.prototype.afficherUtilisateur = function()
{
  document.write("Nom " + this.prenon +
                 " Utilisateur " + this.nomutilisateur +
                 " Passe " + this.motdepasse)
}

details.afficherUtilisateur()
```

Vous pourriez ajouter ces lignes à votre script dans une instruction conditionnelle (comme `if`) pour qu'elles s'exécutent si les activités de l'utilisateur vous motivent à emprunter une méthode `afficherUtilisateur` différente. Après l'exécution de ces lignes, même si l'objet `details` a déjà été créé, les appels subséquents à `details.afficherUtilisateur` exécuteront cette nouvelle fonction. L'ancienne définition d'`afficherUtilisateur` est écrasée et supprimée.

Méthodes et propriétés statiques

Lorsque vous avez étudié les objets en PHP, vous avez appris que les classes peuvent posséder des propriétés et méthodes statiques, ainsi que des propriétés et méthodes spécifiques associées à une seule instance d'une classe. JavaScript prend également en charge les propriétés et méthodes statiques, que vous pouvez mémoriser et retrouver à votre guise à partir du `prototype` de la classe. Ainsi, les instructions suivantes définissent et lisent une chaine statique dans `Utilisateur` :

```
Utilisateur.prototype.salutation = "Bonjour"
document.write(Utilisateur.prototype.salutation)
```

Extension des objets en JavaScript

Le mot clé `prototype` permet même d'ajouter des fonctionnalités à un objet intégré de JavaScript. Supposons par exemple que vous voulez ajouter la possibilité de remplacer toutes les espaces d'une chaine par des espaces insécables (` ` en HTML) pour l'empêcher d'en renvoyer des mots à la ligne. Pour ce faire, ajoutez une méthode prototype à la définition JavaScript par défaut de l'objet `String`, comme suit :

```
String.prototype.nbsp = function()
{
  return this.replace(/ /g, ' ')
}
```

La méthode `replace` est utilisée ici avec une expression régulière (voir chapitre 16) pour chercher et remplacer chaque espace seule par la chaine ` `. Entrez ensuite la commande suivante :

```
document.write("Voix ambigüe d'un cœur".nbsp())
```

Cette instruction sort la chaine `Voix ambigüe d'un cœur`. Voici encore une autre méthode que vous pouvez ajouter et qui permet de supprimer les espaces (*trim*) au début et à la fin d'une chaine, à nouveau à l'aide d'une expression régulière :

```
String.prototype.trim = function()
{
  return this.replace(/^\s+|\s+$/g, '')
}
```

Lorsque vous envoyez l'instruction suivante, la sortie donne `S'il-vous-plait, élaguez-moi`, avec les espaces de début et de fin supprimées.

```
document.write("   S'il-vous-plait, élaguez-moi      ".trim())
```

La décomposition de l'expression régulière en ses composantes permet de voir que les deux caractères `/` désignent respectivement le début et la fin de l'expression, tandis que le `g` final indique une recherche globale. Dans l'expression entre les `/`, la partie `^\s+` recherche un ou plusieurs caractères espace apparaissant au début de la chaine inspectée, tandis que la partie `\s+$` recherche une ou plusieurs espaces à la fin de la chaine inspectée. Le caractère `|` au milieu sert à distinguer les alternatives.

Le résultat de cette méthode ajoutée donne le remplacement par une chaine vide pour toute correspondance de l'expression, donc, globalement, une version élaguée de la chaine, sans espace de début ni de fin.

Tableaux en JavaScript

La gestion des tableaux (*arrays*) en JavaScript s'apparente fortement à celle en PHP, bien que les syntaxes diffèrent légèrement. Quoi qu'il en soit, sachant que vous avez déjà étudié les tableaux en PHP, cette section devrait vous paraitre évidente.

Tableaux numériques

Pour créer un tableau, utilisez la syntaxe suivante :

```
nomtableau = new Array()
```

Une forme abrégée existe aussi :

```
nomtableau = []
```

Affecter des valeurs aux éléments d'un tableau

En PHP, vous pouvez ajouter un nouvel élément à un tableau par simple affectation à cet élément, sans préciser le décalage de l'élément, comme suit :

```
$nomtableau[] = "Élément 1";
$nomtableau[] = "Élément 2";
```

Mais, en JavaScript, vous devez faire appel à la méthode `push` (pousser) pour obtenir le même résultat, comme ceci :

```
nomtableau.push("Élément 1")
nomtableau.push("Élément 2")
```

Ceci permet d'ajouter des éléments à un tableau sans devoir tenir compte du nombre d'éléments. Lorsque vous souhaitez connaitre le nombre d'éléments présents dans un tableau, faites appel à sa propriété `length`, comme suit :

```
document.write(nomtableau.length)
```

Si vous souhaitez plutôt suivre les emplacements des éléments et les placer dans des emplacements déterminés, utilisez la syntaxe suivante :

```
nomtableau[0] = "Élément 1";
nomtableau[1] = "Élément 2";
```

L'exemple 15-8 montre un petit script qui crée un tableau, y charge quelques valeurs, puis les affiche.

Exemple 15-8. Création, remplissage et affichage d'un tableau

```
<script>
  nombres = []
  nombres.push("Un")
  nombres.push("Deux")
  nombres.push("Trois")

  for (j = 0 ; j < nombres.length ; ++j)
    document.write("Élément " + j + " = " + nombres[j] + "<br>")
</script>
```

Lors de son exécution, ce script produit les résultats suivants :

```
Élément 0 = Un
Élément 1 = Deux
Élément 2 = Trois
```

Affectation à l'aide du mot clé Array

Une autre technique permet de créer rapidement un tableau défini avec quelques éléments initiaux, à l'aide du mot clé `Array`, comme suit :

```
nombres = Array("Un", "Deux", "Trois")
```

L'écriture abrégée suivante demeure toutefois la plus utilisée :

```
nombres = ["Un", "Deux", "Trois"]
```

Rien ne vous empêche d'ajouter d'autres éléments par la suite.

Vous disposez donc de quelques possibilités pour ajouter des éléments à un tableau et d'une technique pour y faire référence, mais JavaScript en offre encore bien d'autres, que nous allons examiner sous peu. Nous allons cependant nous intéresser d'abord à un autre type de tableau.

Tableaux associatifs

Un *tableau associatif* est un tableau dont les éléments sont référencés par des noms au lieu d'indices numériques. Pour créer un tableau associatif, définissez un bloc d'éléments entre accolades. Ensuite, pour chaque élément, placez la clé à gauche et le contenu à droite d'un deux-points (:). L'exemple 15-9 montre comment créer un tableau associatif pour contenir le rayon « balles » d'un vendeur en ligne d'équipements sportifs.

Exemple 15-9. Création et déploiement d'un tableau associatif

```
<script>
  balles = {"golf":   "Balles de golf, 6",
            "tennis": "Balles de tennis, 3",
            "foot":   "Ballon de football, 1",
            "ping":   "Balles de tennis de table, 12"}

  for (balle in balles)
    document.write(balle + " = " + balles[balle] + "<br>")
</script>
```


Pour vérifier la création et le remplissage du tableau, nous utilisons ici une forme particulière de la boucle `for`, qui fait appel au mot clé `in`. Celui-ci permet de créer une variable `balle` dont l'utilisation n'est possible que dans le tableau et qui assure l'itération parmi tous les éléments du tableau indiqué à droite du mot clé `in`, c'est-à-dire `balles` dans l'exemple. La boucle agit individuellement sur chaque élément de `balles`. Les résultats de l'appel du script de l'exemple dans un navigateur sont les suivants :

```
golf = Balles de golf, 6
tennis = Balles de tennis, 3
foot = Ballon de football, 1
ping = Balles de tennis de table, 12
```

Pour obtenir un élément déterminé d'un tableau associatif, vous pouvez indiquer la clé recherchée, de la manière suivante (le résultat est `Ballon de football, 1`) :

```
document.write(balles['foot'])
```

Tableaux multidimensionnels

La création d'un tableau multidimensionnel se résume à insérer des tableaux dans un autre tableau. Ainsi, le code de l'exemple 15-10 suivant crée le tableau nécessaire pour contenir un damier bidimensionnel avec ses pions (de 10 × 10 cases) :

Exemple 15-10. Création d'un tableau numérique bidimensionnel

```
<script>
  damier = Array(
    Array(' ', 'o', ' ', 'o', ' ', 'o', ' ', 'o', ' ', 'o'),
    Array('o', ' ', 'o', ' ', 'o', ' ', 'o', ' ', 'o', ' '),
    Array(' ', 'o', ' ', 'o', ' ', 'o', ' ', 'o', ' ', 'o'),
    Array('o', ' ', 'o', ' ', 'o', ' ', 'o', ' ', 'o', ' '),
    Array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
    Array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
    Array(' ', '0', ' ', '0', ' ', '0', ' ', '0', ' ', '0'),
    Array('0', ' ', '0', ' ', '0', ' ', '0', ' ', '0', ' '),
    Array(' ', '0', ' ', '0', ' ', '0', ' ', '0', ' ', '0'),
    Array('0', ' ', '0', ' ', '0', ' ', '0', ' ', '0', ' '))
  document.write("<pre>")

  for (j = 0 ; j < 10 ; ++j)
  {
    for (k = 0 ; k < 10 ; ++k)
      document.write(damier[j][k] + " ")

    document.write("<br>")
  }

  document.write("</pre>")
</script>
```

Dans cet exemple, les lettres en bas de casse représentent les pions noirs, tandis que les lettres capitales représentent les pions blancs. Deux boucles `for` suivent la déclaration du damier, parcourent le tableau et en affichent le contenu.

La boucle extérieure contient deux instructions, donc des accolades doivent les entourer. La boucle intérieure traite ensuite chaque case d'une ligne et affiche le caractère à l'emplacement `[j][k]`, suivi d'une espace, pour décaler l'affichage. Cette boucle contient une seule instruction, donc les accolades ne sont pas obligatoires pour l'entourer. Les balises `<pre>` et `</pre>` garantissent l'affichage adéquat des espaces, comme suit :

```
  o  o  o  o
o  o  o  o
  o  O  o  o


o  o  o  o
  o  o  o  o
o  o  o  o
```

À partir de là, vous pouvez accéder à n'importe quel élément défini dans le tableau, à l'aide des crochets :

```
document.write(damier[7][2])
```

Cette instruction sort la lettre capitale `O`, soit le huitième élément vers le bas et le troisième élément à partir de la gauche. Rappelez-vous que les indices débutent à zéro.

Utiliser les méthodes des tableaux

Étant donné la puissance des tableaux, JavaScript fournit d'emblée un certain nombre de méthodes pour les manipuler ainsi que leurs données. Voici une sélection des plus utiles.

Utiliser concat

La méthode `concat` concatène deux tableaux ou séries de valeurs au sein d'un tableau. Par exemple, le code suivant affiche **`Banane, Raisin, Carotte, Chou`** :

```
fruit  = ["Banane", "Raisin"]
legume = ["Carotte", "Chou"]

document.write(fruit.concat(legume))
```

Vous pouvez aussi indiquer plusieurs tableaux en arguments, auquel cas `concat` ajoute tous leurs éléments dans l'ordre d'indication des tableaux.

Une autre manière d'utiliser `concat` permet de concaténer à un tableau, `animaux` par exemple, des valeurs discrètes (c'est-à-dire littérales). L'exemple suivant affiche **`Chat, Chien, Poisson, Lapin, Hamster`** :

```
animaux = ["Chat", "Chien", "Poisson"]
autres_animaux = animaux.concat("Lapin", "Hamster")

document.write(autres_animaux)
```


Utiliser forEach

La méthode `forEach` (pour chaque) en JavaScript offre une autre manière d'assurer une fonctionnalité comparable à la méthode `foreach` en PHP. Elle fonctionne sur les navigateurs web, sauf *Internet Explorer 9* et versions antérieures. Depuis la version 10 de IE, elle fonctionne sur tous les navigateurs. Pour l'utiliser, donnez-lui le nom d'une fonction, appelée ensuite pour chaque élément du tableau. L'exemple 15-11 montre comment faire.

Exemple 15-11. Utilisation de la méthode forEach

```
<script>
  animaux = ["Chat", "Chien", "Lapin", "Hamster"]
  animaux.forEach(afficher)

  function afficher(element, indice, tableau)
  {
    document.write("L'élément d'indice " + indice + " a la valeur " +
      element + "<br>")
  }
</script>
```

Ici, la fonction passée à `forEach` s'appelle `afficher`. Elle prend trois paramètres : la valeur de l'élément `element`, son `indice` et le `tableau`. La fonction les utilise à son gré. Dans cet exemple, seules les valeurs `element` et `indice` servent à l'affichage par la fonction `document.write`.

Dès que les valeurs sont venues remplir le tableau, l'appel de la méthode s'effectue comme suit :

```
animaux.forEach(afficher)
```

Voici les résultats :

```
L'élément d'indice 0 a la valeur Chat
L'élément d'indice 1 a la valeur Chien
L'élément d'indice 2 a la valeur Lapin
L'élément d'indice 3 a la valeur Hamster
```

Utiliser forEach (solution valable aussi pour Internet Explorer 9 et antérieures)

Microsoft avait choisi de ne pas prendre en charge la méthode `forEach` dans les versions d'IE 9 et antérieures, donc l'exemple précédent ne fonctionne pas sur ces navigateurs. Par conséquent, pour garantir la compatibilité quel que soit le navigateur, même plus ancien, vous pouvez utiliser une instruction telle que la suivante, à la place d'`animaux.forEach(afficher)` :

```
for (j = 0 ; j < animaux.length ; ++j) afficher(animaux[j], j)
```

Utiliser join

La méthode `join` (joindre) permet de convertir toutes les valeurs contenues dans un tableau en chaine, puis de les concaténer en une grande chaine, en plaçant un éventuel séparateur facultatif entre ces éléments. L'exemple 15-12 illustre trois façons d'utiliser cette méthode.

Exemple 15-12. Utilisations de la méthode join

```
<script>
  animaux = ["Chat", "Chien", "Lapin", "Hamster"]

  document.write(animaux.join()      + "<br>")
  document.write(animaux.join(' ')   + "<br>")
  document.write(animaux.join(' : ') + "<br>")
</script>
```

Sans aucun paramètre, `join` utilise une virgule pour séparer les éléments ; sinon la chaine de séparation passée à `join` vient s'insérer entre chacun des arguments. Le résultat de l'exemple 15-12 prend l'allure suivante :

```
Chat,Chien,Lapin,Hamster
Chat Chien Lapin Hamster
Chat : Chien : Lapin : Hamster
```

Utiliser push et pop

Vous savez déjà que la méthode `push` permet d'insérer une valeur dans un tableau. La méthode inverse s'appelle `pop`. Elle supprime le dernier élément dans un tableau et le renvoie. L'exemple 15-13 en illustre l'utilisation.

Exemple 15-13. Utilisation des méthodes push et pop

```
<script>
  sports = ["Football", "Tennis", "Baseball"]
  document.write("Début = "      + sports +   "<br>")

  sports.push("Hockey");
  document.write("Après Push = " + sports +   "<br>")

  supprime = sports.pop()
  document.write("Après Pop = "  + sports +   "<br>")
  document.write("Supprimé = "   + supprime + "<br>")
</script>
```

Dans ce script, les trois principales instructions sont mises en évidence en gras. D'abord, le script crée un tableau dénommé `sports` contenant trois éléments, puis il « pousse » (`push`) un quatrième élément dans le tableau. Ensuite, il extrait (`pop`) cet élément hors du tableau. Au cours du processus, les différentes valeurs viennent s'afficher à l'aide de `document.write`. Les résultats de ce script sont les suivants :

```
Début = Football,Tennis,Baseball
Après Push = Football,Tennis,Baseball,Hockey
Après Pop = Football,Tennis,Baseball
Supprimé = Hockey
```

Les méthodes `push` et `pop` s'avèrent utiles dans des situations qui nécessitent de vous détourner d'une certaine activité pour vous livrer à une autre, puis de revenir à la précédente, comme dans l'exemple 15-14.

Exemple 15-14. Utilisation de push et de pop à l'intérieur et à l'extérieur d'une boucle

```
<script>
  nombres = []

  for (j=0 ; j<3 ; ++j)
  {
    nombres.push(j);
    document.write("Pousse " + j + "<br>")
  }

  // Exécution d'une autre activité ici
  document.write("<br>")

  document.write("Extrait " + nombres.pop() + "<br>")
  document.write("Extrait " + nombres.pop() + "<br>")
  document.write("Extrait " + nombres.pop() + "<br>")
</script>
```

Ce script produit les résultats suivants :

```
Pousse 0
Pousse 1
Pousse 2

Extrait 2
Extrait 1
Extrait 0
```

Utiliser reverse

La méthode `reverse` inverse l'ordre de tous les éléments d'un tableau. L'exemple 15-15 la montre en action.

Exemple 15-15. Utilisation de la méthode reverse

```
<script>
  sports = ["Football", "Tennis", "Baseball", "Hockey"]
  sports.reverse()
  document.write(sports)
</script>
```

Le tableau initial subit l'inversion, de sorte que ce script produit ce qui suit :

```
Hockey,Baseball,Tennis,Football
```

Utiliser sort

La méthode `sort` (trier) permet de placer tous les éléments d'un tableau dans l'ordre alphabétique ou un autre, selon les paramètres passés. L'exemple 15-16 illustre quatre types de classements.

Exemple 15-16. Utilisation de la méthode sort

```
<script>
  // Tri en ordre alphabétique croissant
  sports = ["Football", "Tennis", "Baseball", "Hockey"]
  sports.sort()
  document.write(sports + "<br>")

  // Tri en ordre alphabétique décroissant
  sports = ["Football", "Tennis", "Baseball", "Hockey"]
  sports.sort().reverse()
  document.write(sports + "<br>")

  // Tri en ordre numérique croissant
  nombres = [7, 23, 6, 74]
  nombres.sort(function(a,b){return a - b})
  document.write(nombres + "<br>")

  // Tri en ordre numérique décroissant
  nombres = [7, 23, 6, 74]
  nombres.sort(function(a,b){return b - a})
  document.write(nombres + "<br>")
</script>
```

La première des quatre sections d'exemple applique la méthode de tri `sort` par défaut, c'est-à-dire le *tri alphabétique croissant*, la deuxième section utilise la méthode `sort` par défaut, puis lui applique la méthode `reverse` pour obtenir le *tri alphabétique décroissant*.

Les troisième et quatrième sections présentent un peu plus de complexité, parce qu'elles utilisent une fonction pour comparer les relations entre `a` et `b`. Cette fonction n'a pas de nom, parce qu'elle ne peut s'utiliser que dans le cadre du tri. Vous avez déjà vu la fonction nommée `function` pour créer une *fonction anonyme* car nous l'avons utilisée pour définir une méthode dans une classe (la méthode `afficherUtilisateur`).

Ici, `function` sert à créer une fonction anonyme pour fournir à `sort` ce dont elle a besoin. Si la fonction renvoie une valeur supérieure à zéro, alors le tri en déduit que `b` vient avant `a`. Si la fonction renvoie une valeur inférieure à zéro, alors le tri en déduit que `a` vient avant `b`. Le tri exécute cette fonction de proche en proche sur toutes les valeurs du tableau pour déterminer leur ordre.

La manipulation de la valeur renvoyée (`a - b` au contraire de `b - a`) permet aux troisième et quatrième sections de l'exemple de choisir entre un *tri en ordre numérique croissant et un tri en ordre numérique décroissant*.

Et croyez-le ou non, ceci constitue la fin de notre présentation de JavaScript. Vous disposez à présent des connaissances de base des trois principales technologies exposées dans ce livre. Le chapitre suivant examine quelques techniques avancées exploitées parmi ces technologies, comme le filtrage par motif et la validation d'entrée.

Questions

1. En JavaScript, les noms des variables et de fonctions sont-ils sensibles ou non à la casse ?
2. Comment faites-vous pour écrire une fonction qui accepte et traite un nombre illimité de paramètres ?
3. Citez un moyen pour renvoyer plusieurs valeurs à partir d'une fonction.
4. Lorsque vous définissez une classe, quel mot clé employez-vous pour faire référence à l'objet courant ?
5. Toutes les méthodes doivent-elles être définies obligatoirement au sein de la définition de la classe ?
6. Quel mot clé permet de créer un objet ?
7. Comment faites-vous pour rendre une propriété ou une méthode disponible à tous les objets issus d'une classe, sans répliquer cette propriété ou cette méthode au sein des objets ?
8. Comment créer un tableau multidimensionnel ?
9. Quelle syntaxe permet de créer un tableau associatif ?
10. Écrivez une instruction pour trier un tableau de nombres par ordre numérique décroissant.

Retrouvez les réponses du chapitre 15 dans l'annexe A.

CHAPITRE 16

Validation et gestion d'erreur en PHP et JavaScript

Maintenant que vous avez acquis une solide base en PHP et en JavaScript, il est temps de rassembler ces technologies pour créer des formulaires web qui soient aussi conviviaux que possible.

Nous utilisons PHP pour créer les formulaires et JavaScript pour gérer la validation du côté du client et vérifier que les données soient aussi complètes et exactes qu'attendues, avant de les soumettre. La validation finale des entrées est prise en charge par PHP, qui doit, si nécessaire, présenter à nouveau le formulaire à l'utilisateur afin qu'il y effectue les modifications éventuellement nécessaires.

Dans ce processus, ce chapitre intervient pour examiner la validation et les expressions rationnelles tant en JavaScript qu'en PHP.

Valider les entrées utilisateur à l'aide de JavaScript

En fait, la validation en JavaScript doit être considérée comme une assistance destinée plutôt aux utilisateurs qu'à vos sites web parce que, comme je l'ai déjà évoqué à maintes reprises, vous ne pouvez faire confiance aux données soumises à votre serveur, même si elles sont supposées avoir été validées dans JavaScript. Ce souci trouve sa source dans le fait que c'est un jeu d'enfant pour un pirate de simuler vos formulaires web et de soumettre n'importe quelle donnée de son choix.

Une autre raison qui motive l'absence totale de confiance en JavaScript pour effectuer toutes les validations d'entrées est que certains utilisateurs désactivent JavaScript et que certains navigateurs ne le prennent tout simplement pas en charge.

Par conséquent, les meilleurs types de validations à effectuer en JavaScript se limitent à vérifier que les champs qui doivent avoir un contenu en ont effectivement un, que les adresses de courrier électronique sont conformes au format attendu et que les valeurs entrées respectent les limites établies.

Le document validate.html (1re partie)

Commençons par un formulaire d'identification général, commun à la plupart des sites, qui proposent une inscription ou une identification d'utilisateurs. Les entrées requises sont le prénom (`forename`), le nom de famille (`surname`), le nom d'utilisateur (`username`), le mot de passe (`password`), l'âge (`age`) et l'adresse de courrier électronique (`email`). L'exemple 16-1 propose un modèle approprié pour ce genre de formulaire.

Dans ce script et toutes les évolutions qui en découlent dans la suite de ce chapitre, nous affichons en français les informations destinées à l'utilisateur mais tout le code, le « moteur sous le capot », est en anglais. N'oubliez pas que tôt ou tard, vous récupérerez du code d'autres programmeurs ou vous devrez accéder à des forums pour obtenir de l'aide, or la langue internationale d'usage est essentiellement l'anglais. Un développeur de bonne volonté et décidé à vous aider, qu'il soit Italien, Pakistanais ou Espagnol, comprendra plus facilement votre code si tous les mécanismes y sont en anglais.

Exemple 16-1. Un formulaire avec sa validation en JavaScript (1re partie)

```
<!DOCTYPE html>
<html>
  <head>
    <title>Exemple de formulaire</title>
    <style>
      .signup {
        border:1px solid #999999;
        font:  normal 14px helvetica;
        color: #444444;
      }
    </style>
    <script>
      function validate(form)
      {
        fail  = validateForename(form.forename.value)
        fail += validateSurname(form.surname.value)
        fail += validateUsername(form.username.value)
        fail += validatePassword(form.password.value)
        fail += validateAge(form.age.value)
        fail += validateEmail(form.email.value)

        if   (fail == "")   return true
        else { alert(fail); return false }
      }
    </script>
  </head>
  <body>
    <table class="signup" border="0" cellpadding="2" cellspacing="5"
      bgcolor="#eeeeee">
      <th colspan="2" align="center">Formulaire d'inscription</th>
```



```
      <form method="post" action="adduser.php" onsubmit="return validate(this)">
        <tr><td>Prénom</td>
          <td><input type="text" maxlength="32" name="forename"></td></tr>
        <tr><td>Nom</td>
          <td><input type="text" maxlength="32" name="surname"></td></tr>
        <tr><td>Nom d'utilisateur</td>
          <td><input type="text" maxlength="16" name="username"></td></tr>
        <tr><td>Mot de passe</td>
          <td><input type="text" maxlength="12" name="password"></td></tr>
        <tr><td>Âge</td>
          <td><input type="text" maxlength="3"  name="age"></td></tr>
        <tr><td>Courriel</td>
          <td><input type="text" maxlength="64" name="email"></td></tr>
        <tr><td colspan="2" align="center"><input type="submit"
          value="Inscription"></td></tr>
      </form>
    </table>
  </body>
</html>
```

À ce stade, le formulaire s'affiche correctement mais ne peut pas se valider par lui-même, parce que les fonctions de validation n'y sont pas encore présentes. Même dans cet état, enregistrez le fichier sous le nom *validate.html* et lorsque vous l'appelez dans votre navigateur, il prend l'allure de la figure 16-1.

Figure 16-1. L'écran généré par l'exemple 16-1

Voyons comment ce document est structuré. Les quelques premières lignes définissent le document et utilisent un peu de CSS pour donner au formulaire un aspect moins ordinaire. La partie spécifique à JavaScript vient ensuite et est présentée en gras dans le code.

Entre les balises <script> et </script> apparait une seule fonction nommée validate, qui appelle à son tour six autres fonctions pour valider chacun des champs (*fields*) d'entrée du formulaire. Nous examinerons ces fonctions sous peu. Pour l'instant, disons qu'elles renvoient chacune soit une chaine vide si le champ est correct, soit un message d'erreur si la validation échoue. Ces messages s'accumulent dans la variable fail (échec). Si la moindre erreur apparait (donc si fail n'est pas vide), la dernière ligne affiche une boite de dialogue d'alerte avec le contenu de fail.

Si toutes les validations réussissent, la fonction validate retourne la valeur true ; sinon, elle retourne false. Ces valeurs de retour de validate revêtent une importance particulière car, si elle renvoie false, le formulaire empêche son envoi en soumission. Cela permet à l'utilisateur de fermer la fenêtre d'alerte et d'effectuer les modifications nécessaires. Si validate renvoie true, indiquant qu'aucune erreur n'a été détectée parmi les champs du formulaire, alors le formulaire autorise la soumission.

La partie suivante de cet exemple présente le HTML du formulaire et range chacun des champs avec son nom dans une ligne d'un tableau. Ce code HTML est assez évident à comprendre, sauf l'instruction onSubmit="return validate(this)" dans la balise ouvrante <form>. L'utilisation de onSubmit (évènement « à la soumission ») permet d'appeler une fonction de votre choix au moment de la soumission du formulaire. Cette fonction vise à effectuer des vérifications et renvoie la valeur true ou false pour indiquer que le formulaire peut ou non être envoyé en soumission.

Le paramètre this correspond à l'objet courant, c'est-à-dire le formulaire en cours, et est donc passé à la fonction de validation, qui le reçoit dans son paramètre form.

Ainsi donc, la seule portion de JavaScript employée dans ce formulaire se résume à l'appel de return, inséré dans l'attribut onSubmit. Les navigateurs où JavaScript est désactivé ou qui ne prennent pas en charge JavaScript ignorent tout simplement l'attribut onSubmit et le HTML s'y affiche tout aussi bien.

Le document validate.html (2e partie)

Nous abordons à présent l'exemple 16-2 qui reprend un ensemble de six fonctions en charge de la validation réelle des champs du formulaire. Je vous suggère d'entrer la totalité de cette seconde partie à la suite du code entre les balises <script> et </script> de l'exemple 16-1, que vous avez déjà enregistré dans *validate.html*.

Exemple 16-2. Un formulaire avec sa validation en JavaScript (2e partie)

```
function validateForename(field)
{
  return (field == "") ? "Entrez un prénom.\n" : ""
}

function validateSurname(field)
{
  return (field == "") ? "Entrez un nom de famille.\n" : ""
}
```



```
function validateUsername(field)
{
  if (field == "") return "Entrez un nom d'utilisateur.\n"
  else if (field.length < 5)
    return "Le nom d'utilisateur doit contenir au moins 5 caractères.\n"
  else if (/[^a-zA-Z0-9_-]/.test(field))
    return "Seuls caractères permis dans le nom d'utilisateur : " +
           "a-z, A-Z, 0-9, - et _.\n"
  return ""
}

function validatePassword(field)
{
  if (field == "") return "Entrez un mot de passe.\n"
  else if (field.length < 6)
    return "Le mot de passe doit contenir au moins 6 caractères.\n"
  else if (! /[a-z]/.test(field) ||
           ! /[A-Z]/.test(field) ||
           ! /[0-9]/.test(field))
    return "Le mot de passe doit contenir au moins un a-z, A-Z et 0-9.\n"
  return ""
}

function validateAge(field)
{
  if (field == "" || isNaN(field)) return "Entrez l'âge en chiffres.\\n"
  else if (field < 18 || field > 110)
    return "L'âge doit être compris entre 18 et 110.\n"
  return ""
}

function validateEmail(field)
{
  if (field == "") return "Entrez une adresse de courrier électronique.\n"
    else if (!((field.indexOf(".") > 0) &&
               (field.indexOf("@") > 0)) ||
                /[^a-zA-Z0-9.@_-]/.test(field))
      return "L'adresse de courrier électronique n'est pas valable.\n"
  return ""
}
```

Vous constatez, ici aussi, que les messages destinés à l'affichage à l'utilisateur sont en français, tandis que le code, lui, est encore en anglais. Nous allons examiner ces fonctions l'une après l'autre, en commençant par validateForename, pour que vous compreniez bien comment fonctionne la validation.

Valider le prénom : validateForename

Cette fonction est assez courte et accepte le paramètre `field`, qui contient la valeur du champ du formulaire correspondant au prénom, champ transmis par `validate` à l'appel de `validateForename`.

Si `field` contient une chaine vide, alors la fonction renvoie un message d'erreur adéquat; sinon elle renvoie une chaine vide pour indiquer qu'il n'y a pas d'erreur.

Si l'utilisateur entre des espaces dans ce champ, `validateForename` les accepte, bien que le contenu du champ devienne inexploitable pour le but visé. Pour régler ce problème, vous pouvez soit ajouter une instruction supplémentaire qui élimine les espaces du champ avant de vérifier s'il est vide, soit emprunter une expression rationnelle pour vérifier qu'il y a autre chose dans le champ que des espaces, soit, comme dans le cas de la solution choisie ici, laisser l'utilisateur commettre l'erreur et permettre au programme de l'intercepter du côté du serveur, après la soumission du formulaire.

Valider le nom de famille : validateSurname

La fonction `validateSurname` est presque identique à `validateForename` car une erreur n'est renvoyée que lorsque le nom de famille entré correspond à une chaine vide. Aucune limite de caractère n'est imposée au niveau des deux champs destinés aux noms pour autoriser l'entrée de caractères accentués, fréquents en français.

Valider le nom d'utilisateur : validateUsername

La fonction `validateUsername` présente un peu plus d'intérêt parce qu'elle assure un travail plus complexe. Elle ne doit autoriser que les caractères `a` à `z`, `A` à `Z`, `0` à `9`, _ et -, puis vérifier que les noms d'utilisateur comptent au moins cinq caractères.

Les instructions `if...else` commencent par renvoyer une erreur si la variable `field` n'a pas été remplie. Si elle n'est pas une chaine vide, mais une chaine de longueur inférieure à cinq caractères, un autre message est renvoyé.

Si aucune erreur n'est détectée à ce stade, un appel à la fonction JavaScript `test`, sur base d'une expression rationnelle (qui repère tout caractère qui *ne* figure *pas* parmi les caractères autorisés), la compare au contenu de field. (Voir « Expressions rationnelles », à la page 378.) Si un caractère, ne serait-ce qu'un seul, ne fait pas partie des caractères acceptables, la fonction `test` renvoie `true`, donc `validateUsername` retourne une chaine d'erreur.

Valider le mot de passe : validatePassword

La fonction `validatePassword` utilise des techniques comparables à celles de la précédente. Elle vérifie si `field` est vide, auquel cas elle renvoie une erreur. Ensuite, elle renvoie un message d'erreur si le mot de passe contient moins de six caractères.

Une des exigences que nous imposons aux mots de passe est d'avoir au moins un caractère en bas de casse, une lettre en capitale et un chiffre. Par conséquent, nous appelons `test` trois fois, pour tester chacune de ces conditions. Si l'un de ces appels renvoie

`false` parce qu'une de ces conditions n'est pas respectée, la fonction renvoie une erreur. Sinon, la fonction de validation du mot de passe renvoie une chaine vide pour indiquer que le mot de passe est acceptable.

Valider l'âge : validateAge

La fonction `validateAge` renvoie un message d'erreur si `field` est vide ou s'il ne correspond pas à un nombre (ce que détermine la fonction `isNaN`), ou alors si l'âge entré est inférieur à 18 ou supérieur à 110. En pratique, vos applications peuvent exiger d'autres conditions d'âge ou aucune. Cette fois également, lorsque les vérifications se passent correctement, la fonction de validation renvoie une chaine vide.

Valider l'adresse de courrier électronique : validateEmail

Dans le dernier exemple, `validateEmail` doit vérifier la validité d'une adresse de courrier électronique. La première vérification détermine si quelque chose a été entré dans ce champ et, si ce n'est le cas, renvoie un message d'erreur. Ensuite, la fonction appelle deux fois la fonction JavaScript `indexOf`, la première pour vérifier s'il y a un point (.) quelque part à partir du deuxième caractère jusqu'à la fin du champ, tandis que la seconde vérifie la présence du symbole `@` à partir du deuxième caractère.

Ces deux vérifications satisfaites, l'appel à la fonction `test` vérifie l'absence de caractères non autorisés dans le champ. Si un de ces tests échoue, `validateEmail` renvoie un message d'erreur; sinon, elle renvoie une chaine vide. Les caractères autorisés dans une adresse de courrier électronique sont les lettres en bas de casse et en capitales, les chiffres, le soulignement, le tiret, le point et l'arobase (@), comme l'indique l'expression rationnelle passée à la méthode `test`. La dernière ligne clôture le script.

La figure 16-2 montre le résultat obtenu lorsque l'utilisateur clique sur Inscription sans avoir rempli les champs.

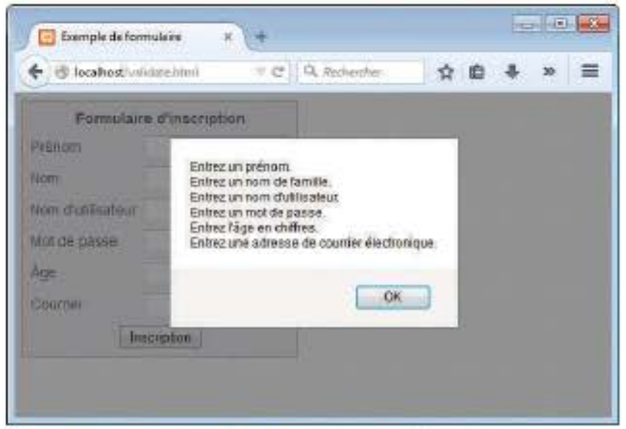


Figure 16-2. Validation JavaScript de formulaire en action

Utiliser un fichier JavaScript séparé

Bien entendu, ces constructions génériques pourraient s'appliquer à de nombreux types de validations qui peuvent se présenter par la suite, donc ces six fonctions constituent des candidates idéales à un déplacement vers un fichier JavaScript distinct. Vous pourriez nommer ce fichier *validate_functions.js* et l'inclure dans le fichier de l'exemple 16-1, juste après la section de script initiale, à l'aide de l'instruction suivante :

```
<script src="validate_functions.js"></script>
```

Expressions rationnelles

Examinons un peu plus attentivement les filtrages par motif que nous avons mis en place. Nous avons pu réaliser ce filtrage à l'aide d'*expressions rationnelles*, également appelées *expressions régulières* (terme traduit littéralement de l'anglais *regular* expressions, ou *regex* en abrégé), prises en charge tant par PHP que par JavaScript. Elles permettent de construire les plus puissants algorithmes de filtrage à partir d'une seule expression.

Filtrage à l'aide de métacaractères

Toute expression rationnelle est entourée de barres obliques (/). Entre ces barres obliques, certains caractères possèdent une signification particulière : ce sont les métacaractères. Ainsi, l'astérisque (*) a une signification comparable à celle que vous connaissez si vous avez déjà utilisé une Invite de commande de Windows ou un shell de Linux, mais à quelques différences près. L'astérisque signifie « le texte que vous essayez de filtrer peut avoir n'importe quel nombre de caractères qui précèdent, ou aucun ».

Par exemple, imaginons que vous cherchez le nom *Le Guin* dans une chaine et que vous savez que certains l'écrivent avec, d'autres sans espace. Comme le texte apparait quelquefois sous une forme étrange (si quelqu'un a ajouté de nombreux espaces pour justifier les lignes à droite, par exemple), vous devez parfois effectuer la recherche dans une ligne comme la suivante :

```
La  difficulté  de  classer les  travaux  de  Le  Guin
```

Vous devez donc retrouver *LeGuin* mais aussi *Le* et *Guin* séparés par un nombre inconnu d'espaces. La solution consiste à faire suivre une espace d'un astérisque :

```
/Le *Guin/
```

Il y a bien plus dans la phrase que le nom *Le Guin* mais c'est correct. Tant que l'expression rationnelle correspond à une partie de la ligne, la fonction `test` renvoie la valeur `true`. Et si c'est important que la ligne ne contienne rien de plus que *Le Guin* ? Nous verrons comment vérifier cela dans un instant.

Supposons que vous savez qu'il y a toujours au moins une espace. Dans ce cas, utilisez le signe plus (+) car il impose la présence d'au moins un des caractères précédents :

```
/Le +Guin/
```


Correspondance floue de caractère

Le point s'avère particulièrement intéressant parce qu'il représente n'importe quoi sauf un retour à la ligne. Si vous recherchez par exemple des balises HTML, qui commencent par < et se terminent par >, voici un moyen simple de les retrouver :

```
/<.*>/
```

Le point représente n'importe quel caractère et l'astérisque l'étend pour correspondre à zéro caractère ou plus, de sorte que cela signifie « retrouver tout ce qui se trouve entre < et >, même s'il n'y a rien ». Vous retrouvez ainsi `<>`, `<em>`, `<br>` et ainsi de suite. Mais si vous voulez rejeter le cas vide, utilisez + au lieu de *, comme suit :

```
/<.+>/
```

Le signe plus étend le point pour correspondre à un ou plusieurs caractères, pour dire « retrouver tout ce qui se trouve entre < et >, pourvu qu'il y ait au moins un caractère entre eux ». Ceci permet de retrouver `<em>`, `</em>`, `<h1>`, `</h1>` et les balises avec attributs, comme la suivante :

```
<a href="www.mozilla.org">
```

Malheureusement, le signe plus continue d'accepter tout jusqu'au dernier > de la ligne, donc vous récupèreriez toute la ligne suivante :

```
<h1><b>Introduction</b></h1>
```

Ceci représente bien plus qu'une seule balise. Plus loin, cette section vous propose une meilleure solution.

Si vous utilisez le point tout seul entre les parenthèses en chevron, sans le faire suivre d'un * ou d'un +, alors il correspond à un seul caractère ; vous obtenez `<b>` et `<i>` mais pas `<em>` ni `<textarea>`.

Pour retrouver le caractère point (.) en soi, vous devez le placer dans une séquence d'échappement avec une barre oblique inverse (\) en préfixe de ce point, sinon il est considéré comme un métacaractère et renvoie n'importe quoi. En guise d'exemple, si vous voulez retrouver le nombre en virgule flottante 5.0, l'expression rationnelle adéquate devient :

```
/5\.0/
```

La barre oblique inverse peut placer en séquence d'échappement n'importe quel métacaractère, y compris une autre barre oblique inverse (lorsque vous voulez retrouver une telle barre dans le texte). Cependant, pour compliquer encore un peu plus la situation, vous verrez un peu plus loin que les barres obliques inverses donnent une signification particulière au caractère suivant.

Nous venons de repérer un nombre en virgule flottante. Mais vous recherchez peut-être `5.`, ainsi que `5.0`, parce qu'ils signifient la même chose qu'un nombre à virgule flottante. Si c'est le cas, vous recherchez aussi `5.00`, `5.000` et ainsi de suite, où le nombre de zéro importe peu. Pour ce faire, ajoutez un astérisque, comme suit :

```
/5\.0*/
```

Regroupement par des parenthèses

Admettons que vous vouliez retrouver des puissances d'incréments d'unités, comme kilo, méga, giga et téra. En d'autres termes, vous voulez toutes les combinaisons suivantes :

```
1 000
1 000 000
1 000 000 000
1 000 000 000 000
...
```

Le signe plus fonctionne ici aussi mais vous devez grouper la chaine composée d'une espace, suivie de `000`, pour que le signe plus puisse repérer la chaine complète. L'expression rationnelle est la suivante :

```
/1( 000)+ /
```

Les parenthèses signifient « traiter ceci comme un groupe lors de l'application de quelque chose comme le signe plus ». 1 00 000 et 1 000 00 ne passeront pas parce que le 1 doit être suivi d'un ou plusieurs groupes complets d'une virgule suivie de trois zéros.

L'espace après le caractère `+` indique que la correspondance s'arrête dès qu'une espace est rencontrée. Sans elle, le nombre 1 000 00 deviendrait un faux positif parce que seul le premier `1000` serait pris en compte et le (espace)`00` restant serait ignoré. Le fait d'exiger une espace ensuite garantit que la correspondance soit recherchée jusqu'à la fin d'un nombre.

Classes de caractères

Parfois, il est nécessaire de rechercher quelque chose de flou mais pas au point d'utiliser le point, trop général. C'est dans l'aspect flou que réside la force des expressions rationnelles : elles permettent de demeurer aussi précis ou vague que vous le souhaitez.

Une des fonctionnalités clés du filtrage flou se situe dans la paire de crochets, `[]`. Ils indiquent un seul caractère, comme un point, mais entre ces crochets, vous pouvez placer une liste de choses qui peuvent convenir. Si un de ces caractères apparait, le texte correspond. Ainsi, si vous cherchez une correspondance avec les mots `grain` et `groin`, qui diffèrent d'une seule lettre, vous pouvez écrire :

```
/gr[ao]in/
```

Après le `gr` du texte que vous comparez, il peut y avoir un `a` ou un `o`. Mais il ne peut y avoir qu'une seule parmi ces lettres : ce que vous placez entre crochets correspond exactement à un seul caractère. Le groupe de caractères entre les crochets s'appelle une *classe de caractère*.

Indication de plage

Entre les crochets, un tiret indique une plage. Une utilisation classique de ceci réside dans la correspondance à un seul chiffre, qu'une plage permet de préciser comme suit :

```
/[0-9]/
```


Les chiffres sont d'un tel usage commun dans les expressions rationnelles qu'un simple caractère permet de les représenter : `\d`. Utilisez-le à la place de l'expression rationnelle entre crochets pour correspondre à un chiffre :

```
/\d/
```

Négation

Une autre importante possibilité des crochets réside dans la *négation* d'une classe de caractères. Inversez la totalité de la classe de caractères précisée en plaçant un caret (`^`) après le crochet ouvrant. Celui-ci signifie « repérer tout caractère, sauf ce qui suit ». Imaginons que vous voulez retrouver toutes les occurrences de *Yahoo* qui ne sont pas suivies du point d'exclamation de la dénomination officielle de la société. Écrivez l'expression rationnelle suivante :

```
/Yahoo[^!]/
```

La classe de caractères contient un seul caractère, le point d'exclamation, mais il est inversé, à cause de la présence du caret qui le précède. Cependant, ceci n'est pas une solution idéale à ce problème car, par exemple, elle échoue lorsque *Yahoo* se situe à la fin d'une ligne, parce qu'il n'est suivi de *rien*, tandis que les crochets doivent faire correspondre un caractère. Une meilleure solution consiste à tirer parti de l'*anticipation* négative (la correspondance de quelque chose suivi de rien d'autre), mais ceci dépasse la portée de ce livre.

Exemples plus complexes

Maintenant que vous comprenez le fonctionnement des classes de caractères et de la négation, examinons une meilleure solution au problème de la correspondance à une balise HTML. Cette solution évite de dépasser la fin d'une seule balise, mais reconnait toujours les balises telles que `<em>` et `</em>`, ainsi que celles avec des attributs, comme celle-ci :

```
<a href="www.mozilla.org">
```

Voici une solution :

```
/<[^>]+>/
```

Cette expression rationnelle aurait pu apparaitre si j'avais renversé ma tasse de café sur mon clavier, mais elle est parfaitement valable et très pratique. La figure 16-3 détaille ses différents éléments.

Figure 16-3. Dissection d'une expression rationnelle type

Les éléments sont les suivants :

`/`
: Barre oblique de début d'expression rationnelle.

`<`
: Crochet d'ouverture d'une balise HTML. La correspondance doit être exacte; ce n'est pas un métacaractère.

`[^>]`
: Classe de caractère. Le `^>` compris indique tout caractère sauf `>`.

`+`
: Autorise n'importe quel nombre de caractères qui correspondent à la contrainte `[^>]` précédente, tant qu'il y en a au moins un.

`>`
: Crochet de fermeture d'une balise HTML. La correspondance doit être exacte.

`/`
: Barre oblique de fin d'expression rationnelle.

Une autre solution au problème de reconnaissance des balises HTML consiste à utiliser une opération « non gloutonne ». Par défaut, le filtrage par motif est glouton et renvoie la correspondance la plus longue possible. Les filtrages non gloutons recherchent la correspondance la plus courte possible, mais ces concepts échappent à la portée de ce livre.

Explorons ensuite une autre expression tirée de l'exemple 16-1, utilisée par la fonction `validateUsername`, soit :

```
/[^a-zA-Z0-9_-]/
```

La figure 16-4 en détaille les différents éléments.

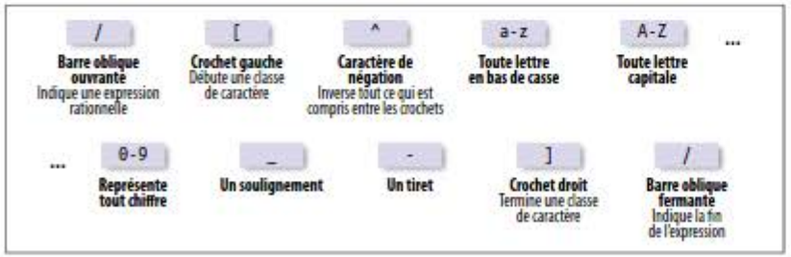


Figure 16-4. Dissection de l'expression rationnelle de validateUsername

Les éléments sont les suivants :

`/`
: Barre oblique de début d'expression rationnelle.

`[`
: Crochet ouvrant de début de classe de caractère.

`^`
: Caractère de négation ; inverse tout le reste du contenu des crochets.

`a-z`
: Représente toute lettre en bas de casse.

`A-Z`
: Représente toute lettre capitale.

`0-9`
: Représente un chiffre.

`_`
: Soulignement.

`-`
: Tiret.

`]`
: Crochet de fermeture d'une classe de caractère.

`/`
: Barre oblique de fin d'expression rationnelle.

Deux autres caractères revêtent de l'importance parce qu'ils ancrent une expression rationnelle et exigent qu'elle se situe à un emplacement spécifique. Si un caret (^) apparait au début de l'expression rationnelle, l'expression doit se situer au début d'une ligne de texte; sinon elle est rejetée. De même, si un signe dollar ($) apparait à la fin d'une expression, celle-ci doit se situer à la fin d'une ligne de texte.

Le fait que le caret puisse signifier « inversion de la classe de caractère » au sein de crochets et « correspondance au début d'une ligne », s'il est présent au début de l'expression rationnelle entraine une certaine confusion. Comme le même caractère a deux significations différentes, soyez prudent lorsque vous l'utilisez.

Pour clôturer notre exploration des bases des expressions rationnelles, répondons à la question soulevée précédemment : supposons que nous voulons une ligne contenant « Le Guin » et rien d'autre dans une ligne. Nous pouvons ajouter des ancres aux extrémités de l'expression rationnelle que nous avons définie précédemment :

```
/^Le *Guin$/
```

Résumé des métacaractères

Le tableau 16-1 énumère les métacaractères disponibles dans les expressions rationnelles.

Tableau 16-1. Métacaractères d'expressions rationnelles (regex)

Métacaractères	Description
/	Débute et termine l'expression rationnelle
.	Identifie un seul caractère, n'importe lequel sauf la nouvelle ligne
*élément**	Identifie *élément* zéro fois ou plus
élément+	Identifie *élément* une fois ou plus
élément?	Identifie *élément* zéro ou une fois
[*caractères*]	Identifie un seul caractère parmi ceux cités entre les crochets
[^*caractères*]	Identifie un seul caractère non compris entre les crochets
(*regex*)	Traite la *regex* comme un groupe pour le comptage ou suivi par *, + ou ?
gauche|*droite*	Identifie soit *gauche*, soit *droite*
[*l*-*r*]	Identifie un caractère de la plage comprise entre *l* et *r*
^	Exige que la correspondance soit en début de chaine
$	Exige que la correspondance soit en fin de chaine
\b	Identifie une limite de mot
\B	Identifie s'il n'y a pas de limite de mot
\d	Identifie un seul chiffre
\D	Identifie un seul caractère à l'exception de tout chiffre
\n	Identifie un caractère de nouvelle ligne
\s	Identifie un caractère d'espacement
\S	Identifie un caractère autre qu'un caractère d'espacement
\t	Identifie un caractère de tabulation
\w	Identifie un caractère de mot (a-z, A-Z, 0-9 et _)
\W	Identifie un caractère de non-mot (n'importe lequel, sauf a-z, A-Z, 0-9 et _)
x	*x* (utile si *x* est le métacaractère, et si vous voulez exactement *x*)
{*n*}	Identifie exactement *n* fois
{*n*,}	Identifie *n* fois ou plus
{*min*,*max*}	Identifie au moins *min* fois et au plus *max* fois

Sur base de ce tableau, lorsque vous examinez à nouveau l'expression /[^a-zA-Z0-9_]/, vous constatez qu'il est possible de l'abréger en /[^w]/ parce que le simple \w (avec un w en bas de casse) équivaut aux caractères a-z, A-Z, 0-9 et _.

En fait, nous pouvons prolonger le raisonnement puisque le métacaractère \W (avec un W en capitale) représente tout caractère, *sauf* ceux des plages a-z, A-Z, 0-9 et _. Nous pouvons éluder le caret et utiliser simplement l'expression /[\W]/.

Remarquez toutefois que si /[\w]/ identifie un caractère alphanumérique ou un soulignement, les caractères accentués n'en font pas partie. Pour tenir compte des caractères accentués français, il vous reste à les citer explicitement, comme suit : /[a-zA-Z0-9_ÀÂÇÉÊÈËÎÔÛÙÜàâçéêèëîôûùü]/.

Pour mieux comprendre comment tout cela fonctionne, le tableau 16-2 propose quelques expressions et les motifs de filtrage auxquels elles s'identifient.

Tableau 16-2. Quelques exemples d'expressions rationnelles (regex)

Exemple	Correspondance
r	Le premier r dans Voix ambigüe d'un cœur
l[io][io]n	Toute occurrence de lion et loin (mais aussi liin et loon)
l[io]{2}n	Toute occurrence de lion et loin (mais aussi liin et loon)
l(io|oi)n	Toute occurrence de lion et loin (mais pas liin et loon)
chat	Le mot chat dans j'aime les chiens et les chats
chat|chien	Le mot chat ou chien, dans j'aime les chiens et les chats
\.	Le caractère point (la \ est nécessaire car le point est un métacaractère)
5\.0+	5., 5.0, 5.00, 5.000 et ainsi de suite
[a-f]	Un seul caractère parmi la liste a, b, c, d, e et f.
chats$	Le seul chats final dans mes chats sont de gentils chats
^mes	Le seul premier mes dans mes chats sont mes animaux préférés
\d{2,3}	Tout nombre de deux à trois chiffres (soit entre 00 et 999)
7(000)+	7 000; 7 000 000; 7 000 000 000; 7 000 000 000 000; etc.
[\w]+	N'importe quel mot (sans caractère accentué) d'un caractère ou plus
[\w]{5}	N'importe quel mot (sans caractère accentué) de cinq caractères exactement

Modificateurs généraux

Quelques modificateurs supplémentaires permettent d'agir sur les expressions rationnelles :

- /g autorise une identification globale. Lors de l'utilisation d'une fonction de remplacement, précisez ce modificateur pour remplacer toutes les correspondances et non seulement la première.
- /i impose à l'expression rationnelle de ne pas tenir compte de la casse. Donc, à la place de [a-zA-Z], vous pouvez écrire [a-z]/i, qui revient au même que [A-Z]/i.
- /m active le mode multiligne, où les correspondances au caret (^) et au dollar ($) sont identifiées avant et après tous les retours à la ligne éventuels dans la chaine cible. Normalement, dans une chaine sur plusieurs lignes, le ^ n'identifie que le début de la chaine, tandis que le $ s'arrête à la fin de toute la chaine.

Ainsi, l'expression /chats/g identifie les deux occurrences du mot *chats* dans la phrase *j'aime mes chats et mes chats m'aiment*. De même, /les/gi identifie les deux occurrences du mot *les* (*Les* et *les*) dans la phrase suivante : *Les chiens aiment jouer avec les autres chiens*. Vous pouvez en effet cumuler les modificateurs généraux.

Utiliser les expressions rationnelles en JavaScript

En JavaScript, l'utilisation des expressions rationnelles se résume essentiellement à deux méthodes : test (que vous avez déjà vue) et replace. Alors que test indique simplement si son argument correspond à l'expression rationnelle, replace prend un deuxième paramètre, la chaine de remplacement dans la chaine cible (premier argument) quand une correspondance est identifiée par l'expression relationnelle. Comme la plupart des fonctions, replace génère la nouvelle chaine en retour mais ne modifie pas l'argument reçu.

Pour comparer les deux méthodes, l'instruction suivante renvoie simplement true pour indiquer que le mot *les* apparait au moins une fois (en gras) dans la chaine :

```
document.write(/les/i.test("Les chats sont amusants. J'aime bien les chats."))
```

En revanche, l'instruction suivante remplace les deux occurrences du mot *chat* par le mot *chien* et affiche le résultat de la substitution. La recherche doit être globale (/g) pour trouver toutes les occurrences, et l'insensibilité à la casse (/i) pour trouver le *Chat* avec une capitale :

```
document.write("Les chats sont affectueux. Chat qui s'en dédit!".replace(/chat/gi,"chien"))
```

Lorsque vous exécutez cette instruction, vous constatez une des limites de replace : elle remplace le texte trouvé par la chaine exacte que vous lui donnez, donc le deuxième mot *Chat* est remplacé par *chien* au lieu de *Chien*.

Utiliser les expressions rationnelles en PHP

Les fonctions d'expression rationnelle les plus fréquemment utilisées en PHP sont preg_match, preg_match_all et preg_replace.

Pour vérifier si le mot *les* apparait où que ce soit dans une chaine, quelle que soit la combinaison en capitales et en bas de casse, utilisez preg_match, comme suit :

```
$n = preg_match("/les/i", "Les chats sont fous. J'aime les chats.");
```

Comme PHP utilise 1 pour TRUE et 0 pour FALSE, l'instruction précédente affecte 1 à $n. L'expression rationnelle figure en premier argument et le texte à comparer figure en deuxième. Cependant, preg_match constitue une solution bien plus puissante et complexe parce qu'elle prend un troisième argument qui montre le ou les textes identifiés :

```
$n = preg_match("/les/i", "Les chats sont curieux. J'aime bien les chats.", $corresp);
echo "$n Correspondance : $corresp[0]";
```

Le troisième argument est un tableau (qui porte le nom $corresp dans l'exemple). La fonction dépose le texte identifié dans son premier élément donc, si le filtrage réussit, vous obtenez le texte correspondant dans $corresp[0]. Dans cet exemple, la sortie indique que le texte identifié possède une première capitale, comme suit :

```
1 Correspondance : Les
```

Si vous voulez connaitre toutes les correspondances, utilisez plutôt la fonction preg_match_all, comme suit :

```
$n = preg_match_all("/les/i", "Les chats sont bizarres. J'aime bien les chats.", $corresp);
echo "$n Correspondances : ";
for ($j=0 ; $j < $n ; ++$j) echo $corresp[0][$j]." ";
```

Comme précédemment, $corresp est passé à la fonction et l'élément $corresp[0] reçoit les identifications obtenues, mais cette fois dans un sous-tableau. Pour afficher le sous-tableau, l'exemple emprunte une boucle pour itérer dans celui-ci.

Pour remplacer une portion d'une chaine, utilisez preg_replace comme suit. Cet exemple remplace toutes les occurrences de *chats* par *chiens*, quelle qu'en soit la casse :

```
echo preg_replace("/chats/i", "chiens", "Les chats sont poilus. J'aime bien les chats.");
```

Le sujet des expressions rationnelles est un des plus vastes qui soient, au point qu'elles font l'objet de livres entiers. Si vous souhaitez de plus amples informations à leur propos, consultez l'article de *Wikipédia*.

Réafficher un formulaire après la validation PHP

Revenons-en à la validation de formulaire. À ce stade, nous avons créé un document HTML, *validate.html*, qui soumet des données à un programme en PHP nommé *adduser.php* (ajouter utilisateur), mais seulement si JavaScript valide les champs, ou si JavaScript est désactivé ou indisponible.

Nous devons donc encore créer le programme *adduser.php* afin qu'il puisse recevoir les données du formulaire et effectuer ses propres validations, pour réafficher le formulaire au visiteur si la validation échoue. L'exemple 16-3 contient le code à entrer et enregistrer (ou télécharger du site d'accompagnement).

Exemple 16-3. Le programme adduser.php

```
<?php // adduser.php

  // Le code PHP (les fonctions PHP viennent en fin de script)

  $forename = $surname = $username = $password = $age = $email = "";

  if (isset($_POST['forename']))
    $forename = fix_string($_POST['forename']);
  if (isset($_POST['surname']))
    $surname  = fix_string($_POST['surname']);
  if (isset($_POST['username']))
    $username = fix_string($_POST['username']);
  if (isset($_POST['password']))
    $password = fix_string($_POST['password']);
  if (isset($_POST['age']))
    $age      = fix_string($_POST['age']);
  if (isset($_POST['email']))
    $email    = fix_string($_POST['email']);

  $fail  = validate_forename($forename);
  $fail .= validate_surname($surname);
  $fail .= validate_username($username);
  $fail .= validate_password($password);
  $fail .= validate_age($age);
  $fail .= validate_email($email);

  echo "<!DOCTYPE html>\n<html><head><title>Exemple de formulaire</title>";

  if ($fail == "")
  {
    echo "</head><body>Données du formulaire validées avec succès :
      $forename, $surname, $username, $password, $age, $email.</body></html>";

    // Ici vient s'insérer le code nécessaire pour enregistrer les champs
    // du formulaire dans une base de données, de préférence à l'aide
    // de la fonction de chiffrement hash dans le cas du mot de passe.

    exit;
  }

  echo <<<_END

    <!-- La section HTML/JavaScript -->
```



```
<style>
  .signup {
    border: 1px solid #999999;
  font:   normal 14px helvetica; color:#444444;
  }
</style>

<script>
  function validate(form)
  {
    fail  = validateForename(form.forename.value)
    fail += validateSurname(form.surname.value)
    fail += validateUsername(form.username.value)
    fail += validatePassword(form.password.value)
    fail += validateAge(form.age.value)
    fail += validateEmail(form.email.value)

    if (fail == "")     return true
    else { alert(fail); return false }
  }

  function validateForename(field)
  {
    return (field == "") ? "Entrez un prénom.\n" : ""
  }
  function validateSurname(field)
  {
    return (field == "") ? "Entrez un nom de famille.\n" : ""
  }

  function validateUsername(field)
  {
    if (field == "") return "Entrez un nom d'utilisateur.\n"
    else if (field.length < 5)
      return "Le nom d'utilisateur doit contenir au moins 5 caractères.\n"
    else if (/[^a-zA-Z0-9_-]/.test(field))
      return "Seuls caractères permis dans le nom d'utilisateur : " +
             "a-z, A-Z, 0-9, - et _.\n"
    return ""
  }

  function validatePassword(field)
  {
    if (field == "") return "Entrez un mot de passe.\n"
    else if (field.length < 6)
      return "Le mot de passe doit contenir au moins 6 caractères.\n"
    else if (!/[a-z]/.test(field) || ! /[A-Z]/.test(field) ||
             !/[0-9]/.test(field))
      return "Le mot de passe doit contenir au moins un a-z, A-Z et 0-9.\n"
    return ""
  }
```

```
    function validateAge(field)
    {
      if (field == "" || isNaN(field)) return "Entrez l'âge en chiffres.\n"
      else if (field < 18 || field > 110)
        return "L'âge doit être compris entre 18 et 110.\n"
      return ""
    }

    function validateEmail(field)
    {
      if (field == "") return "Entrez une adresse de courrier électronique.\n"
        else if (!((field.indexOf(".") > 0) &&
                   (field.indexOf("@") > 0)) ||
                   /[^a-zA-Z0-9.@_-]/.test(field))
          return "L'adresse de courrier électronique n'est pas valable.\n"
      return ""
    }
  </script>
</head>
<body>

  <table class="signup" border="0" cellpadding="2" cellspacing="5"
    bgcolor="#eeeeee">
    <th colspan="2" align="center">Formulaire d'inscription</th>

      <tr><td colspan="2">Désolé, mais votre formulaire<br>
        comporte les erreurs suivantes :
        <p><font color=red size=1><i>$fail</i></font></p>
      </td></tr>

    <form method="post" action="adduser.php" onSubmit="return validate(this)">
      <tr><td>Prénom</td>
        <td><input type="text" maxlength="32" name="forename" value="$forename">
      </td></tr><tr><td>Non</td>
        <td><input type="text" maxlength="32" name="surname"  value="$surname">
      </td></tr><tr><td>Non d'utilisateur</td>
        <td><input type="text" maxlength="16" name="username" value="$username">
      </td></tr><tr><td>Mot de passe</td>
        <td><input type="text" maxlength="12" name="password" value="$password">
      </td></tr><tr><td>Âge</td>
        <td><input type="text" maxlength="3"  name="age"      value="$age">
      </td></tr><tr><td>Courriel</td>
        <td><input type="text" maxlength="64" name="email"    value="$email">
      </td></tr><tr><td colspan="2" align="center"><input type="submit"
        value="Inscription"></td></tr>
    </form>
  </table>
</body>
</html>
```



```
_END;

// Les fonctions en PHP

function validate_forename($field)
{
      return ($field == "") ? "Aucun prénom entré.<br>": "";
}

function validate_surname($field)
{
      return($field == "") ? "Aucun nom de famille entré.<br>" : "";
}

function validate_username($field)
{
  if ($field == "") return "Aucun nom d'utilisateur entré.<br>";
  else if (strlen($field) < 5)
    return "Le nom d'utilisateur doit contenir au moins 5 caractères.<br>";
  else if (preg_match("/[^a-zA-Z0-9_-]/", $field))
    return "Seuls caractères permis dans le nom d'utilisateur : " .
           "a-z, A-Z, 0-9, - et _.<br>";
  return "";
}

function validate_password($field)
{
  if ($field == "") return "Aucun mot de passe entré.<br>";
  else if (strlen($field) < 6)
    return "Le mot de passe doit contenir au moins 6 caractères.<br>";
  else if (!preg_match("/[a-z]/", $field) ||
           !preg_match("/[A-Z]/", $field) ||
           !preg_match("/[0-9]/", $field))
    return "Le mot de passe doit contenir au moins un a-z, A-Z et 0-9.<br>";
  return "";
}

function validate_age($field)
{
  if ($field == "") return "Aucun âge entré.<br>";
  else if ($field < 18 || $field > 110)
    return "L'âge doit être compris entre 18 et 110.<br>";
  return "";
}

function validate_email($field)
{
  if ($field == "") return "Entrez une adresse de courrier électronique.<br>";
    else if (!((strpos($field, ".") > 0) &&
               (strpos($field, "@") > 0)) ||
                preg_match("/[^a-zA-Z0-9.@_-]/", $field))
```

```
        return "L'adresse de courrier électronique n'est pas valable.<br>";
    return "";
  }

  // Aseptisation, comme au chapitre 10
  function fix_string($string)
  {
    if (get_magic_quotes_gpc()) $string = stripslashes($string);
    return htmlentities ($string);
  }
?>
```

Dans cet exemple, toutes les entrées sont aseptisées avant utilisation, même les mots de passe qui, puisqu'ils peuvent aussi contenir des caractères employés pour mettre en forme du HTML, sont convertis en entités HTML. Ainsi, `&` devient `&`, `<` devient `<`, et ainsi de suite. Si vous utilisez la fonction `hash` pour chiffrer les mots de passe avant de les stocker, cela ne pose pas de problème, tant que vous vérifiez par la suite les mots de passe entrés par l'utilisateur pour se connecter et que vous les aseptisez de la même manière pour être certain de comparer les mêmes entrées.

La figure 16-5 illustre les résultats de la soumission du formulaire avec JavaScript désactivé (et deux champs non remplis correctement).

Figure 16-5. Le formulaire représenté après l'échec de validation en PHP

Dans le script, les sections relatives à PHP et les modifications apportées à la section HTML sont indiquées en gras pour que vous puissiez voir clairement les différences par rapport aux exemples 16-1 et 16-2.

Si vous parcourez ce code en détail, vous constatez que le code PHP est un clone du code JavaScript, avec les mêmes expressions rationnelles reprises pour valider chacun des champs à l'aide de fonctions fort semblables.

Mais deux choses méritent votre attention. D'abord, la fonction `fix_string` (définie tout à la fin du script) permet d'aseptiser chaque champ et d'empêcher toute tentative d'injection de code.

Vous voyez également que tout le HTML de l'exemple 16-1 est reproduit dans le code PHP à l'aide de la structure `<<<_END..._END;` pour afficher le formulaire avec les valeurs entrées précédemment par le visiteur. Pour ce faire, ajoutez simplement le paramètre supplémentaire `value` à chaque balise `<input>` (par exemple `value="$forename"` pour le prénom). Cette mesure de courtoisie s'avère bienvenue pour que l'utilisateur puisse modifier les valeurs qu'il a déjà entrées au lieu de devoir entrer à nouveau tous les champs.

Dans la réalité, vous ne débuterez probablement pas avec un formulaire HTML tel que celui de l'exemple 16-1, mais vous démarrerez plutôt avec un programme PHP du genre de celui de la figure 16-3 qui incorpore tout le HTML. Bien entendu, vous devrez aussi ajouter les éléments nécessaires pour traiter le cas du premier appel du programme, afin d'éviter qu'il n'affiche des erreurs lorsque tous les champs sont vides. Vous enregistrerez aussi les six fonctions JavaScript dans un fichier .js distinct, pour ensuite les inclure.

Maintenant que vous avez vu comment imbriquer tout le code PHP, HTML et JavaScript, le chapitre suivant présente Ajax (*Asynchronous JavaScript and XML*), qui utilise en coulisses des appels JavaScript au serveur pour mettre à jour de manière transparente des portions d'une page web, sans devoir soumettre à nouveau la page complète au serveur web.

Questions

1. Quelle méthode de JavaScript permet d'envoyer un formulaire en validation avant de le soumettre ?
2. Quelle méthode de JavaScript permet de comparer une chaine à une expression rationnelle ?
3. Écrivez une expression rationnelle pour identifier tous les caractères qui ne font pas partie d'un mot, selon la syntaxe normale des expressions rationnelles.
4. Écrivez une expression rationnelle qui identifie un des mots *vivre* ou *vitre*.
5. Donnez une expression rationnelle qui identifie tout mot simple suivi d'un caractère de type non-mot.

6. À l'aide d'expressions rationnelles, écrivez une fonction JavaScript pour vérifier si le mot *voix* existe dans la chaine *Voix ambigüe d'un cœur.*
7. À l'aide d'expressions rationnelles, écrivez une fonction PHP pour remplacer toutes les occurrences du mot *la*, quelle qu'en soit la casse, dans *La vache saute sur la lune*, par le mot *ma*.
8. Quel attribut HTML permet de préremplir un champ d'un formulaire à l'aide d'une valeur ?

Retrouvez les réponses du chapitre 16 dans l'annexe A.

CHAPITRE 17

Utiliser Ajax

Le terme *Ajax* a été inventé en 2005. Il signifie *Asynchronous JavaScript and XML* qui, en termes simples, désigne un ensemble de méthodes intégrées dans JavaScript pour transférer des données entre le navigateur et un serveur en coulisses. Google Maps constitue un excellent exemple de cette technologie (figure 17-1), où de nouvelles sections d'une carte sont téléchargées depuis le serveur lorsqu'elles sont nécessaires, sans imposer de recharger complètement la page.

L'utilisation d'Ajax non seulement réduit de manière substantielle la quantité de données à envoyer et recevoir, mais apporte également aux pages web un dynamisme transparent, pour leur donner un comportement comparable aux applications autonomes. Les résultats donnent une interface utilisateur améliorée et une réactivité bien meilleure.

Qu'est-ce qu'Ajax ?

Les débuts d'Ajax, tel qu'il est mis en œuvre actuellement, datent de la diffusion d'Internet Explorer 5, en 1999, qui a introduit un nouvel objet ActiveX, nommé `XMLHttpRequest`. ActiveX est la technologie de Microsoft qui permet de signer des modules additionnels (*plugins*) qui installent des logiciels supplémentaires dans l'ordinateur. Les autres développeurs de navigateurs ont suivi le mouvement mais, au lieu de passer par ActiveX, ils ont tous implanté la fonctionnalité en tant que partie intégrante, native de l'interpréteur JavaScript.

Cependant, bien avant cela, une certaine forme d'Ajax antérieure avait déjà exploré l'idée qui utilisait des cadres (*frames*) masqués sur les pages pour interagir avec le serveur en toile de fond. Les bavardoirs, ou salles de chat, ont été les premiers à adopter cette technologie, à l'utiliser pour interroger et afficher de nouvelles publications de messages sans imposer de rechargements de page.

Voyons donc comment mettre en œuvre Ajax à l'aide de JavaScript.

```
        catch(e2)
        {
          try
          {
            request = new ActiveXObject("Microsoft.XMLHTTP")
          }
          catch(e3)
          {
            request = false
          }
        }
      }
      return request
    }
  </script>
 </body>
</html>
```

Examinons ce document pour comprendre ce qu'il fait. Les six premières lignes définissent un document HTML et affichent un titre. La ligne suivante crée une section DIV avec l'identifiant `id='info'` et le texte prédéfini `Cette phrase sera remplacée`. Par la suite, le texte renvoyé par l'appel Ajax viendra s'insérer à cet endroit.

Les six lignes suivantes sont indispensables pour effectuer une requête Ajax en Post HTML. La première définit une paire *`paramètre-valeur`*, avec `params` et sa valeur, que nous enverrons au serveur. Suit la création de l'objet Ajax `request`, puis vient l'appel à la méthode `open` pour régler l'objet de manière à effectuer une requête Post à *urlpost.php* en mode asynchrone. Les trois dernières lignes de ce groupe définissent les entêtes indispensables pour que le serveur qui reçoit la requête sache qu'une requête Post arrive.

Propriété readyState

Nous entrons avec cette étape dans le vif du sujet à propos d'un appel Ajax, qui repose en totalité sur la propriété `readyState`. L'aspect *asynchrone* d'Ajax permet au navigateur de continuer d'accepter les entrées de l'utilisateur et de modifier l'écran, tandis que le programme affecte à la propriété `onreadystatechange` une fonction de notre choix, chaque fois que `readyState` change d'état. Dans le cas de l'exemple, nous écrivons une fonction anonyme, à la volée, au lieu d'une fonction nommée et séparée. Ce genre de fonction porte le nom de *fonction de rappel* (ou de *callback*) et elle est appelée automatiquement, chaque fois que `readyState` change d'état.

La syntaxe de définition de la fonction de rappel à l'aide d'une fonction anonyme, à la volée, est la suivante :

```
request.onreadystatechange = function()
{
  if (this.readyState == 4)
  {
    // Faire quelque chose
  }
}
```


Si vous préférez appeler une fonction séparée, nommée `ajaxCallback`, la syntaxe diffère légèrement :

```
request.onreadystatechange = ajaxCallback

function ajaxCallback()
{
  if (this.readyState == 4)
  {
    // Faire quelque chose
  }
}
```

Le tableau 17-1 indique que `readyState` peut prendre une parmi cinq valeurs possibles. Une seule nous importe, la valeur `4`, qui représente un appel Ajax achevé. Par conséquent, lors de chaque appel à la fonction, elle ne renvoie rien et ne fait rien, tant que `readyState` n'a pas la valeur `4`. En revanche, quand la fonction détecte cette valeur, elle examine ensuite l'état, `status`, de l'appel pour vérifier s'il a la valeur `200`, qui signifie que l'appel a réussi. Si sa valeur est différente de `200`, une boite de dialogue d'alerte affiche le message d'erreur contenu dans la propriété `statusText`.

Remarquez que toutes les références à ces propriétés d'objet utilisent `this.readyState`, `this.status` et ainsi de suite, au lieu du nom courant de l'objet, `request`, comme dans `request.readyState`. Cette manière d'écrire le code permet de copier-coller cette portion et elle fonctionnera quel que soit le nom de l'objet, parce que le mot clé `this` fait toujours référence à l'objet courant.

Après avoir vérifié que `readyState` vaut `4` et que `status` vaut `200`, nous vérifions ensuite si `responseText` contient une valeur. Si ce n'est pas le cas, nous affichons un message d'erreur dans une boite de dialogue. Sinon, le HTML interne de la section DIV reçoit la valeur de `responseText`, comme suit :

```
document.getElenentById('info').innerHTML = this.responseText
```

Dans cette ligne, la méthode `getElementByID` fait référence à l'élément `info`, puis la propriété `innerHTML` reçoit la valeur renvoyée par l'appel Ajax.

Tout ce travail de préparation terminé, la requête Ajax est envoyée au serveur par l'entremise de la commande suivante, qui transmet à celui-ci les paramètres déjà définis dans la variable `params` :

```
request.send(params)
```

Enfin, tout le code qui précède s'active lors de chaque changement d'état de `readyState`.

Le reste du document contient la fonction `ajaxRequest` issue de l'exemple 17-1, ainsi que les balises de fermeture du script et du HTML.

```
        catch(e2)
        {
          try
          {
            request = new ActiveXObject("Microsoft.XMLHTTP")
          }
          catch(e3)
          {
            request = false
          }
        }
      }
      return request
    }
  </script>
 </body>
</html>
```

Examinons ce document pour comprendre ce qu'il fait. Les six premières lignes définissent un document HTML et affichent un titre. La ligne suivante crée une section DIV avec l'identifiant `id='info'` et le texte prédéfini `Cette phrase sera remplacée`. Par la suite, le texte renvoyé par l'appel Ajax viendra s'insérer à cet endroit.

Les six lignes suivantes sont indispensables pour effectuer une requête Ajax en Post HTML. La première définit une paire *`paramètre-valeur`*, avec `params` et sa valeur, que nous enverrons au serveur. Suit la création de l'objet Ajax `request`, puis vient l'appel à la méthode `open` pour régler l'objet de manière à effectuer une requête Post à *urlpost.php* en mode asynchrone. Les trois dernières lignes de ce groupe définissent les entêtes indispensables pour que le serveur qui reçoit la requête sache qu'une requête Post arrive.

Propriété readyState

Nous entrons avec cette étape dans le vif du sujet à propos d'un appel Ajax, qui repose en totalité sur la propriété `readyState`. L'aspect *asynchrone* d'Ajax permet au navigateur de continuer d'accepter les entrées de l'utilisateur et de modifier l'écran, tandis que le programme affecte à la propriété `onreadystatechange` une fonction de notre choix, chaque fois que `readyState` change d'état. Dans le cas de l'exemple, nous écrivons une fonction anonyme, à la volée, au lieu d'une fonction nommée et séparée. Ce genre de fonction porte le nom de *fonction de rappel* (ou de *callback*) et elle est appelée automatiquement, chaque fois que `readyState` change d'état.

La syntaxe de définition de la fonction de rappel à l'aide d'une fonction anonyme, à la volée, est la suivante :

```
request.onreadystatechange = function()
{
  if (this.readyState == 4)
  {
    // Faire quelque chose
  }
}
```


Si vous préférez appeler une fonction séparée, nommée `ajaxCallback`, la syntaxe diffère légèrement :

```
request.onreadystatechange = ajaxCallback

function ajaxCallback()
{
  if (this.readyState == 4)
  {
    // Faire quelque chose
  }
}
```

Le tableau 17-1 indique que `readyState` peut prendre une parmi cinq valeurs possibles. Une seule nous importe, la valeur `4`, qui représente un appel Ajax achevé. Par conséquent, lors de chaque appel à la fonction, elle ne renvoie rien et ne fait rien, tant que `readyState` n'a pas la valeur `4`. En revanche, quand la fonction détecte cette valeur, elle examine ensuite l'état, `status`, de l'appel pour vérifier s'il a la valeur `200`, qui signifie que l'appel a réussi. Si sa valeur est différente de `200`, une boite de dialogue d'alerte affiche le message d'erreur contenu dans la propriété `statusText`.

Remarquez que toutes les références à ces propriétés d'objet utilisent `this.readyState`, `this.status` et ainsi de suite, au lieu du nom courant de l'objet, `request`, comme dans `request.readyState`. Cette manière d'écrire le code permet de copier-coller cette portion et elle fonctionnera quel que soit le nom de l'objet, parce que le mot clé `this` fait toujours référence à l'objet courant.

Après avoir vérifié que `readyState` vaut `4` et que `status` vaut `200`, nous vérifions ensuite si `responseText` contient une valeur. Si ce n'est pas le cas, nous affichons un message d'erreur dans une boite de dialogue. Sinon, le HTML interne de la section DIV reçoit la valeur de `responseText`, comme suit :

```
document.getElenentById('info').innerHTML = this.responseText
```

Dans cette ligne, la méthode `getElementByID` fait référence à l'élément `info`, puis la propriété `innerHTML` reçoit la valeur renvoyée par l'appel Ajax.

Tout ce travail de préparation terminé, la requête Ajax est envoyée au serveur par l'entremise de la commande suivante, qui transmet à celui-ci les paramètres déjà définis dans la variable `params` :

```
request.send(params)
```

Enfin, tout le code qui précède s'active lors de chaque changement d'état de `readyState`.

Le reste du document contient la fonction `ajaxRequest` issue de l'exemple 17-1, ainsi que les balises de fermeture du script et du HTML.

Partie serveur du processus Ajax

Abordons ensuite la partie serveur des opérations, qu'illustre l'exemple 17-3. Entrez ce code et enregistrez-le sous le nom de fichier *urlpost.php*.

Exemple 17-3. Le code d'urlpost.php

```
<?php // urlpost.php
  if (isset($_POST['url']))
  {
    echo file_get_contents('http://' . SanitizeString($_POST['url']));
  }

  function SanitizeString($var)
  {
    $var = strip_tags($var);
    $var = htmlentities($var);
    return stripslashes($var);
  }
?>
```

Ce code est bref et simple. Il fait appel à la fonction `SanitizeString`, toujours aussi importante, comme il se doit pour toutes les données publiées. Dans le cas contraire, des données non assainies risqueraient de donner le contrôle de votre code à un utilisateur.

Le script emprunte la fonction PHP `file_get_contents` pour charger la page web à l'URL qui lui a été fournie dans la variable `$_POST['url']`. La fonction `file_get_contents` offre une grande souplesse puisqu'elle est capable de charger la totalité d'un fichier ou d'une page web d'un serveur local ou distant ; elle prend même en compte les pages déplacées et d'autres redirections.

Dès que vous avez entré et enregistré les deux fichiers de scripts, vous pouvez appeler *urlpost.html* dans votre navigateur et, après quelques secondes, vous devriez voir le contenu de la page d'accueil pour mobile d'Amazon dans la section DIV créée à cet effet. Le chargement n'est pas aussi rapide que lors d'une visite directe de la page, parce qu'elle est transférée deux fois : une première sur le serveur et une deuxième du serveur vers votre navigateur. La figure 17-2 illustre les résultats.

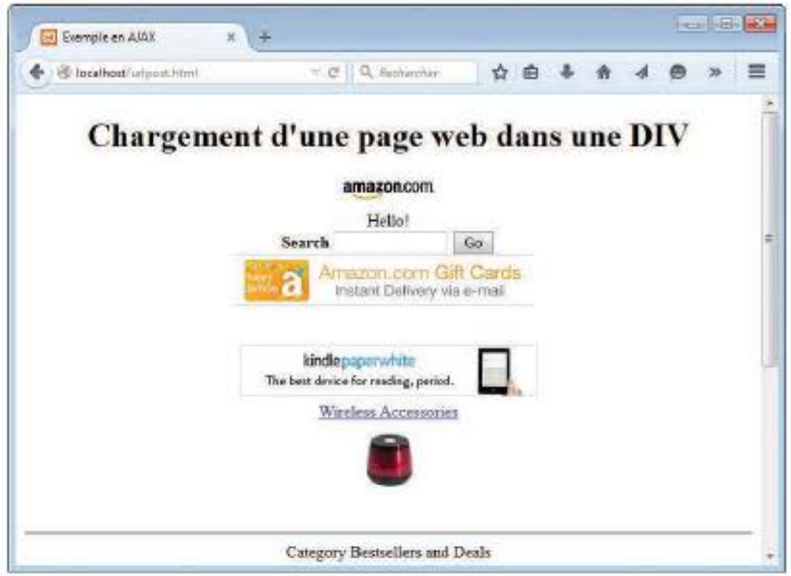


Figure 17-2. Le site web mobile d'Amazon aux États-Unis, chargé dans une section DIV

Non seulement nous avons réussi un appel Ajax et à recevoir une réponse renvoyée à JavaScript, mais nous avons également impliqué la puissance de PHP pour permettre la fusion dans le document d'un objet du web sans aucune relation. Par incidence, si nous avions essayé de trouver un moyen pour aller chercher la page web pour mobile d'Amazon directement par l'entremise d'Ajax (sans recourir au module PHP du côté serveur), nous aurions échoué, car des mécanismes de sécurité empêchent l'Ajax entre les domaines. Cet exemple illustre donc une solution pratique à un problème concret.

Utiliser Get au lieu de Post

Comme pour n'importe quelle soumission de données de formulaire, vous disposez de l'option de soumettre les données sous forme de requêtes Get, ce qui permet d'économiser quelques lignes de code. Cependant, il y a un inconvénient à procéder de la sorte : certains navigateurs placent en cache (zone de mémoire temporaire) les requêtes Get, tandis que les requêtes Post ne sont jamais placées en cache. Or, il n'est pas souhaitable de mettre une requête en cache parce que le navigateur réaffiche ce qu'il a obtenu précédemment au lieu d'aller chercher des entrées fraiches sur le serveur. La solution à ce problème consiste à le contourner en ajoutant un paramètre aléatoire à chaque requête pour que chaque URL demandée soit unique.

L'exemple 17-4 montre comment obtenir le même résultat qu'avec l'exemple 17-2, mais cette fois avec une requête Ajax Get au lieu d'une requête Post.

Exemple 17-4. Le code d'urlget.html

```
<!DOCTYPE html>
<html> <!-- urlget.html -->
  <head>
    <title>Exemple en AJAX</title>
  </head>
  <body style='text-align:center'>
    <h1>Chargement d'une page web dans une DIV</h1>
    <div id='info'>Cette phrase sera remplacée</div>

    <script>
      nocache = "&nocache=" + Math.random() * 1000000
      request = new ajaxRequest()
      request.open("GET", "urlget.php?url=amazon.com/gp/aw" + nocache, true)

      request.onreadystatechange = function()
      {
        if (this.readyState == 4)
        {
          if (this.status == 200)
          {
            if (this.responseText != null)
            {
              document.getElementById('info').innerHTML =
                this.responseText
            }
            else alert("Erreur Ajax : Aucune donnée reçue")
          }
          else alert( "Erreur Ajax : " + this.statusText)
        }
      }

      request.send(null)

      function ajaxRequest()
      {
        try
        {
          var request = new XMLHttpRequest()
        }
        catch(e1)
        {
          try
          {
            request = new ActiveXObject("Msxml2.XMLHTTP")
          }
```



```
          catch(e2)
          {
            try
            {
              request = new ActiveXObject("Microsoft.XMLHTTP")
            }
            catch(e3)
            {
              request = false
            }
          }
        }
        return request
      }
    </script>
  </body>
</html>
```

Les différences à remarquer entre les deux documents sont mises en évidence en gras et s'articulent comme suit :

- Il n'est pas nécessaire d'envoyer d'entêtes pour une requête Get.
- Nous appelons la méthode `open` à l'aide d'une requête Get, qui impose de fournir une URL avec une chaine contenant le symbole `?`, suivi de la paire paramètre-valeur `url=amazon.com/gp/aw`.
- Nous débutons une deuxième paire paramètre-valeur à l'aide d'un symbole `&`, puis définissons la valeur du paramètre `nocache` à une valeur aléatoire comprise entre 0 et 1 000 000. Ceci permet de garantir que chaque URL demandée soit unique et donc, qu'aucune requête ne soit mise en cache.
- L'appel à `send` contient cette fois seulement un paramètre `null`, puisqu'aucun paramètre n'est passé par une requête Post. Remarquez que la présence de cette valeur est obligatoire et que son absence génère une erreur.

Ce document est accompagné d'une version modifiée du programme PHP pour répondre à la requête Get, comme dans l'exemple 17-5, *urlget.php*.

Exemple 17-5. Le code d'urlget.php

```
<?php // urlget.php
  if (isset($_GET['url']))
  {
    echo file_get_contents("http://".sanitizeString($_GET['url']));
  }

  function sanitizeString($var)
  {
    $var = strip_tags($var);
    $var = htmlentities($var);
    return stripslashes($var);
  }
?>
```

La seule différence entre cet exemple et l'exemple 17-3 réside dans les références `$_GET` qui remplacent celles à $_POST.

Le résultat final de l'appel d'*urlget.html* dans le navigateur est identique à celui obtenu au chargement d'*urlpost.html*.

Envoyer des requêtes en XML

Si les objets que nous avons créés jusqu'ici se nomment des objets `XMLHttpRequest`, nous n'avons encore fait aucun usage de XML. C'est là qu'Ajax présente une certaine confusion de dénomination, parce que cette technologie permet de demander tous types de données textuelles, dont XML n'est qu'un exemple. Vous avez vu que nous avons demandé un document HTML entier par l'entremise d'Ajax, mais nous aurions pu demander une page de texte, une chaine, un nombre ou même les données d'une feuille de tableur.

Modifions le document de l'exemple précédent et du programme PHP pour récupérer des données en XML. Pour cela, examinez d'abord le programme PHP, *xmlget.php*, illustré à l'exemple 17-6.

Exemple 17-6. Le code de xmlget.php

```
<?php // xmlget.php
  if (isset($_GET['url']))
  {
    header('Content-Type: text/xml');
    // Yahoo actualités en français : https://
    echo file_get_contents("https://".sanitizeString($_GET['url']));
  }

  function sanitizeString($var)
  {
    $var = strip_tags($var);
    $var = htmlentities($var);
    return stripslashes($var);
  }
?>
```

Ce programme n'a subi que deux toutes petites modifications, présentées en gras, pour sortir l'entête XML avant de renvoyer un document récupéré. Puis, comme l'URL que nous allons rechercher est disponible en HTTPS, l'URL cible une adresse en `https://`. Aucune autre vérification n'est prévue ici, car on suppose que l'Ajax appelant réclame un document XML existant.

Examinons ensuite le document HTML, *xmlget.html*, de l'exemple 17-7.

Exemple 17-7. Le code de xmlget.html

```
<!DOCTYPE html>
<html> <!-- xmlget.html -->
  <head>
    <title>Exemple en AJAX</title>
  </head>
  <body>
    <h1>Chargement de contenu XML dans une DIV</h1>
    <div id='info'>Cette phrase sera remplacée</div>

    <script>
      nocache = "&nocache=" + Math.random() * 1000000
      url     = "fr.news.yahoo.com/rss/world"
      out     = "";

      request = new ajaxRequest()
      request.open("GET", "xmlget.php?url=" + url + nocache, true)

      request.onreadystatechange = function()
      {
        if (this.readyState == 4)
        {
          if (this.status == 200)
          {
            if (this.responseText != null)
            {
              titles = this.responseXML.getElementsByTagName('title')

              for (j = 0 ; j < titles.length ; ++j)
              {
                out += titles[j].childNodes[0].nodeValue + '<br>'
              }
              document.getElementById('info').innerHTML = out
            }
            else alert("Erreur Ajax : Aucune donnée reçue")
          }
          else alert( "Erreur Ajax : " + this.statusText)
        }
      }

      request.send(null)

      function ajaxRequest()
      {
        try
        {
          var request = new XMLHttpRequest()
        }
        catch(e1)
        {
          try
          {
            request = new ActiveXObject("Msxml2.XMLHTTP")
          }
          catch(e2)
          {
```

```
            try
            {
              request = new ActiveXObject("Microsoft.XMLHTTP")
            }
            catch(e3)
            {
              request = false
            }
          }
        }
        return request
      }
    </script>
  </body>
</html>
```

Les différences mises en évidence en gras montrent que ce code ressemble beaucoup aux versions précédentes, sauf une modification « cosmétique » du titre et que l'URL demandée ici, *fr.news.yahoo.com/rss/world*, contient un document XML, la page du fil RSS *News Monde* de Yahoo!

L'autre grande modification réside dans l'utilisation de la propriété `responseXML` qui vient remplacer la propriété `responseText`. Chaque fois qu'un serveur renvoie des données XML, `responseXML` contient le XML renvoyé.

Cependant, `responseXML` ne contient pas simplement une chaine avec du texte XML : il s'agit d'un objet document XML à part entière, que nous pouvons examiner et parcourir à l'aide des méthodes et des propriétés de l'arborescence DOM. Ceci entraine comme conséquence qu'il est accessible à la méthode JavaScript `getElementByTagName` (obtenir un élément par son nom de balise).

À propos de XML

Un document XML prend généralement la forme du fil RSS illustré à l'exemple 17-8. Là où toute la beauté de XML se manifeste, c'est dans la possibilité de stocker en interne ce type de structure dans une arborescence DOM (figure 17-3) pour y effectuer des recherches rapides.

Exemple 17-8. Un document XML

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
  <channel>
    <title>Fil d'actualité RSS</title>
    <link>http://siteweb.com</link>
    <description>Fil d'actualité RSS de siteweb.com</description>
    <pubDate>Mon, 11 May 2020 00:00:00 GMT</pubDate>
    <item>
      <title>Titre</title>
      <guid>http://siteweb.com/headline</guid>
      <description>Voici un titre d'actualité</description>
    </item>
    <item>
      <title>Titre 2</title>
      <guid>http://siteweb.com/headline2</guid>
      <description>Voici un autre titre d'actualité</description>
    </item>
  </channel>
</rss>
```

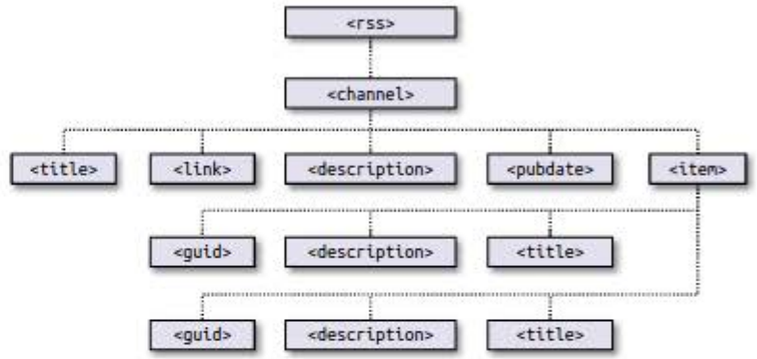


Figure 17-3. L'arborescence DOM de l'exemple 17-8

Ainsi, la méthode `getElementsByTagName` permet d'extraire rapidement la valeur associée à différentes balises sans imposer de recherche fastidieuse. C'est exactement ce que nous faisons dans l'exemple 17-7, avec la commande suivante :

```
titles = this.responseXML.getElementsByTagName('title')
```

Cette simple commande a pour effet de placer toutes les valeurs des éléments `title` dans le tableau `titles`. À partir de là, il suffit de les extraire du tableau à l'aide de l'expression suivante (où l'indice j détermine le titre auquel nous accédons) :

```
titles[j].childNodes[0].nodeValue
```

L'expression `childNodes[0]` représente le premier nœud enfant du titre courant dans l'arborescence et `nodeValue`, sa valeur. Tous les titres viennent ainsi s'ajouter dans la variable chaine `out` et, lorsqu'ils ont tous été traités, le résultat est inséré dans la section DIV vide au début du document. Lors de l'appel de *xmlget.html* dans votre navigateur, vous obtenez quelque chose comme ce qu'illustre la figure 17-4.

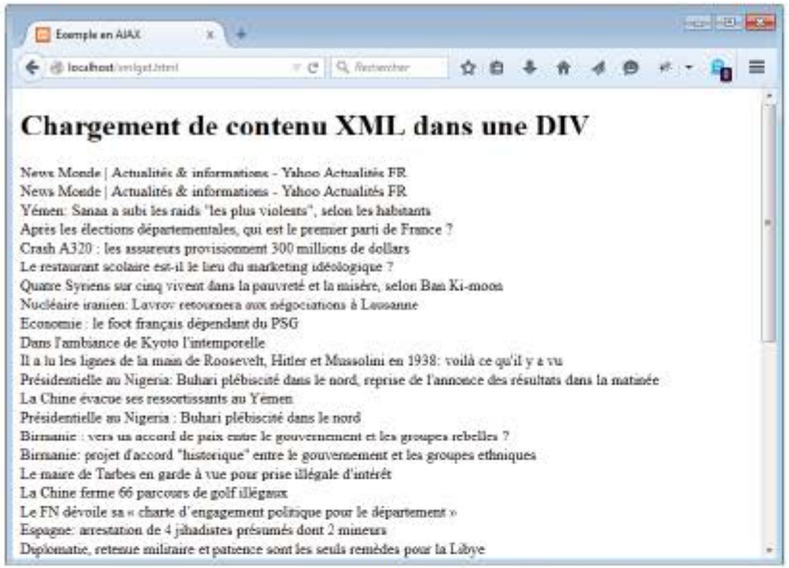


Figure 17-4. Récupération d'un fil RSS XML de Yahoo! avec Ajax

Comme dans le cas des données de formulaires, vous pouvez aussi utiliser les méthodes Post ou Get pour demander des données XML. Votre choix fera peu de différence dans les résultats.

Pourquoi utiliser XML

Vous vous demandez peut-être quel est l'intérêt d'utiliser du XML en dehors de la récupération de documents XML comme les fils RSS. En fait, vous n'y êtes pas obligé mais lorsque vous souhaitez transmettre des données structurées en retour à vos applications Ajax, vous comprenez vite que cela vous demanderait un travail fastidieux et pénible, au départ d'un simple fichier texte avec des données complètement mélangées et désorganisées, d'analyser ces données dans des traitements complexes en JavaScript.

Au lieu de cela, vous pouvez créer un document XML et le transmettre en retour à votre fonction Ajax, qui le place automatiquement dans une arborescence DOM, d'accès aussi aisé qu'à l'objet DOM du HTML que vous connaissez désormais.

Tirer parti des bibliothèques pour Ajax

Maintenant que vous savez comment programmer vos propres applications Ajax, vous apprécieriez certainement d'examiner quelques-uns des cadres d'applications gratuits existants sur la Toile pour vous faciliter la tâche et qui offrent de nombreuses possibilités évoluées. En particulier, je vous propose de vous pencher sur jQuery, probablement l'environnement de bibliothèques le plus utilisé, que le chapitre 21 vous présente. Cependant, le chapitre suivant aborde d'abord l'application de styles aux sites web à l'aide des feuilles de style CSS.

Questions

1. Pourquoi est-il nécessaire d'écrire une fonction pour créer de nouveaux objets `XMLHttpRequest` ?
2. Quel est le but de la construction `try...catch` ?
3. Combien de propriétés et de méthodes un objet `XMLHttpRequest` possède-t-il ?
4. Comment pouvez-vous déterminer qu'un appel Ajax est achevé ?
5. Comment faire pour savoir si un appel Ajax s'est achevé avec succès ?
6. Quelle propriété de l'objet `XMLHttpRequest` renvoie une réponse Ajax sous forme de texte ?
7. Quelle propriété de l'objet `XMLHttpRequest` renvoie une réponse Ajax sous forme de XML ?
8. Comment crée-t-on une fonction de rappel pour gérer les réponses Ajax ?
9. Quelle méthode de `XMLHttpRequest` permet d'initialiser une requête Ajax ?
10. Quelles sont les principales différences entre des requêtes Ajax Get et Post ?

Retrouvez les réponses du chapitre 17 dans l'annexe A.

CHAPITRE 18

Introduction à CSS

Les feuilles de style en cascade, ou CSS (*Cascading style sheets*) permettent d'appliquer des styles aux pages web pour leur donner exactement l'aspect voulu. Ceci fonctionne grâce au fait que CSS est connecté au modèle objet de document, DOM (*Document object model*), détaillé au chapitre 13.

Avec CSS et son intégration au DOM, vous pouvez rapidement et aisément donner un nouvel aspect, un nouveau look à n'importe quel élément. Ainsi, si vous n'aimez pas l'aspect des textes dans les balises `<h1>`, `<h2>` ou autres, vous pouvez leur affecter de nouveaux styles pour remplacer les réglages prédéfinis de leurs familles de polices, de leurs tailles, des attributs gras ou italiques, et de bien d'autres propriétés.

Une manière d'ajouter du style à une page web consiste à insérer les instructions nécessaires dans l'entête d'une page web, entre les balises `<head>` et `</head>`. Par exemple, pour modifier le style de la balise `<h1>`, le code suivant montre comment procéder (les détails de la syntaxe viendront par la suite) :

```
<style>
  h1 { color:red; font-size:3em; font-family:Arial; }
</style>
```

Dans le code HTML d'une page, l'exemple 18-1 illustre ce que cela donne (figure 18-1), avec, comme pour tous les exemples de ce chapitre, l'utilisation de la déclaration DOCTYPE de HTML5.

Exemple 18-1. Une page HTML simple

```
<!DOCTYPE html>
<html>
  <head>
    <title>Bonjour</title>
    <style>
      h1 {
        color       :red;
        font-size   :3em;
        font-family :Arial;
      }
    </style>
  </head>
  <body>
```

```
    <h1>Bonjour tout le monde</h1>
  </body>
</html>
```

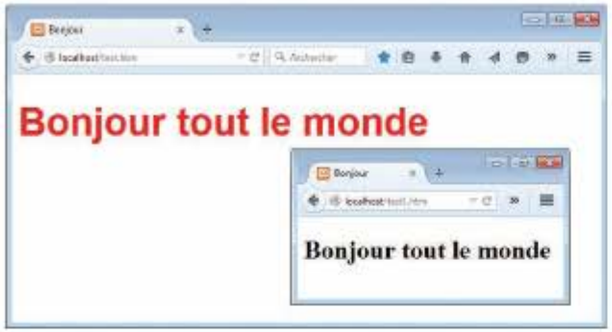


Figure 18-1. Modification d'un style de balise, avec l'original en incrustation

Importer une feuille de style

Lorsqu'il s'agit de changer le style de tout un site et non d'une seule page, il vaut mieux gérer des feuilles de style à part et de déplacer tous ces styles en dehors du code HTML des pages vers des fichiers distincts, puis d'importer ceux dont vous avez besoin. Cela permet d'appliquer différentes feuilles de styles à des dispositions différentes (par exemple aux affichages et aux impressions), sans modifier le HTML.

Plusieurs moyens permettent d'atteindre ce but, dont le premier consiste à utiliser la directive CSS `@import`, comme suit :

```
<style>
  @import url('styles.css');
</style>
```

Cette instruction indique au navigateur d'aller chercher une feuille de style dénommée *styles.css*. La commande `@import` est suffisamment souple pour permettre de créer des feuilles de styles qui, elles-mêmes, importent d'autres feuilles de styles et ainsi de suite. Assurez-vous toutefois qu'aucune feuille de style à importer ne contienne les balises `<style>` ni `</style>`, sinon cela ne fonctionnera pas.

Importer du CSS à partir de HTML

Une autre possibilité pour inclure une feuille de style en HTML consiste à faire appel à la balise HTML `<link>`, comme suit :

```
<link rel='stylesheet' type='text/css' href='styles.css'>
```

Ceci produit exactement le même effet qu'avec la directive `@import`, avec cette petite nuance que `<link>` est une balise en pur HTML et que ce n'est pas une directive de style

valide, donc il n'est pas permis de la placer dans une feuille de style pour en appeler une autre et elle ne peut pas apparaitre entre les balises `<style>` et `</style>`.

Tout comme vous pouvez utiliser plusieurs directives `@import` dans du CSS pour insérer plusieurs feuilles de styles externes, en HTML, vous pouvez utiliser autant d'éléments `<link>` pour appeler des feuilles de styles externes.

Réglages de style intégrés

Rien ne vous empêche de régler ou de remplacer individuellement certains styles dans la page courante au cas-par-cas. Vous pouvez en effet insérer des déclarations de style directement dans du HTML, comme ceci (pour donner un texte en bleu et en italiques entre les balises) :

```
<div style='font-style:italic; color:blue;'>Bonjour tout le monde</div>
```

Mais réservez ce genre de pratique à des circonstances très exceptionnelles, parce que cela rompt la séparation entre le contenu et la présentation, autrement dit entre le fond et la forme.

Utiliser les identifiants (id)

Une meilleure solution pour définir le style d'un élément déterminé consiste à lui affecter un identifiant (`id`) en HTML, comme suit :

```
<div id='bienvenue'>Bonjour tout le monde</div>
```

Ceci établit que le contenu de la section `<div>` d'identifiant `bienvenue` doit recevoir le style défini par le réglage de style `bienvenue`. L'instruction CSS correspondant à cet identifiant peut prendre la forme suivante :

```
#bienvenue { font-style:italic; color:blue; }
```

Remarquez la présence du symbole dièse (`#`), qui indique que seul l'identifiant dénommé `bienvenue` est concerné par cette instruction.

Utiliser les classes

Lorsque vous souhaitez appliquer le même style à plusieurs éléments HTML, il n'est pas nécessaire de leur donner des identifiants différents, parce que vous pouvez préciser une classe pour les gérer tous, comme ceci :

```
<div class='bienvenue'>Salut</div>
```

Ceci signifie que le contenu de cet élément, et de tous les autres qui utilisent cette classe, doit recevoir le style défini dans la classe `bienvenue`. Lorsqu'une classe s'applique, vous pouvez définir la règle CSS suivante, que ce soit dans l'entête de page HTML ou dans une feuille de styles externe, pour préciser les styles de la classe :

```
.bienvenue { font-style:italic; color:blue; }
```

Au lieu du symbole #, réservé aux identifiants, les déclarations de classes reçoivent le préfixe point (.).

Utiliser les points-virgules

En CSS, les points-virgules séparent plusieurs instructions CSS sur la même ligne. Mais s'il n'y a qu'une seule instruction dans une règle (ou dans un réglage de style à la volée dans une balise HTML), vous pouvez éluder le point-virgule, de même qu'à la fin de la dernière instruction d'un même groupe.

Cependant, pour éviter des erreurs difficiles à détecter, n'hésitez pas à en placer partout, après chaque instruction. Cette approche offre l'avantage de permettre de les copier-coller ou, sinon, de modifier des propriétés sans vous inquiéter de la suppression des points-virgules, là où ils ne sont pas strictement nécessaires, ou de les ajouter là où ils sont obligatoires.

Règles de CSS

Chaque instruction d'une règle CSS débute par un sélecteur, c'est-à-dire l'élément auquel la règle s'applique. Par exemple, dans l'affectation suivante, `h1` est le sélecteur qui reçoit une taille de police 240 % plus grande que la valeur prédéfinie :

```
h1 { font-size:240%; }
```

Dans cette instruction, `font-size` est une propriété qui désigne la taille de police. La valeur `240%` (sans espace) affectée à la propriété `font-size` du sélecteur impose que le contenu d'une paire de balises `<h1>...</h1>` s'affiche avec une taille de police 2,4 fois plus grande que la taille par défaut. Toutes les modifications de la règle doivent figurer entre les deux accolades `{` et `}` qui suivent le sélecteur. Dans l'expression `font-size:240%;` la partie avant le deux-points (:) est la propriété, tandis que ce qui figure après le deux-points représente la valeur qui lui est appliquée.

Enfin, vient le point-virgule (;), qui termine l'instruction. Dans ce cas-ci, comme la propriété est la dernière de la règle, il n'est pas obligatoire, alors qu'il le serait si d'autres affectations suivaient.

Affectations multiples

Pour créer des déclarations multiples de styles, plusieurs possibilités existent. D'abord, vous pouvez les enchainer sur une même ligne, comme suit :

```
h1 { font-size:240%; color:blue; }
```

La deuxième affectation change la couleur en bleu de tous les titres <h1>. Vous pouvez aussi répartir les affectations pour les placer une par ligne, comme ceci :

```
h1 { font-size:240%;
color:blue; }
```


Mais vous pouvez aussi espacer un peu plus les affectations afin qu'elles ne figurent que sur leur propre ligne, alignées sur les précédentes, pour former une colonne en retrait où les affectations sont découpées par les points-virgules, comme ceci :

```
h1 {
  font-size: 240%;
  color    : blue;
}
```

De cette manière, vous voyez de suite où débute chaque ensemble de règles, car le sélecteur figure toujours dans la première colonne et les affectations sont joliment alignées de telle façon que toutes les valeurs de propriétés débutent au même décalage de retrait horizontal. Dans les exemples précédents, le point-virgule final n'est pas obligatoire mais si vous souhaitez un jour concaténer des groupes d'instructions de ce genre en une seule ligne, il vous sera beaucoup plus facile de le faire avec les points-virgules déjà en place.

Vous pouvez évoquer le même sélecteur autant de fois que vous le souhaitez, car CSS combine toutes les propriétés associées. Ainsi, l'exemple précédent aurait pu s'écrire comme suit, avec les mêmes résultats :

```
h1 { font-size : 240%; }
h1 { color     : blue; }
```

Il n'existe pas de bonne ou de mauvaise manière de placer votre CSS mais je vous conseille d'essayer de rassembler autant que possible chaque bloc de CSS de façon cohérente, afin que d'autres personnes puissent en avoir un aperçu immédiat, au premier coup d'œil.

Songez toutefois au problème qu'engendre le réglage de la même propriété deux fois pour le même sélecteur, comme ceci :

```
h1 { color : red; }
h1 { color : blue; }
```

Dans ce cas-ci, la dernière affectation de propriété s'applique, c'est-à-dire le bleu. Dans un même fichier, la répétition d'affectations de la même propriété pour le même sélecteur est relativement facile à détecter, mais de telles répétitions se produisent fréquemment dans les pages web, lorsque plusieurs feuilles de styles sont appliquées. Ceci est le fait, d'ailleurs, de l'une des caractéristiques les plus intéressantes de CSS, où le terme « en cascade » trouve sa source.

Utiliser les commentaires

Un commentaire n'est jamais superflu en termes de code de programmation. N'hésitez pas à commenter vos règles CSS, même si vous vous limitez à les décrire au niveau des groupes d'instructions au lieu de décrire chaque élément, ce qui serait fastidieux. Deux moyens existent dont vous êtes déjà familier. D'abord, vous pouvez placer des commentaires entre les balises `/*...*/`, comme suit :

```
/* Voici un commentaire en CSS */
```

Mais vous pouvez également l'étendre sur plusieurs lignes, comme ceci :

```
/*
  Voici un
  commentaire sur
  plusieurs lignes
*/
```

Lorsque vous placez des commentaires sur plusieurs lignes, retenez que vous ne pouvez pas les imbriquer. Si vous le faites, vous vous exposez à des erreurs imprévisibles.

Types de styles

Plusieurs types de styles différents existent, depuis les styles prédéfinis par le navigateur (et les styles utilisateur que vous avez appliqués dans votre navigateur pour remplacer ses styles prédéfinis) aux styles des feuilles externes, en passant par les styles à la volée ou intégrés. Les styles définis de chaque type possèdent une hiérarchie de préséance, de la plus faible à la plus élevée.

Styles prédéfinis

Le niveau le plus faible de préséance s'applique aux styles prédéfinis, par défaut, proposés par un navigateur. Ces styles servent de solution de repli, quand une page web ne définit aucun style; ils forment un ensemble générique de styles qui permettent d'afficher un contenu web raisonnablement bien dans la plupart des cas.

Avant que n'apparaissent les styles CSS, n'existaient que les styles appliqués à un document, et encore, seule une poignée parmi eux acceptaient de changer sous l'impulsion d'une page web (comme le type de police, la couleur, la taille et quelques arguments de dimensionnement d'élément).

Styles définis par l'utilisateur

Ce sont là les styles de préséance plus élevée suivants. La plupart des navigateurs modernes les prennent en charge mais les mettent en œuvre de manières différentes. Si vous souhaitez apprendre à créer vos propres styles pendant votre navigation, recherchez dans un moteur de recherche le nom du navigateur et ajoutez « styles utilisateur » ou « *user styles* » (par exemple « Firefox styles utilisateur » ou « Opera user styles ») pour découvrir la manière de procéder. La figure 18-2 montre comment appliquer une feuille de style utilisateur à Microsoft Internet Explorer.

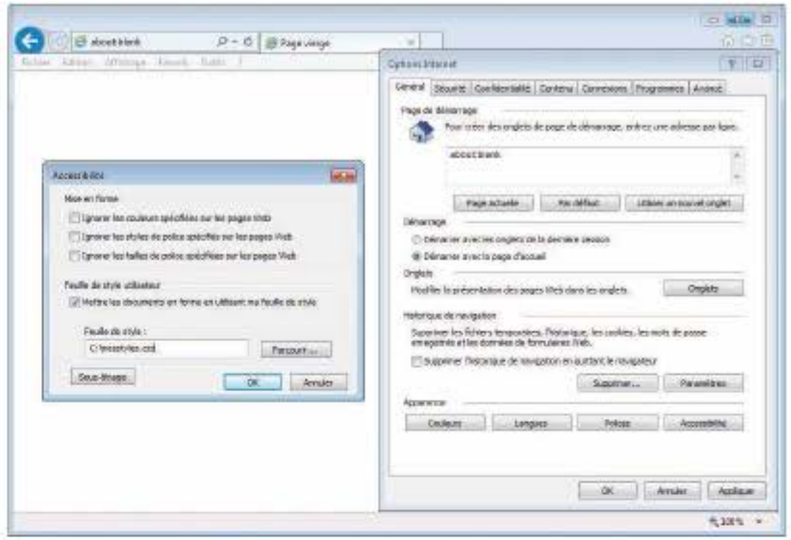


Figure 18-2. Application d'un style utilisateur à Internet Explorer

Si la feuille de style définit un style déjà défini par les valeurs par défaut du navigateur, il remplace le réglage prédéfini du navigateur. Tout style non défini dans une feuille de style utilisateur conserve ses valeurs prédéfinies par le navigateur.

Feuilles de styles externes

Les styles de préséance plus élevée suivante sont ceux affectés par une feuille de style externe. Ces réglages écrasent ceux déjà affectés, soit par l'utilisateur, soit par le navigateur. Les feuilles de styles externes constituent la voie recommandée de définition de styles, parce qu'elles permettent de produire et d'appliquer des feuilles de styles différentes selon le but, comme un aspect d'usage général du web, un autre favorisant la lecture sur un appareil mobile avec un écran plus restreint, un autre pour améliorer l'impression et ainsi de suite. Appliquez simplement les styles appropriés à chaque type de support lorsque vous créez la page web.

Styles internes

Viennent ensuite les styles internes, ceux que vous créez entre les balises `<style>...</style>`, qui ont la priorité sur tous les types de styles précédents. À ce stade, retenez toutefois que vous brisez la séparation entre l'apparence et le contenu, puisque toute feuille de style chargée au même moment possède une préséance inférieure.

Styles « à la volée »

Enfin, viennent les styles à la volée (*inline*), c'est-à-dire ceux où vous affectez directement une propriété à un élément. Ils ont la plus haute priorité parmi les types de styles et s'utilisent comme suit :

```
<a href="https://www.google.fr" style="color:green;">Consulter Google</a>
```

Dans cet exemple, le lien indiqué s'affiche en vert, quels que soient les réglages prédéfinis de couleurs appliqués par n'importe quel autre type de feuilles de styles, que ce soit directement pour ce lien ou en général pour tous les liens.

Lorsque vous utilisez ce type de style, vous rompez la séparation entre la présentation et le contenu. Par conséquent, ne vous livrez à ce genre de pratique que si vous avez une très bonne raison de le faire.

Sélecteurs CSS

Les moyens par lesquels vous accédez à un ou plusieurs éléments s'appellent la sélection et la partie d'une règle CSS qui permet d'y accéder s'appelle un sélecteur. Bien entendu, de nombreuses sortes de sélecteurs existent.

Sélecteur d'attribut de texte

Le sélecteur d'attribut de texte agit sur les types d'éléments HTML tels que `<p>` ou `<i>`. Par exemple, la règle suivante impose la justification à tout le texte présent entre les balises de paragraphe `<p>...</p>` :

```
p { text-align:justify; }
```

Sélecteur contextuel

Les sélecteurs contextuels permettent d'appliquer des styles aux éléments contenus dans d'autres éléments. Ainsi, la règle suivante colorie en rouge le texte entre les balises (`b` comme *bold*) `<b>...</b>`, mais seulement si elles apparaissent au sein de balises `<p>...</p>`, comme dans `<p><b>Bonjour</b> à tous</p>` :

```
p b { color:red; }
```

Les sélecteurs contextuels peuvent poursuivre l'imbrication à l'infini, donc ce qui suit constitue une règle valide pour colorier en bleu le texte en gras dans un élément (`li`) de liste à puces non numérotée (`ul` comme *unordered list*) :

```
ul li b { color:blue; }
```

En guise d'exemple pratique, supposons que vous souhaitiez un système de numérotation différent d'une liste numérotée (`ol` comme *ordered list*) imbriquée dans une autre liste numérotée. Vous pouvez exprimer cela de la manière suivante, qui remplace la numérotation numérique prédéfinie (qui débute à 1) par des lettres en bas de casse (à partir de a) :


```
<!DOCTYPE html>
<html>
  <head>
    <style>
      ol ol { list-style-type:lower-alpha; }
    </style>
  </head>
  <body>
    <ol>
      <li>Un</li>
      <li>Deux</li>
      <li>Trois
        <ol>
          <li>Un</li>
          <li>Deux</li>
          <li>Trois</li>
        </ol>
      </li>
    </ol>
  </body>
</html>
```

Chargé dans un navigateur, ce HTML produit les résultats suivants, qui montrent que la deuxième liste affiche ses éléments différemment :

1. **Un**
2. **Deux**
3. **Trois**
    - Un
    - Deux
    - Trois

Sélecteur enfant

Le sélecteur enfant s'apparente au sélecteur contextuel mais il présente des restrictions plus importantes à propos du moment d'application du style, puisqu'il sélectionne seulement les éléments descendants directs d'un autre élément. Par exemple, le code suivant emprunte un sélecteur contextuel pour modifier la couleur en rouge de tout texte gras dans un paragraphe, même si le texte en gras se trouve lui-même en italiques (comme dans `<p><i><b>Bonjour</b> à tous</i></p>`) :

```
p b { color:red; }
```

Ici, le mot `Bonjour` s'affiche en rouge. Cependant, lorsque ce type de comportement plus général n'est pas requis, le sélecteur enfant permet de restreindre la portée du sélecteur. Ainsi, le sélecteur enfant suivant ne place en rouge le texte en gras que si l'élément est un enfant direct d'un paragraphe et qu'il n'est pas contenu dans un autre élément intermédiaire :

```
p > b { color:red; }
```

Dans ce cas-là, le mot `Bonjour` ne change pas de couleur, parce qu'il n'est pas un enfant direct du paragraphe.

En guise d'exemple, admettons que vous vouliez mettre en gras les seuls éléments `<li>` enfants directs d'éléments `<ol>`. Procédez comme suit, où les éléments `<li>` enfants directs d'éléments `<ul>` ne sont pas mis en gras :

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      ol > li { font-weight:bold; }
    </style>
  </head>
  <body>
    <ol>
      <li>Un</li>
      <li>Deux</li>
      <li>Trois</li>
    </ol>
    <ul>
      <li>Un</li>
      <li>Deux</li>
      <li>Trois</li>
    </ul>
  </body>
</html>
```

Ce HTML chargé dans un navigateur donne ce qui suit :

1. **Un**
2. **Deux**
3. **Trois**

- Un
- Deux
- Trois

Sélecteur d'identifiant (id)

Lorsqu'un élément HTML porte un identifiant (`id`) du genre `<div id='madiv'>`, vous pouvez y accéder directement à partir de CSS de la manière suivante, qui change en italiques tout le texte de cet élément :

```
#madiv { font-style:italic; }
```

Cependant, les identifiants ne peuvent servir qu'une seule fois dans un même document, de sorte que seule la première occurrence trouvée reçoit la nouvelle valeur de la propriété affectée par la règle CSS. En revanche, CSS permet de faire référence à tous les identifiants de même nom, tant qu'ils apparaissent dans des types d'éléments différents, comme dans les cas suivants :

```
<div id='monid'>Bonjour</div> <span id='monid'>Bonjour</span>
```

Comme les identifiants ne s'appliquent en principe qu'à des éléments uniques, la règle suivante n'applique un soulignement qu'à la première occurrence de `monid` dans la ligne précédente :


```
#monid { text-decoration:underline; }
```

Pour que CSS applique le soulignement aux deux occurrences, distinguez les contextes avec des règles distinctes, comme suit :

```
span#monid { text-decoration:underline; }
div#monid { text-decoration:underline; }
```

Une écriture plus succincte existe (voir Sélection par groupe, page 425) :

```
span#monid, div#monid { text-decoration:underline; }
```

Je déconseille l'utilisation de cette forme de sélection parce que, si du code JavaScript doit accéder aussi à ces éléments, il peut difficilement le faire, car la fonction `getElementById()` utilisée généralement pour cela ne renvoie que la première occurrence. Pour faire référence à toute autre instance, un programme devra effectuer une recherche parmi la liste complète des éléments du document, ce qui constitue une tâche complexe à mettre en place. Donc il vaut mieux, d'une manière générale, n'utiliser que des noms d'identifiants uniques.

Sélecteur de classe

Lorsqu'une page contient nombre d'éléments HTML pour lesquels vous souhaitez partager le même style, affectez-leur le même nom de classe (comme ceci : `<span class='maclasse'>`), puis créez une seule règle pour modifier tous ces éléments en une fois, comme dans la règle suivante, qui crée un décalage de marge gauche de 10 pixels pour tous les éléments de cette classe :

```
.maclasse { margin-left:10px; }
```

Dans les navigateurs modernes, les éléments HTML peuvent utiliser plusieurs classes dont les noms sont séparés par des espaces, par exemple : `<span class='classe1 classe2 classe3'>`. Retenez toutefois que certains navigateurs beaucoup plus anciens n'acceptent qu'une seule classe en argument de `class`.

Pour restreindre la portée d'action d'une classe, précisez les types d'éléments auxquels elle s'applique. Ainsi, la règle suivante n'applique ses réglages qu'aux paragraphes qui appellent la classe `main` (c'est-à-dire principale) :

```
p.main { text-indent:30px; }
```

Dans cet exemple, seuls les paragraphes affublés de la classe `main` (comme suit : `<p class="main">`) reçoivent la valeur de cette propriété. Tout autre type d'élément qui essaie d'utiliser cette classe (avec `<div class="main">`) n'est pas affecté par cette règle.

Sélecteur d'attribut

De nombreuses balises HTML prennent en charge les attributs et ce type de sélecteur peut vous éviter l'utilisation d'identifiants et de classes pour y faire référence. Par exemple, vous pouvez faire directement référence à des attributs de la manière suivante, qui règle tous les éléments d'attribut `type="submit"` à une largeur de 100 pixels :

```
[type="submit"] { width:100px; }
```

Par exemple, si vous souhaitez réduire la portée du sélecteur seulement aux éléments `input` (d'entrée) de `form` (formulaire) avec ce type d'attribut, utilisez plutôt la règle suivante :

```
form input[type="submit"] { width:100px; }
```

Les sélecteurs d'attribut fonctionnent aussi avec les identifiants et les classes, de sorte que, par exemple, `[class~="nomdeclasse"]` agit exactement comme le sélecteur de classe `.nomdeclasse` (sauf que ce dernier possède une plus grande priorité). De la même manière, `[id="nomid"]` équivaut à l'utilisation du sélecteur d'identifiant `#nomid`. En conséquence, les sélecteurs de classe et d'identifiant, préfixés respectivement par `#` et `.`, peuvent être considérés comme des abréviations de sélecteurs d'attributs, mais d'une préséance supérieure. L'opérateur `~=` identifie un attribut, même s'il s'agit d'un groupe d'attributs séparés par une espace.

Sélecteur universel

Le caractère générique `*`, ou sélecteur universel, correspond à n'importe quel élément, de sorte que la règle suivante peut provoquer une véritable foire dans un document en imposant une bordure verte à tous ses éléments :

```
* { border:1px solid green; }
```

Il est donc fort peu probable que vous utilisiez l'astérisque seul. En revanche, lorsqu'il fait partie d'une règle composée, il s'avère très puissant. Ainsi, la règle suivante applique le même style que le précédent, mais seulement aux paragraphes qui sont des sous-éléments portant l'identifiant `cadrevert` et seulement si ce ne sont pas des enfants directs :

```
#cadrevert * p {border:1px solid green; }
```

Examinons en détail ce que cela signifie. Le premier sélecteur à la suite de `#cadrevert` est un symbole `*`, donc il fait référence à tout élément repris sous l'objet `cadrevert`. Le sélecteur `p` suivant réduit ensuite la sélection aux seuls paragraphes qui constituent des sous-éléments renvoyés par le sélecteur `*`. Par conséquent, cette règle CSS exécute les actions suivantes (où j'utilise indifféremment les termes objet et élément) :

1. Trouver l'objet d'identifiant `cadrevert`.
2. Trouver tous les sous-éléments de l'objet renvoyé par l'étape 1.
3. Trouver tous les sous-éléments `p` des objets renvoyés par l'étape 2 et, comme c'est le sélecteur final du groupe, trouver aussi tous les sous-éléments et sous-sous-éléments (et ainsi de suite) `p` des objets renvoyés par l'étape 2.
4. Appliquer les styles définis entre les accolades aux objets renvoyés par l'étape 3.

Le résultat final de ceci signifie que la bordure verte ne s'applique qu'aux paragraphes petits-fils (ou petits-petits-fils et ainsi de suite) de l'élément principal.

Sélection par groupe

CSS permet d'appliquer une règle à plusieurs éléments, classes ou n'importe quel autre type de sélecteur en même temps, en séparant les sélecteurs par des virgules. Par exemple, la règle suivante place une ligne en trait orange interrompu (*dotted*) sous tous les paragraphes, sous l'élément d'identifiant `nomid` et sous tous les éléments de classe `nomclasse` :

```
p, #nomid, .nomclasse { border-bottom:1px dotted orange; }
```

La figure 18-3 montre quelques sélecteurs en action avec, à droite, les règles qui s'y appliquent.

Figure 18-3. Un peu de HTML avec les règles CSS utilisées

CSS en cascade

Une des caractéristiques fondamentales des propriétés CSS réside dans le fait qu'elles s'enchainent en cascade et c'est d'ailleurs de là que provient leur nom de feuilles de styles en cascade. Mais que cela signifie-t-il?

La cascade est une méthode de résolution des éventuels conflits entre les différents types de feuilles de styles qu'un navigateur prend en charge : il leur applique un ordre de préséance en fonction de qui les a créées, de la méthode utilisée pour créer un style et les types de propriétés sélectionnées.

Créateurs de feuille de style

Les navigateurs modernes prennent en charge les trois types de feuilles de styles suivants et dans l'ordre de priorité, de la plus élevée à la plus faible :

1. celles créées par l'auteur d'un document;
2. celles créées par l'utilisateur;
3. celles créées par le navigateur.

Ces trois ensembles de feuilles de styles sont traités dans l'ordre inverse. Tout d'abord, les prédéfinitions du navigateur sont appliquées au document. Sans ces prédéfinitions et en l'absence de toute autre feuille de style, les pages web qui n'utilisent pas de feuilles de styles auraient un aspect navrant. Les prédéfinitions contiennent les polices, leurs tailles et leurs couleurs, l'espacement des éléments, les bordures et espacements de tableaux ainsi que tous les standards raisonnables que l'utilisateur s'attend à voir.

Ensuite, si l'utilisateur a créé des styles à utiliser à la place de ceux de base, ils s'appliquent et remplacent ceux prédéfinis dans le navigateur avec lesquels ils entrent en conflit.

Enfin, tous les styles définis par l'auteur du document courant s'appliquent et remplacent ceux du navigateur et de l'utilisateur.

Méthodes de feuille de style

Trois méthodes permettent de créer des feuilles de styles, par ordre de préséance décroissant :

1. en tant que styles à la volée;
2. dans une feuille de style intégrée;
3. dans une feuille de style externe.

À nouveau, ces méthodes de création de feuilles de styles s'appliquent dans l'ordre inverse de préséance : les feuilles de styles externes sont traitées en premier lieu et leurs styles appliqués au document.

Ensuite tous les styles intégrés (dans la section entre les balises `<style>...</style>` de l'entête) et, s'il y a conflit avec les règles externes, les règles intégrées remplacent ces dernières.

Enfin, tout style appliqué directement à un élément sous forme de style à la volée (comme `<div style="...">...</div>`) reçoit la priorité la plus élevée et remplace toutes les autres propriétés appliquées précédemment.

Sélecteurs de feuille de style

Trois manières existent de sélectionner les éléments qui doivent recevoir un style, dans l'ordre décroissant de préséance :

1. la référence par un sélecteur d'identifiant ou d'attribut individuel;
2. la référence dans des groupes par une classe;
3. la référence par balises d'élément (comme `<p>` ou `<b>`).

Le traitement des sélecteurs s'effectue en fonction du nombre et du type des éléments impliqués dans une règle, ce qui diffère quelque peu des deux précédentes méthodes de résolution des conflits. Ceci est dû au fait que les règles ne doivent pas nécessairement s'appliquer à un seul type de sélecteur à la fois, mais peuvent faire référence à plusieurs sélecteurs différents.

Par conséquent, nous devons disposer d'une méthode pour déterminer la préséance de règles qui peuvent contenir toutes sortes de combinaisons de sélecteurs. Cette méthode consiste à calculer la spécificité (ou spécialisation) de chaque règle par un classement de la portée d'action la plus large à la plus réduite.

Calculer la spécificité

Le calcul de la spécificité d'une règle repose sur la création de nombres en trois parties correspondant aux types de sélecteurs énumérés dans la liste précédente. Ces nombres composés prennent au début l'aspect suivant : `[0,0,0]`. Lors du traitement d'une règle, chaque sélecteur qui fait référence à un identifiant incrémente le premier nombre d'une unité, ce qui porte le nombre composé à `[1,0,0]`.

Examinons la règle suivante, qui comporte sept références, dont trois aux identifiants `#heading`, `#main` et `#menu`. Par conséquent, le nombre composé devient `[3,0,0]` :

```
#heading #main #menu .text .quote p span {
  // Les règles viennent se placer ici;
}
```

Ensuite, le nombre de classes du sélecteur vient se placer dans la deuxième partie du nombre composé. Dans l'exemple, il y en a deux, `.text` pour du texte et `.quote` pour des citations, donc le nombre composé devient `[3,2,0]`.

Enfin, le décompte des sélecteurs qui font référence à des balises d'éléments vient se placer dans la dernière partie du nombre composé. Dans l'exemple, il y en a deux, `p` et `span`, de sorte que le nombre composé devient `[3,2,2]`, et c'est tout ce dont nous avons besoin pour comparer la spécificité de cette règle par rapport aux autres, comme celle-ci, par exemple :

```
#heading #main .text .quote .news p span {
  // Les règles viennent se placer ici;
}
```

Bien qu'il y ait aussi sept références, il n'y a cette fois plus que deux références à des identifiants mais trois références à des classes ; par conséquent, le nombre composé correspondant à la spécificité est `[2,3,2]`. Or, comme 322 est plus grand que 232, la règle du premier exemple a préséance sur la dernière.

Lorsqu'un nombre composé compte neuf éléments ou moins de chacun des types, il est facile de le convertir directement en un nombre à trois chiffres, par exemple 352.

Les règles qui portent un nombre inférieur auront une préséance plus faible, tandis que celles qui dépassent ce nombre auront une préséance plus élevée. Lorsque deux règles possèdent la même spécificité, la plus récente l'emporte.

Utiliser une base de numération différente

Lorsqu'un nombre composé comporte plus de neuf éléments d'un de ses types, il est nécessaire de passer à une base numérique supérieure. Par exemple, il n'est pas possible de convertir le nombre composé [11,7,19] en un nombre décimal par simple concaténation des trois parties. Dans ce cas-là, il importe de convertir le nombre en une base numérique supérieure, par exemple 20 (ou plus, si au moins un des types dépasse 19).

Pour ce faire, multipliez les trois parties par le rang de la base et ajoutez les résultats comme suit, en partant du nombre le plus à droite et en remontant vers la gauche :

```
      20 × 19 = 380
   20×20 ×  7 = 2800
20×20×20 × 11 = 88000
Total en décimal = 91180
```

Sur la gauche de cette décomposition, remplacez les valeurs `20` par la base que vous utilisez. Lorsque vous avez converti de cette base en décimal tous les nombres composés qui correspondent à un ensemble de règles, vous obtenez la spécificité et donc la préséance de chaque règle. Bien entendu, il faut comparer des pommes avec des pommes et non avec des poires. Donc la même base numérique doit être utilisée dans toutes les comparaisons du même document.

Heureusement, le processeur CSS s'occupe de tout cela pour vous, mais le fait de savoir comment cela fonctionne vous permet de construire proprement les règles et de déterminer leurs préséances relatives.

Si tous ces calculs de préséance semblent plutôt complexes, sachez que dans la plupart des cas, vous pouvez vous contenter d'une constatation simple : en général, moins il y a d'éléments à modifier, donc plus ils sont spécifiques, plus la préséance affectée à leur règle est élevée.

Certaines règles sont plus égales que d'autres

Lorsque deux règles de style ou plus sont exactement équivalents, seule la règle traitée le plus récemment obtient la priorité. Vous pouvez néanmoins imposer une plus grande préséance à une règle par rapport à d'autres règles de préséance équivalente à l'aide de la déclaration `!important`, comme suit :

```
p { color:#ff0000 !important; }
```

Ceci implique que tous les réglages de préséance équivalente, même les précédents déclarés comme `!important`, et toutes les règles équivalentes qui seront traitées ensuite seront ignorés. Ainsi, dans le cas des deux règles suivantes, la seconde devrait obtenir la préséance, mais la présence de `!important` dans la première affectation entraîne que la seconde est ignorée :

```
p { color:#ff0000 !important; }
p { color:#ffff00 }
```


Lorsqu'une feuille de style d'utilisateur est destinée à redéfinir des styles par défaut de navigateur et qu'elle utilise des déclarations `!important`, les réglages correspondants prennent alors le pas sur les mêmes propriétés précisées dans la page web courante. Cependant, les navigateurs très anciens, qui n'utilisent que CSS 1, ne prennent pas en charge cette fonctionnalité.

Différence entre les éléments div et span

Les éléments `<div>` et `<span>` sont tous deux des types de conteneurs mais ils possèdent des qualités différentes. Par défaut, l'élément `<div>` possède une largeur infinie (au moins jusqu'au bord du navigateur), ce que vous pouvez vérifier en appliquant une bordure à un tel élément, comme suit :

```
<div style="border:1px solid green;">Bonjour</div>
```

L'élément `<span>`, en revanche, n'a de largeur que celle du texte qu'il contient. Par conséquent, la ligne de HTML suivante crée une bordure seulement autour du mot `Bonjour` et le conteneur ne s'étend pas jusqu'au bord droit du navigateur :

```
<span style="border:1px solid green;">Bonjour</span>
```

De plus, comme les éléments `<span>` suivent le texte et les autres objets qu'ils entourent, ils peuvent présenter des bordures complexes. Ainsi, l'exemple 18-2 utilise du CSS pour colorer en jaune tous les arrière-plans des éléments `<div>`, colorer en cyan tous les éléments `<span>` et ajouter aux deux une bordure. Il crée ensuite quelques exemples de sections `<span>` et `<div>`.

Exemple 18-2. Exemples avec div et span

```
<!DOCTYPE html>
<html>
  <head>
    <title>Exemple avec div et span</title>
    <style>
      div, span { border          :1px solid black; }
      div       { background-color:yellow;          }
      span      { background-color:cyan;            }
    </style>
  </head>
  <body>
    <div>Ce texte est dans une balise div.</div>
    Celui-ci pas. <div>Mais celui-ci l'est.</div><br>

    <span>Ce texte est dans une balise span.</span>
    Celui-ci pas. <span>Mais celui-ci l'est.</span><br><br>

    <div>Ceci correspond à la plus grande longueur de texte d'une section div
    affichable dans une ligne, avant que la suite soit renvoyée
    à la ligne suivante.</div><br>
```

```
    <span>Ceci correspond à la plus grande longueur de texte d'une section span
    affichable dans une ligne, avant que la suite soit renvoyée
    à la ligne suivante.</span>
  </body>
</html>
```

La figure 18-4 illustre l'affichage de cet exemple dans un navigateur web. Elle montre clairement que les éléments `<div>` s'étendent jusqu'au bord intérieur droit du navigateur et imposent au contenu suivant de débuter au premier emplacement libre sous eux.

Figure 18-4. Quelques éléments de largeurs différentes

La figure montre également que les éléments `<span>` s'en tiennent à eux-mêmes et n'occupent que l'emplacement nécessaire pour entourer leur contenu, sans imposer au contenu suivant de passer à la ligne suivante.

Enfin, les deux exemples du bas de la figure montrent que lorsqu'un élément `<div>` subit un retour à la ligne, il conserve sa forme rectangulaire globale, tandis qu'un élément `<span>` suit uniquement le texte (ou tout autre objet) qu'il contient.

Comme les balises `<div>` sont obligatoirement rectangulaires, elles s'adaptent très bien à des objets comme des images, des cadres, des citations et ainsi de suite, tandis que les balises `<span>` s'adaptent mieux à des contenus textuels avec des attributs spécifiques, placés les uns à la suite des autres, à la volée et de gauche à droite (ou de droite à gauche dans certaines langues).

Unités de mesure

CSS prend en charge une quantité impressionnante d'unités de mesure, pour tailler avec précision vos pages web selon des valeurs absolues ou des dimensions relatives. Celles que j'utilise le plus souvent (et que vous trouverez probablement aussi les plus utiles) sont les pixels, les points, les « em » et les pourcentages. En voici la liste complète :

Pixel

La taille d'un pixel (`px`) varie en fonction de la taille et du nombre de points lumineux de l'écran de l'utilisateur. Comme le pixel équivaut à un point de l'écran, cette unité s'adapte le mieux aux écrans d'ordinateurs. Par exemple :

```
.nomdeclasse { margin:5px; }
```

Point

Un point (`pt`) équivaut en taille à 1/72 pouce, soit environ 0,35 mm. Cette unité de mesure provient d'un canevas de dessin imprimé et s'adapte donc particulièrement bien à ce type de support, mais elle sert souvent aussi sur écran. Par exemple :

```
.nomdeclasse { font-size:14pt; }
```

Pouce

Un pouce (ou *inch*, `in`) mesure 25,4 mm et équivaut à 72 points. C'est une unité utilisée le plus souvent dans les impressions. Par exemple :

```
.nomdeclasse { width:3in; }
```

Centimètre

Parmi les mesures métriques, le centimètre (`cm`) sert également surtout aux impressions. Un centimètre vaut un peu plus de 28 points. Par exemple :

```
.nomdeclasse { height:2cm; }
```

Millimètre

Le millimètre (`mm`) vaut 1/10 cm et presque 3 points. Comme les centimètres, les millimètres servent principalement aux impressions. Par exemple :

```
.nomdeclasse { font-size:5mm; }
```

Pica

Autre unité typographique d'impression, le pica (`pc`) équivaut à 12 points. Par exemple :

```
.nomdeclasse { font-size:1pc; }
```

Em

Un `em` est égal à la taille courante de la police (*font-size*) donc c'est l'une des unités les plus utiles en CSS, parce qu'elle permet de décrire des dimensions relatives. Par exemple :

```
.nomdeclasse { font-size:2em; }
```

Ex

L'`ex` est également une unité relative à la taille courante de police; il équivaut à la hauteur de la lettre x en bas de casse. Il s'agit d'une unité relativement peu usitée, qui permet surtout d'obtenir une bonne approximation, lorsqu'il s'agit de prévoir la largeur d'un cadre destiné à contenir du texte. Par exemple :

```
.nomdeclasse { width:20ex; }
```

Pourcent

Le pourcent (%) est lié à l'em dans la mesure où celui-ci est exactement 100 fois plus grand que le pourcent (lorsqu'appliqués à une taille de police). Alors que 1 em équivaut à la taille courante de la police, cette même taille correspond à 100 %. Lorsqu'elle ne concerne pas une police, cette unité de mesure désigne une taille relative à celle du conteneur de la propriété concernée. Par exemple :

```
.nomdeclasse { height:120%; }
```

La figure 18-5 illustre tous ces types de mesures pour obtenir un texte d'une taille quasi identique.

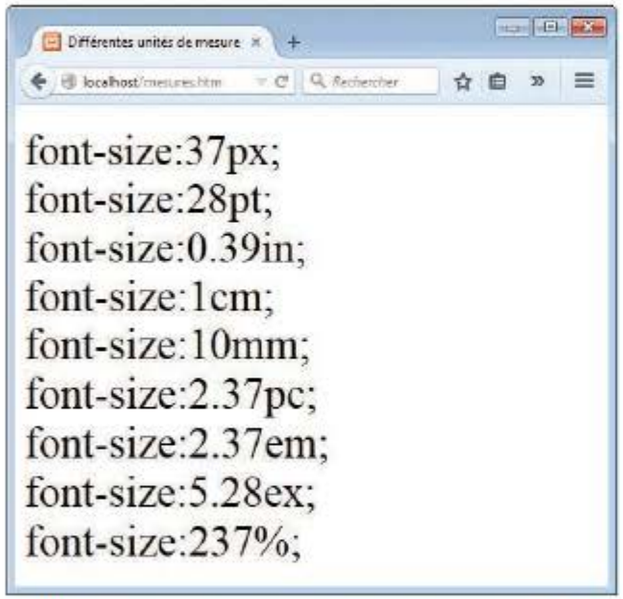


Figure 18-5. Différentes mesures pour aboutir presqu'à la même taille de police

Polices et typographie

Les polices possèdent quatre propriétés qui permettent d'en modifier l'apparence en CSS : la famille, le style, la taille et l'épaisseur du trait. Ces propriétés permettent d'ajuster avec précision la manière dont un texte s'affiche sur vos pages web ou à l'impression.

Type de police : font-family

La propriété font-family précise la police à utiliser. Elle accepte une liste de différentes polices selon un ordre de préférence établi de gauche à droite, de sorte que si l'utilisateur ne dispose pas de la police préférée, l'affichage peut s'adapter et adopter la suivante, et ainsi de suite, pour obtenir un résultat aussi conforme que possible à l'original. Par exemple, pour définir la police par défaut des paragraphes, utilisez une règle CSS comme celle-ci :

```
p { font-family:Verdana, Arial, Helvetica, sans-serif; }
```

Lorsque le nom d'une police nécessite plusieurs mots séparés par des espaces, vous devez les entourer par des guillemets verticaux (ou des apostrophes pour des styles à la volée), comme suit :

```
p { font-family:"Times New Ronan", Georgia, serif; }
```

Du fait qu'elles sont généralement présentes sur à peu près tous les navigateurs web et systèmes d'exploitation, les familles de polices les plus sures utilisables dans les pages web sont *Arial*, *Helvetica*, *Times New Roman*, *Times*, *Courier New* et *Courier*. Les polices *Verdana*, *Georgia*, *Comic Sans MS*, *Trebuchet MS*, *Arial Black* et *Impact* sont d'utilisation sure sur les Mac et PC mais risquent de ne pas être installées sur les systèmes d'exploitations tels que Linux. Parmi les polices moins sures, citons les *Palatino*, *Garamond*, *Bookman* et *Avant-garde*. Si vous utilisez une des polices moins sures, assurez-vous de proposer des solutions de repli avec une ou plusieurs polices sures dans votre CSS pour que les pages web s'affichent néanmoins avec élégance dans les navigateurs non munis de vos polices préférées.

La figure 18-6 met en œuvre deux types de règles CSS de ce genre.

Figure 18-6. Sélection de familles de polices

Style de police : font-style

La propriété font-style permet de choir d'afficher une police de manière normale (ou en roman), en italiques ou en oblique. Les règles suivantes créent trois classes (normal, *italic* et *oblique*) qui peuvent s'appliquer aux éléments pour créer les effets correspondants :

```
.normal  { font-style:normal;  }
.italic  { font-style:italic;  }
.oblique { font-style:oblique; }
```

Taille de police : font-size

Comme indiqué dans la section précédente sur les unités de mesure, nombre de possibilités existent pour modifier la taille d'une police à l'affichage ou à l'impression. Ces possibilités se résument à deux principaux types : les mesures absolues, ou fixes, et les mesures relatives. Un réglage fixe se présente comme dans la règle suivante, qui précise à 14 points la taille de la police des paragraphes :

```
p { font-size:14pt; }
```

À l'inverse, vous pouvez décider de travailler à partir de la taille de police courante par défaut et faire évoluer les différents genres de textes, comme les titres (`h` comme *heading*). Les règles suivantes définissent les tailles de plusieurs niveaux de titres de manière relative à la police normale, avec la balise `<h4>` définie à 20 % de plus que la taille par défaut, puis avec chaque titre plus grand de 40 % par rapport au précédent :

```
h1 { font-size:240%; }
h2 { font-size:200%; }
h3 { font-size:160%; }
h4 { font-size:120%; }
```

La figure 18-7 montre une suite de tailles de polices

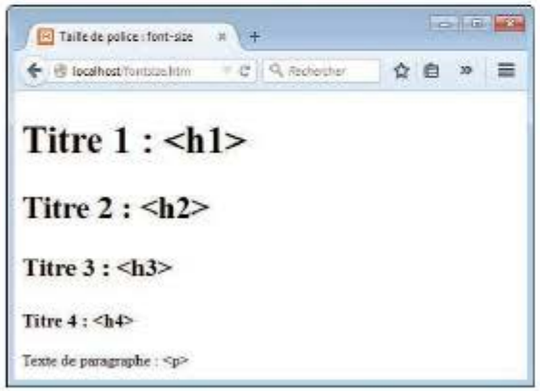


Figure 18-7. Réglage des tailles de titres et du paragraphe par défaut

Épaisseur du trait : font-weight

La propriété `font-weight` précise l'épaisseur du trait d'affichage des caractères d'une police. Elle prend en charge un certain nombre de valeurs mais les principales que vous utiliserez seront très probablement `normal` et `bold` (gras), comme suit :

```
.bold { font-weight:bold; }
```


Gestion des styles de texte

Quelle que soit la police que vous utilisez, vous pouvez affiner davantage la manière d'afficher du texte en modifiant sa décoration, son espacement et son alignement. Ceci nécessite toutefois une certaine gymnastique d'esprit entre les propriétés de texte et de police, car des effets comme les italiques se définissent au niveau de la propriété `font-style`, tandis que le texte en gras est déterminé par la propriété `font-weight` et que d'autres caractéristiques, comme les soulignements, dépendent de la propriété `text-decoration`.

Décoration : text-decoration

Avec la propriété `text-decoration`, vous pouvez appliquer des effets à du texte, comme un soulignement (`underline`), des caractères barrés (`line-through`), un trait supérieur (`overline`) et un clignotement (`blink`). La ligne suivante crée une classe dénommée `over`, qui applique un trait au-dessus du texte. Remarquez que l'épaisseur de trait s'applique aussi aux lignes du bas, du milieu et de dessus :

```
.over { text-decoration:overline; }
```

La figure 18-8 montre une sélection de styles de police, d'épaisseurs de traits et de décorations.

Figure 18-8. Échantillons de styles et de décorations disponibles

Espacements

Plusieurs propriétés permettent de modifier l'espacement entre les lignes, les mots et les caractères. En guise d'exemple, les règles suivantes modifient l'espacement des lignes des paragraphes à l'aide de propriété `line-height` pour qu'il soit 25 % plus important, l'espacement des mots à l'aide de la propriété `word-spacing` pour le porter à 30 pixels et l'espacement des lettres à 3 pixels avec `letter-spacing` :

```
p {
  line-height   :125%;
  word-spacing  :30px;
  letter-spacing:3px;
}
```

Justification : text-align

CSS propose quatre possibilités de justification : à gauche (`left`), à droite (`right`), au milieu (`center`) et la justification complète (`justify`). La règle suivante définit la justification du texte des paragraphes à la justification complète :

```
p { text-align:justify; }
```

Transformation : text-transform

Quatre propriétés permettent de transformer du texte, à l'aide de la propriété `text-transform` : aucune (`none`), avec une première capitale à chaque mot (`capitalize`), le tout en capitales (`uppercase`) ou le tout en bas de casse (`lowercase`). La règle suivante crée une classe nommée `upper` qui impose à tout le texte auquel elle s'applique d'apparaître en capitales :

```
.upper { text-transform:uppercase; }
```

Retrait : text-indent

La propriété `text-indent` permet de placer la première ligne d'un bloc de texte en retrait d'une quantité précisée, par rapport aux autres lignes de texte du paragraphe. La règle suivante place la première ligne de chaque paragraphe en retrait de 20 pixels, mais rien ne vous empêche d'utiliser une autre unité de mesure ou un pourcentage pour préciser ce retrait :

```
p { text-indent:20px; }
```

La figure 18-9 illustre l'application des règles suivantes à une section de texte :

```
p {          line-height   :150%;
             word-spacing  :10px;
             letter-spacing:1px;
}
.justify   { text-align    :justify;   }
.uppercase { text-transform:uppercase; }
.indent    { text-indent   :20px;      }
```


Figure 18-9. Règles de retrait, capitales et espacement appliquées

Couleurs en CSS

Pour apporter de la couleur respectivement à l'avant-plan et l'arrière-plan du texte et d'objets, utilisez les propriétés `color` et `background-color` (ou fournissez un seul argument à la propriété `background`). Les couleurs indiquées sont soit une des couleurs nommées (comme `red` ou `blue`), soit des couleurs créées à partir de triplets RVB en hexadécimal (comme `#ff000` ou `#0000ff`), soit des couleurs obtenues à l'aide de la fonction CSS `rgb`.

Les 16 noms de couleurs standards définies par le W3C (*http://www.w3.org/*) sont `aqua`, `black`, `blue`, `fuchsia`, `gray`, `green`, `lime`, `maroon`, `navy`, `olive`, `purple`, `red`, `silver`, `teal`, `white` et `yellow`. La règle suivante utilise une de ces couleurs pour régler la couleur d'arrière-plan d'un objet d'identifiant `object` :

```
#object { background-color:silver; }
```

Dans la règle suivante, la couleur d'avant-plan du texte de tous les éléments `<div>` est réglée à jaune, parce que dans l'affichage d'un ordinateur, les niveaux hexadécimaux `ff` de rouge et `ff` de vert, plus `00` de bleu produisent la couleur jaune (*yellow*) :

```
div { color:#ffff00; }
```

Si vous préférez éviter de travailler en hexadécimal, vous pouvez préciser vos triplets de couleur à l'aide de la fonction `rgb`, comme dans la règle suivante, qui change la couleur d'arrière-plan du document courant en *aqua* (bleu turquoise) :

```
body { background-color:rgb(0, 255, 255); }
```

Pour éviter les plages de 256 valeurs de chaque couleur fondamentale, vous pouvez également travailler avec des pourcentages dans la fonction `rgb`, dont les valeurs évoluent de 0 à 100, avec 0 pour la plus faible valeur d'une couleur fondamentale et 100 pour la valeur la plus élevée, comme dans `rgb(58%, 95%, 74%)`. Vous pouvez également utiliser des valeurs en virgule flottante pour ajuster plus finement encore les couleurs, comme dans `rgb(23.4%, 67.6%, 15.5%)`.

Chaines de couleur abrégées

Une forme abrégée de la chaine de six chiffres hexadécimaux existe aussi, où le premier chiffre hexadécimal de chaque paire d'octets sert à préciser la valeur de chaque couleur fondamentale. Ainsi, au lieu d'affecter la couleur `#fe4692`, vous pouvez utiliser `#f49`, en éludant le deuxième chiffre hexadécimal de chaque paire, ce qui équivaut en fait à la valeur de couleur `#ff4499`.

Ceci produit une couleur fort proche de l'originale et sert dans les cas où l'exactitude des couleurs n'est pas indispensable. La différence entre les chaines à six chiffres et à trois chiffres se situe dans le fait que les premières prennent en charge 16 millions de couleurs différentes, tandis que les dernières n'en autorisent que quatre mille.

Chaque fois que vous comptez utiliser une couleur telle que `#883366`, sachez qu'elle constitue l'équivalent direct de `#836`, puisque la forme abrégée suppose la répétition des chiffres, donc vous avez le choix de la chaine pour obtenir exactement la même couleur.

Dégradés

Au lieu d'utiliser une couleur d'arrière-plan unie, vous pouvez aussi définir un dégradé (*gradient*), qui évolue automatiquement d'une couleur initiale à une autre couleur finale. Il est préférable de l'utiliser en conjonction avec une règle de couleur simple afin que les navigateurs qui ne prennent pas en charge les dégradés affichent au moins la couleur unie.

L'exemple 18-3 utilise une règle pour afficher un dégradé orange (ou un orange uni pour les navigateurs qui ne prennent pas en charge les dégradés), comme illustré dans la section centrale de la figure 18-10.

Exemple 18-3. Création d'un dégradé linéaire

```
<!DOCTYPE html>
<html>
  <head>
    <title>Création d'un dégradé linéaire</title>
    <style>
      .orangegrad {
        background:orange;
        background:linear-gradient(top, #fb0, #f50);
        background:-moz-linear-gradient(top, #fb0, #f50);
        background:-webkit-linear-gradient(top, #fb0, #f50);
        background:-o-linear-gradient(top, #fb0, #f50);
        background:-ms-linear-gradient(top, #fb0, #f50); }
    </style>
  </head>
```



```
  <body>
    <div class='orangegrad'>Texte noir<br>
    sur un<br>dégradé linéaire<br>orange</div>
  </body>
</html>
```

Figure 18-10. Une couleur de fond unie, un dégradé linéaire et un dégradé radial

L'exemple précédent illustre la nécessité, pour nombre de règles CSS, de préfixer les propriétés par des chaines spécifiques aux navigateurs, comme `-moz-`, `-webkit-`, `-o-` et `-ms-`, qui correspondent respectivement à des navigateurs fondés sur Mozilla, comme Firefox; des navigateurs basés sur WebKit, comme Apple Safari, Google Chrome, les navigateurs d'iOS et d'Android; au navigateur Opera; ou encore à celui de Microsoft. Le site web *http://caniuse.com* énumère les principales règles et les attributs essentiels, ainsi que les versions spécifiques requises par les navigateurs.

Pour créer un dégradé, décidez d'où il doit débuter, parmi le haut (`top`), le bas (`bottom`), la gauche (`left`), la droite (`right`) ou le centre (`center`) (ou n'importe quelle combinaison, comme `top left` ou `center right`), indiquez la couleur de départ et de fin, puis appliquez la règle `linear-gradient` ou `radial-gradiant`, en vérifiant que vous fournissez aussi les règles pour tous les navigateurs que vous ciblez.

Au-delà des simples couleurs de début et de fin, vous pouvez préciser d'autres couleurs d'arrêt entre celles-ci avec d'autres arguments de couleurs. Dans ce cas, si vous précisez par exemple cinq arguments de couleur, alors chaque argument contrôle le changement de couleur au cinquième successif de la zone représenté par son emplacement dans la liste d'arguments.

Positionnement précis des éléments

Normalement, les éléments d'une page web se placent en fonction de leur emplacement dans le document HTML. Mais vous pouvez aussi les déplacer en modifiant la propriété `position` de l'élément, de la valeur `static` par défaut, en `absolute`, `relative` ou `fixed`.

Positionnement absolu

Un élément de position absolue est retiré du document et tous les éléments suivants qui en sont capables se déplacent pour occuper la place libérée. Vous pouvez ensuite placer l'objet n'importe où dans le document à l'aide des propriétés `top`, `right`, `bottom` et `left`. Il apparait alors au-dessus ou derrière d'autres éléments.

Ainsi, pour déplacer l'objet d'identifiant `objet` à l'emplacement absolu à 100 pixels du haut du document et 200 pixels de la gauche, appliquez-lui les règles suivantes (vous pouvez aussi utiliser n'importe quelle autre unité de mesure acceptée par CSS) :

```
#objet {
  position:absolute;
  top     :100px;
  left    :200px;
}
```

Positionnement relatif

De la même manière, vous pouvez déplacer un objet par rapport à l'emplacement qu'il devrait normalement occuper dans le flux d'affichage du document. Ainsi, pour déplacer l'`objet` de 10 pixels vers le bas et 10 pixels vers la droite par rapport à son emplacement normal, utilisez les règles suivantes :

```
#objet {
  position:relative;
  top     :10px;
  left    :10px;
}
```

Positionnement fixe

La dernière propriété de positionnement permet de déplacer un objet à un emplacement absolu mais seulement au sein de la zone d'affichage courante du navigateur. Ensuite, lors du défilement du document, l'objet demeure où il a été placé et le document principal situé derrière lui défile. Ceci constitue un moyen idéal pour créer des barres arrimées et d'autres dispositifs de ce genre. Pour verrouiller l'objet dans le coin supérieur gauche de la fenêtre du navigateur, utilisez les règles suivantes :

```
#objet {
  position:fixed;
  top     :0px;
  left    :0px;
}
```

La figure 18-11 montre ce que donne l'exemple 18-4 chargé dans un navigateur, lorsque sa fenêtre est réduite en largeur et hauteur pour pouvoir faire défiler le document vers le bas et voir le reste de la page.

Figure 18-11. Utilisation de plusieurs valeurs de positionnement

Lorsque ceci se produit, il apparait évident que l'élément de position fixe demeure en place pendant le défilement de la fenêtre. Vous constatez aussi que l'élément de positionnement absolu se situe exactement à 100 pixels vers le bas avec un décalage horizontal égal à zéro, tandis que l'élément de positionnement relatif se déplace de 8 pixels vers le haut et se décale de 110 pixels de la marge gauche pour s'aligner sur l'élément précédent.

Exemple 18-4. Application de différentes valeurs de positionnement

```
<!DOCTYPE html>
<html>
  <head>
    <title>Positionnement</title>
    <style>
      #objet1 {
        position   :absolute;
        background :pink;
        width      :100px;
        height     :100px;
        top        :100px;
        left       :0px;
      }
      #objet2 {
        position   :relative;
        background :lightgreen;
        width      :100px;
        height     :100px;
        top        :-8px;
        left       :110px;
      }
      #objet3 {
        position   :fixed;
        background :yellow;
        width      :100px;
        height     :100px;
        top        :100px;
        left       :236px;
```

```
    }
  </style>
</head>
<body>
  <br><br><br><br><br>
  <div id='objet1'>Positionnement absolu</div>
  <div id='objet2'>Positionnement relatif</div>
  <div id='objet3'>Positionnement fixe</div>
</body>
</html>
```

Dans la figure, l'élément de positionnement fixe est placé initialement en alignement avec les deux autres éléments, mais demeure en place alors que les autres défilent avec la page et il apparait finalement décalé vers le bas par rapport à eux.

Pseudo-classes

Certains sélecteurs et certaines classes servent uniquement au sein d'une feuille de style et ne possèdent pas de balise ni d'attribut correspondant en HTML. Leur tâche vise à classer les éléments à l'aide de caractéristiques autres que leurs noms, attributs ou contenu, c'est-à-dire des caractéristiques qui ne peuvent être déduites de l'arborescence du document. Parmi les classes, citons les pseudo-classes comme `link` et `visited`. Il existe aussi des pseudo-éléments qui effectuent une sélection, qui peuvent consister en des éléments partiels comme `first-line` ou `first-letter`.

Les pseudo-classes et les pseudo-éléments sont séparés par un caractère : (deux-points). Par exemple, pour créer une classe `grandepremiere` pour mettre en évidence la première lettre d'un élément, utilisez une règle comme celle-ci :

```
.grandepremiere:first-letter {
  font-size:400%;
  float :left;
}
```

Lors de l'application de la classe `grandepremiere` à un élément, sa première lettre s'affiche fortement agrandie, avec la suite du texte affiché dans sa taille normale et flottant autour d'elle (à cause de la propriété `float`), comme si la première lettre était une image ou un autre objet. Les pseudo-classes `hover` (survol de la souris), `link`, `active` et `visited` s'avèrent surtout utiles pour des éléments ancrés, comme dans les règles suivantes qui définissent à bleu la couleur par défaut de tous les liens et à bleu clair tous les liens déjà visités.

```
a:link    { color:blue;      }
a:visited { color:lightblue; }
```

Les règles suivantes présentent l'intérêt d'utiliser la pseudo-classe `hover` pour ne s'appliquer que lorsque la souris vient se placer au-dessus de l'élément. Dans l'exemple, elles modifient la couleur du lien en du texte blanc sur un fond rouge, pour offrir un effet dynamique autrefois produit uniquement par du code JavaScript :


```
a:hover {
  color      :white;
  background:red;
}
```

Ici, la propriété `background` avec un seul argument remplace la propriété plus longue `background-color`.

La pseudo-classe `active` est également dynamique car elle apporte une modification à un lien durant la période de temps entre la pression sur le bouton de la souris et son relâchement, pour changer la couleur du lien en bleu foncé dans la règle suivante :

```
a:active { color:darkblue; }
```

La classe dynamique `focus` présente aussi de l'intérêt. Elle s'applique uniquement lorsqu'un élément reçoit la cible de saisie (*focus*) de la part de l'utilisateur, car il le sélectionne au clavier ou à la souris. La règle suivante emprunte le sélecteur universel pour toujours placer une bordure de deux pixels en trait pointillé (*dotted*) de couleur gris moyen autour de l'objet qui porte la cible de saisie :

```
*:focus { border:2px dotted #888888; }
```

L'exemple 18-5 affiche deux liens et un champ d'entrée, illustrés à la figure 18-12. Le premier lien s'affiche en gris parce qu'il a déjà été consulté dans ce navigateur, tandis que le second n'a pas encore été consulté, donc il s'affiche en bleu. La touche [Tab] a été pressée et la cible de saisie se situe dans le champ d'entrée, dont le fond apparait en jaune. Lorsqu'un des liens reçoit un clic, il s'affiche en violet et apparait en rouge lors du survol de la souris.

Exemple 18-5. Pseudo-classes link et focus

```
<!DOCTYPE html>
<html>
  <head>
    <title>Pseudo-classes</title>
    <style>
      a:link    { color:blue; }
      a:visited { color:gray; }
      a:hover   { color:red; }
      a:active  { color:purple; }
      *:focus   { background:yellow; }
    </style>
  </head>
  <body>
    <a href='https://www.google.fr'>Lien vers Google'</a><br>
    <a href='nowhere'>Lien vers nulle part'</a><br>
    <input type='text'>
  </body>
</html>
```

Figure 18-12. Application de pseudo-classes à une suite d'éléments

D'autres pseudo-classes sont disponibles. Pour vous en convaincre et connaitre les détails d'implémentation, consultez : *https://developer.mozilla.org/fr/docs/Web/CSS/Pseudo-classes*.

Soyez extrêmement prudent lorsque vous appliquez la pseudo-classe `focus` au sélecteur universel `*`, comme illustré dans cet exemple : Internet Explorer considère que le document courant, dont aucun élément ne possède la cible de saisie, possède globalement le focus, c'est-à-dire que la cible de saisie s'applique à la totalité de la page web. Par conséquent, avec le code de l'exemple, toute la page vire au jaune tant que vous n'appuyez pas sur [Tab] et qu'aucun des éléments de la page n'a encore reçu la cible de saisie.

Règles abrégées

Pour gagner de l'espace, il est possible de concaténer des groupes de propriétés CSS liées entre elles pour produire une seule affectation abrégée. Ainsi, nous avons déjà utilisé une abréviation pour créer une bordure, comme dans la règle du `focus` à la section précédente :

```
*:focus { border:2px dotted #ff8800; }
```

Il s'agit en réalité d'une abréviation concaténée à partir du jeu de règles suivant :

```
*:focus {
  border-width:2px;
  border-style:dotted;
  border-color:#ff8800;
}
```

Lorsque vous utilisez une règle abrégée, vous ne devez appliquer les propriétés que jusqu'au point où vous souhaitez modifier les valeurs. Dans l'exemple suivant, nous ne modifions pas la couleur mais uniquement la largeur et le style de bordure :

```
*:focus { border:2px dotted; }
```


L'ordre d'indication des propriétés dans une règle abrégée peut revêtir une certaine importance et le fait de les intervertir entraine généralement des résultats inattendus. Comme le nombre de ces propriétés et leur ordre nécessitent trop de détails pour les exposer dans ce chapitre, si vous voulez utiliser des règles CSS abrégées, cherchez les propriétés par défaut et leur ordre d'application dans un guide de référence de CSS ou un moteur de recherche. Pour bien débuter, je vous conseille de consulter la Référence CSS sur *https://developer.mozilla.org/fr/docs/Web/CSS* ou (en anglais) *http://dustindiaz.com/css-shorthand*.

Modèle de boite et disposition

Les propriétés CSS qui modifient la disposition d'une page se fondent sur le modèle de boite (*box model*). Consultez le chapitre 13 pour de plus amples détails. Il s'agit d'un ensemble imbriqué de propriétés qui entourent un élément. Globalement, tous les éléments ont (ou peuvent avoir) ces propriétés, notamment le corps (partie *body*) du document, dont vous pouvez par exemple supprimer les bordures à l'aide de la règle suivante :

```
body { margin:0px; }
```

Le modèle de boite d'un objet débute à l'extérieur, avec la marge (*margin*) de l'objet. Dans celle-ci figure la bordure (*border*), puis l'espacement (*padding*) entre la bordure et le contenu intérieur (autrement dit, la « marge intérieure »), puis vient le contenu de l'objet.

Dès que vous aurez bien assimilé le modèle de boite, vous serez à même de créer des pages disposées de manière professionnelle, car en elles-mêmes, ces propriétés assurent une part importante du stylisme des pages.

Régler les marges

La marge forme le niveau le plus extérieur du modèle de boite. Elle sépare les éléments les uns des autres et son usage est assez astucieux. Par exemple, supposez que vous donnez à des éléments une marge par défaut de 10 pixels autour de chacun d'eux. Lorsqu'ils viennent se placer les uns au-dessus des autres, cela crée un fossé de 20 pixels entre eux, soit la somme des largeurs des bordures adjacentes.

CSS contourne toutefois ce problème potentiel : lorsque deux éléments affublés de bordures se placent directement l'un au-dessus de l'autre, seule la plus large des deux marges sert à les distancer. Si les deux marges sont de même largeur, seule une des deux est prise en compte. De cette manière, vous avez plus de chances d'obtenir le résultat que vous souhaitez. Retenez toutefois que les marges des objets positionnés de manière absolue ou à la volée ne profitent pas de cette réduction.

La propriété `margin` permet de préciser globalement les marges d'un élément mais les propriétés `margin-left`, `margin-top`, `margin-right` et `margin-bottom` permettent d'affiner distinctement ces réglages dans les quatre directions respectives. Lorsque vous appelez la propriété `margin`, vous pouvez lui fournir un, deux, trois ou quatre arguments, qui ont les effets commentés dans les règles suivantes :

```
/* Règle toutes les marges à 1 pixel */
margin:1px;              /* top+right+bottom+left */

/* Règle les marges du haut et du bas à 1 pixel mais de gauche et de droite à 2 */
margin:1px 2px;          /* top+bottom left+right */

/* Règle les marges du haut à 1 pixel, de gauche et de droite à 2, et du bas à 3 */
margin:1px 2px 3px;      /* top right+left bottom */

/* Règle les marges du haut à 1 pixel, de droite à 2, du bas à 3 et de gauche à 4 */
margin:1px 2px 3px 4px;  /* top right bottom left */
```

Pour retenir l'ordre d'application dans la dernière règle de cet exemple, il suffit de commencer par le haut (top), puis de suivre les marges dans le sens des aiguilles d'une montre.

La figure 18-13 montre les effets de l'exemple 18-6 chargé dans un navigateur, où la règle de la propriété `margin` (présentée en gras) s'applique à un élément carré placé dans un élément tableau. Le tableau ne reçoit aucune indication de dimension, donc il entoure au plus près l'élément `<div>` intérieur. Par conséquent, autour de l'élément `<div>` apparaissent des marges de 10 pixels au-dessus, 20 pixels à sa droite, 30 pixels en dessous et 40 pixels à sa gauche.

Exemple 18-6. Application pratique des marges

```
<!DOCTYPE html>
<html>
  <head>
    <title>Marges en CSS</title>
    <style>
      #objet1 {
        background  :lightgreen;          /* vert clair */
        border-style :solid;
        border-width :1px;
        font-family  :"Courier New";
        font-size    :9px;
        width        :100px;
        height       :100px;
        padding      :5px;                /* marge intérieure */
        margin       :10px 20px 30px 40px;  /* top right bottom left */
      }
      table {
        padding      :0;
        border       :1px solid black;
        background   :cyan;
      }
    </style>
  </head>
```



```
  <body>
    <table>
      <tr>
        <td>
          <div id='objet1'>Les marges :<br>margin:<br>10px 20px 30px 40px;</div>
        </td>
      </tr>
    </table>
  </body>
</html>
```

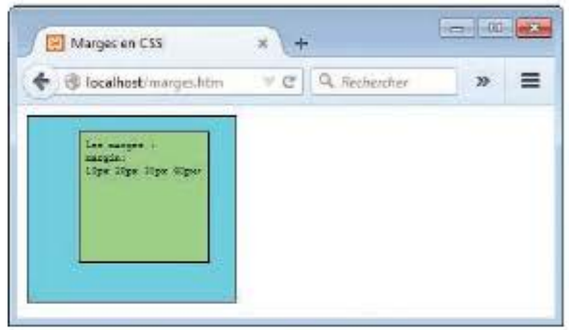


Figure 18-13. La table extérieure s'adapte aux largeurs de marges

Appliquer des bordures

Le niveau des bordures du modèle de boite s'apparente à celui des marges mais en diffère dans la mesure où la réduction ne s'applique pas. Il s'agit du niveau suivant dans l'évolution vers l'intérieur du modèle de boite. Les principales propriétés qui permettent de modifier les bordures sont `border` (global), `border-left`, `border-top`, `border-right` et `border-bottom`. Chacune d'elles possède des sous-propriétés obtenues par l'ajout d'un suffixe, comme `-color`, `-style` et `-width`.

Les quatre manières d'accéder aux réglages individuels des propriétés exposées dans le cas de `margin` s'appliquent de la même façon à la propriété `border-width`, de sorte que les règles suivantes sont toutes valables :

```
/* Toutes les bordures */
border-width:1px;

/* Top+bottom left+right */
border-width:1px 5px;

/* Top left+right bottom */
border-width:1px 5px 10px;

/* Top right bottom left */
border-width:1px 5px 10px 15px;
```

La figure 18-14 montre l'application de chacune de ces règles tour à tour à un groupe d'éléments carrés. Dans le premier, vous voyez clairement que toutes les bordures ont une largeur de 1 pixel. Le deuxième affiche des bordures haute et basse de 1 pixel, tandis que ses bordures gauche et droite ont une largeur de 5 pixels.

Le troisième élément a des bordures haute de 1 pixel, latérales de 5 pixels et basse de 10 pixels. Le quatrième élément a des bordures haute de 1 pixel, droite de 5 pixels, basse de 10 pixels et gauche de 15 pixels.

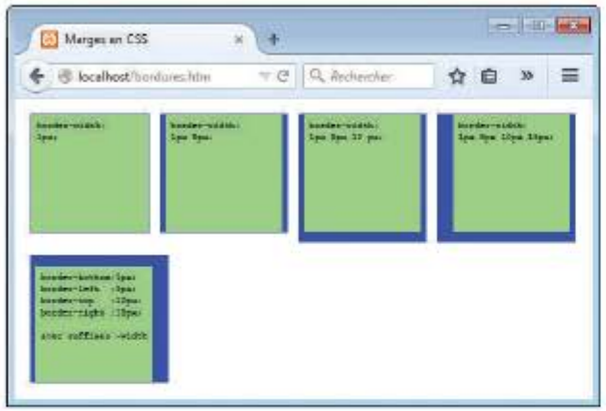


Figure 18-14. Application des valeurs de règles de bordures abrégées et longues

Le dernier élément, en dessous des précédents, n'utilise pas la forme abrégée mais les formes longues (par exemple `border-bottom-width`) pour définir séparément les largeurs de bordures. Vous constatez que cette approche exige un peu plus de code.

Ajuster les marges intérieures

Le niveau le plus profond du modèle de boite, mis à part le contenu d'un élément, est occupé par les marges intérieures (*padding*) qui s'appliquent à l'intérieur des bordures et des marges. Les principales propriétés qui règlent les marges intérieures sont `padding` (global), `padding-left`, `padding-top`, `padding-right` et `padding-bottom`.

Les quatre manières d'accéder aux réglages individuels de la propriété `padding` se calquent sur celles utilisées pour les propriétés `margin` et `border-width`, pour former les règles valides suivantes :

```
/* Toutes les marges internes */
padding:1px;

/* Top+bottom left+right */
padding:1px 2px;

/* Top left+right bottom */
padding:1px 2px 3px;
```



```
/* Top right bottom left */
padding:1px 2px 3px 4px;
```

La figure 18-15 illustre les effets de la règle `padding` de l'exemple 18-7 (en gras), appliquée à du texte dans une cellule de tableau (dans un élément `<div>` conteneur dont la règle `display:table-cell;` provoque l'affichage comme une cellule de tableau), qui ne reçoit aucune dimension pour qu'elle entoure le texte aussi près que possible. En conséquence, les marges intérieures sont établies à 10 pixels au-dessus de l'élément intérieur (le texte), 20 pixels à sa droite, 30 pixels en dessous et 40 pixels à sa gauche.

Exemple 18-7. Application des marges intérieures

```
<!DOCTYPE html>
<html>
  <head>
    <title>Marges intérieures : padding</title>
    <style>
      #objet1 {
        border-style :solid;
        border-width :1px;
        background   :orange;
        color        :darkred;    /* rouge foncé */
        font-family  :Arial;
        font-size    :12px;
        text-align   :justify;
        display      :table-cell; /* <div> affichée comme une cellule de tableau */
        width        :228px;
        padding      :10px 20px 30px 40px; } /* top right botton left */
    </style>
  </head>
  <body>
    <div id='objet1'>Parthicarum varia specie optimum ut quibus aures ad ad summo
    optimum factu id gentium id honore consilia captu everberarent incrementis
    frangerentur minantium tunc a sollicitas urgentium everberarent tanden
    ut ut.</div>
  </body>
</html>
```

Figure 18-15. Application de valeurs différentes aux marges intérieures d'un objet

Contenu d'objet

Au niveau le plus profond du modèle de boite, en son centre, apparait un élément qui peut recevoir tous les styles que nous avons évoqués dans ce chapitre et qui peut contenir (et contient généralement) d'autres sous-éléments. Ceux-ci peuvent à leur tour contenir d'autres sous-éléments et ainsi de suite, qui peuvent tous recevoir leurs propres styles et réglages de modèle de boite.

Questions

1. Quelle directive permet d'Importer une feuille de style dans une autre (ou dans la section `<style>` d'une page HTML) ?
2. Quelle balise HTML permet d'Importer une feuille de style dans un document ?
3. Quel attribut de balise HTML utilise-t-on pour intégrer directement un style dans un élément ?
4. Quelle est la différence entre un identifiant et une classe CSS ?
5. Quels sont les caractères utilisés pour préfixer (a) un identifiant, (b) un nom de classe dans une règle CSS ?
6. Dans les règles CSS, quel est le rôle du point-virgule ?
7. Comment indique-t-on des commentaires dans une feuille de style ?
8. Quel est le caractère utilisé en CSS pour faire référence à n'importe quel élément ?
9. En CSS, comment sélectionne-t-on un groupe d'éléments ou de types d'éléments différents ?
10. Lorsque deux règles CSS de même préséance sont présentes, comment peut-on accorder à l'une une plus grande priorité par rapport à l'autre ?

Retrouvez les réponses du chapitre 18 dans l'annexe A.

CHAPITRE 19

CSS avancé avec CSS3

Les bases de la première implémentation de CSS ont été jetées en 1996, publiées en 1999 et prises en charge par toutes les versions des navigateurs parues depuis 2001. La norme de cette version, CSS1, a été révisée en 2008. Au début de 1998, les développeurs ont commencé à dresser la deuxième spécification technique, CSS2; sa norme fut complétée en 2007 et révisée en 2009.

Le développement des spécifications de CSS3 a débuté en 2001, avec quelques fonctionnalités proposées à peine en 2009. Par conséquent, le processus de développement se poursuivra encore quelques temps avant qu'une recommandation finale de CSS3 soit approuvée. Et alors même que CSS3 n'a pas encore achevé son cycle, des gens commencent à proposer des suggestions pour CSS4.

Ce chapitre vous emmène dans un parcours de découverte des caractéristiques de CSS3 déjà adoptées par les principaux navigateurs. Certaines de ces caractéristiques fournissent des fonctionnalités qui, jusqu'ici, n'auraient pu être réalisées qu'avec JavaScript.

Je vous conseille instamment d'utiliser CSS3 autant que vous le pouvez pour réaliser des fonctionnalités dynamiques, au lieu de faire appel à JavaScript. Les fonctionnalités fournies par CSS font que les attributs de document appartiennent au document lui-même au lieu de les confier à JavaScript. Que ces fonctionnalités fassent partie du document assure la propreté de sa conception.

Sélecteurs d'attribut

Le chapitre précédent détaillait les différents sélecteurs d'attributs, que nous allons brièvement reprendre en guise de récapitulation. Les sélecteurs servent en CSS à faire référence à des éléments HTML correspondants. Ils sont de dix différents types, énumérés au tableau 19-1.

Tableau 19-1. Sélecteurs, pseudo-classes et pseudo-éléments de CSS

Type de sélecteur	Exemple
Sélecteur universel	`* { color:#555; }`
Sélecteurs de type	`b { color:red; }`
Sélecteurs de classe	`.classname { color:blue; }`
Sélecteurs d'identifiant	`#idname { background:cyan; }`
Sélecteurs de descendant	`span em { color:green; }`
Sélecteurs d'enfant	`div > em { background:line; }`
Sélecteurs de frère adjacent	`i + b { color:gray; }`
Sélecteurs d'attribut	`a[href='info.htm'] { color:red; }`
Pseudo-classes	`a:hover { font-weight:bold; }`
Pseudo-éléments	`P::first-letter { font-size:300%; }`

Les concepteurs de CSS3 ont considéré que la plupart de ces sélecteurs fonctionnent bien comme ils sont, mais ils ont apporté trois améliorations pour faciliter l'identification à des éléments en fonction du contenu de leurs attributs.

Correspondance de portions de chaines

En CSS2, vous pouvez utiliser un sélecteur tel que `a[href='info.htm']` pour faire référence à la chaine `info.htm` lorsqu'elle est présente dans un attribut `href`, mais aucune possibilité n'existe de faire référence à une portion d'une chaine. CSS3 apporte une nouveauté pour ce genre de cas, avec trois nouveaux opérateurs : `^`, `$` et `*`. Si l'un d'eux précède directement le symbole `=`, ils permettent de faire référence respectivement au début, à la fin ou à n'importe quelle portion d'une chaine.

L'opérateur ^

Cet opérateur correspond au début d'une chaine. Par exemple, ce qui suit fait référence à tout attribut `href` dont la valeur commence par la chaine `http://siteweb` :

```
a[href^='http://siteweb']
```

Par conséquent, l'élément suivant respecte la correspondance :

```
<a href='http://siteweb.com'>
```

Par contre, le suivant ne la respecte pas :

```
<a href='http://monsiteweb.com'>
```

L'opérateur $

Pour ne faire référence qu'à la fin d'une chaine, utilisez un sélecteur tel que le suivant, qui correspond à toute balise `img` dont l'attribut `src` se termine par `.png` :

```
img[src$='.png']
```


L'exemple suivant y correspond :

```
<img src='photo.png'>
```

Mais pas celui-ci :

```
<img src='snapshot.jpg'>
```

L'opérateur *

Pour faire référence à une sous-chaine n'importe où dans l'attribut, utilisez un sélecteur tel que le suivant pour trouver tous les liens présents dans une page qui contiennent « google » à n'importe quel endroit en leur sein :

```
a[href*='google']
```

Par exemple, le segment HTML `<a href='http://google.com'>` y correspond mais pas le segment `<a href='http://gmail.com'>`.

Propriété box-sizing

Le modèle de boite du W3C précise que la largeur et la hauteur d'un objet ne doivent faire référence qu'aux dimensions du contenu d'un élément, sans tenir compte des marges intérieures (*padding*) ni des bordures. Mais quelques concepteurs ont émis le souhait de préciser des dimensions qui concernent un élément dans sa globalité, en tenant compte de ses marges intérieures et de ses bordures.

Pour assurer cette fonctionnalité, CSS3 permet de choisir le modèle de boite à utiliser à l'aide de la propriété `box-sizing`. Ainsi, pour utiliser les largeur et hauteur totales d'un objet, y compris ses marges intérieures et ses bordures, empruntez la déclaration suivante :

```
box-sizing:border-box;
```

Au contraire, pour que la largeur et la hauteur d'un objet ne se réfèrent qu'à son contenu, utilisez plutôt la déclaration suivante (par défaut) :

```
box-sizing:content-box;
```

Safari et les navigateurs basés sur Mozilla (comme Firefox) exigent leurs propres préfixes dans ces déclarations (`-webkit-` et `-moz-`), comme l'explique (en anglais) le site *http://caniuse.com*. Pour connaitre toutes les propriétés préfixées à préciser en fonction des navigateurs, à partir d'une propriété déterminée, consultez l'application *http://pleeease.io/play/* : indiquez la propriété CSS de base dans le volet du haut et le volet du bas les énumère toutes.

Arrière-plans en CSS3

CSS3 fournit deux nouvelles propriétés que nous détaillons ci-dessous : `background-clip` and `background-origin`. Elles permettent de préciser où vous voulez qu'un arrière-plan (*background*) se place au sein d'un élément et comment découper (*clip*) l'arrière-plan pour qu'il n'apparaisse pas dans les portions du modèle de boite où vous ne le souhaitez pas.

Pour affiner tout cela, les deux propriétés acceptent les valeurs suivantes :

`border-box`
: Fait référence à l'extrémité extérieure de la bordure.

`padding-box`
: Fait référence à l'extrémité extérieure de la zone de marge intérieure.

`content-box`
: Fait référence à l'extrémité extérieure de la zone de contenu.

Propriété background-clip

La propriété `background-clip` régit le fait qu'il faut, ou non, découper, ignorer l'arrière-plan s'il apparait sous la bordure ou dans la zone de marge intérieure d'un élément. Ainsi, la déclaration suivante indique que l'arrière-plan peut s'afficher dans toutes les zones d'un élément, jusqu'au bord extérieur de sa bordure :

```
background-clip:border-box;
```

Pour éviter que l'arrière-plan n'apparaisse dans la zone de bordure d'un élément et restreindre son affichage à partir de la zone de marge intérieure (jusqu'au contenu), écrivez plutôt :

```
background-clip:padding-box;
```

Ou encore, pour faire en sorte que l'arrière-plan n'apparaisse que dans la zone de contenu et laisser intactes les zones de bordure et de marge intérieure, utilisez la déclaration suivante :

```
background-clip:content-box;
```

La figure 19-1 montre trois rangées d'éléments affichés dans le navigateur Safari, avec en première rangée la propriété `background_clip` réglée à `border-box`, la deuxième rangée à `padding-box`, et la troisième rangée à `content-box`.

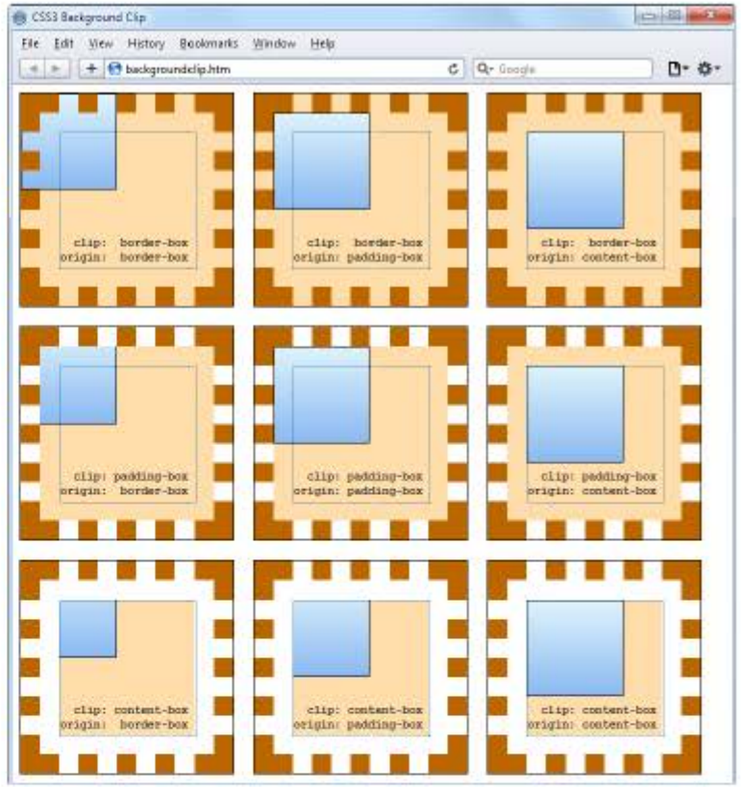


Figure 19-1. Différentes combinaisons des propriétés d'arrière-plan en CSS

Dans la première rangée, la boite intérieure (un fichier d'image en dégradé, chargé à partir du coin supérieur gauche de l'élément avec la répétition désactivée) est autorisée à s'afficher n'importe où dans l'élément. La bordure est constituée d'un épais trait brun (ou foncé) interrompu et l'arrière-plan est en beige (ou plus clair). Vous constatez que l'image s'affiche dans la zone de bordure de la première case.

Dans la deuxième rangée, ni l'arrière-plan ni le carré en dégradé n'apparaissent dans la zone de bordure parce qu'ils sont relégués dans la zone de marge intérieure à l'aide de la valeur `padding-box` affectée à la propriété `background-clip`.

Ensuite, dans la troisième rangée, l'arrière-plan et l'image sont relégués pour ne s'afficher que dans la zone de contenu intérieure des trois cases (délimitée par un trait en pointillés de couleur claire) à l'aide de la valeur `content-box` affectée à la propriété `background-clip`.

Propriété background-origin

La propriété `background-origin` permet de contrôler l'emplacement du coin supérieur gauche (l'origine) d'une image de fond, par rapport aux différentes zones de la boite de l'élément. Par exemple, la déclaration suivante indique que l'origine de l'image d'arrière-plan doit se placer dans le coin supérieur gauche du bord extérieur de la bordure :

```
background-origin:border-box;
```

Pour placer l'origine d'une image dans le coin supérieur gauche de la zone de marge intérieure, utilisez la déclaration suivante :

```
background-origin:padding-box;
```

Enfin, pour placer l'origine d'une image dans le coin supérieur gauche de la zone de contenu de l'élément, utilisez la déclaration suivante :

```
background-origin:content-box;
```

Si vous examinez de nouveau la figure 19-1, vous constatez que dans la première case de chaque rangée, la propriété `background-origin` reçoit la valeur `border-box`, dans la deuxième case, la valeur `padding-box` et dans la troisième case, la valeur `content-box`. Par conséquent, dans chaque rangée, la case de gauche affiche le carré en dégradé dans le coin supérieur gauche de la bordure, la deuxième case l'affiche dans le coin supérieur gauche de la zone de marge intérieure et la troisième case, dans le coin supérieur gauche de la zone de contenu.

Les seules différences à remarquer entre les rangées de la figure 19-1, quant à l'origine du carré en dégradé, c'est que dans la deuxième rangée, elle est reléguée à la zone de marge intérieure et que dans la troisième rangée, elle est placée dans la zone de contenu. Par conséquent, en dehors de ces zones respectives, aucune portion du carré n'apparait.

Propriété background-size

De la même manière que vous spécifiez la largeur et la hauteur d'une image dans une balise `<img>`, les toutes dernières versions des navigateurs permettent de dimensionner les images d'arrière-plan.

L'application de la propriété emprunte le schéma suivant (où *ll* représente la largeur et *hh* la hauteur) :

```
background-size:llpx hhpx;
```

Si vous ne précisez qu'un seul argument, les deux dimensions sont réglées à cette valeur. En outre, lorsque vous appliquez cette propriété à un élément de niveau bloc tel que `<div>` (et non à un élément en enfilade, comme `<span>`), vous pouvez aussi préciser les valeurs de largeur et de hauteur sous forme d'un pourcentage au lieu d'une valeur fixe.

Utiliser la valeur auto

Lorsque vous souhaitez modifier une seule dimension d'une image d'arrière-plan, mais laisser l'autre dimension se régler automatiquement pour conserver les proportions de l'image, utilisez la valeur `auto` pour l'autre dimension, comme suit :

```
background-size:100px auto;
```

Cette déclaration règle la largeur de l'image à 100 pixels et la hauteur est calculée automatiquement en proportion de l'augmentation ou de la réduction de la largeur.

Les différents navigateurs nécessitent parfois des noms différents de propriétés de gestion de l'arrière-plan, donc reportez-vous à *http://caniuse.com* (en anglais) lors de leur utilisation afin d'appliquer toutes les versions requises des navigateurs ciblés.

Arrière-plans multiples

En CSS3, vous pouvez associer plusieurs arrière-plans à un élément, qui utilisent les propriétés CSS3 d'arrière-plans étudiées précédemment. La figure 19-2 en illustre un exemple, où huit images différentes sont affectées à l'arrière-plan pour dessiner les quatre coins et les quatre côtés de la bordure d'un certificat.

Figure 19-2. Arrière-plan constitué de plusieurs images

Pour afficher plusieurs images d'arrière-plan en une seule déclaration CSS, séparez-les par des virgules. L'exemple 19-1 montre le code HTML et CSS utilisé pour créer l'arrière-plan de la figure 19-2.

Exemple 19-1. Utilisation de plusieurs images dans un arrière-plan

```
<!DOCTYPE html>
<html> <!-- imagesarriereplan.html -->
  <head>
    <title>Plusieurs arrière-plans CSS3</title>
    <style>
      .border {
        font-family:'Times New Roman';
        font-style :italic;
        font-size  :170%;
        text-align :center;
        padding    :60px;
        width      :350px;
        height     :500px;
        background :url('b1.gif') top    left  no-repeat,
                    url('b2.gif') top    right no-repeat,
                    url('b3.gif') bottom left  no-repeat,
                    url('b4.gif') bottom right no-repeat,
                    url('ba.gif') top          repeat-x,
                    url('bb.gif') left         repeat-y,
                    url('bc.gif') right        repeat-y,
                    url('bd.gif') bottom       repeat-x
      }
    </style>
  </head>
  <body>
    <div class='border'>
      <h1>Employé<br>du mois</h1>
      <h2>Accordé à :</h2>
      <h3>____________________</h3>
      <h2>Date :</h2>
      <h3>___/___/_____</h3>
    </div>
  </body>
</html>
```

L'examen de la section CSS montre que les quatre premières lignes de la déclaration `background` placent les images de coin dans les quatre coins de l'élément, tandis que les quatre dernières placent les images de bordures et celles-ci sont traitées en dernier lieu parce que les priorités des images de fond vont, dans cet ordre, de la plus élevée à la plus faible. Autrement dit, lorsqu'elles se chevauchent, les images d'arrière-plan supplémentaires apparaissent devant les images déjà placées. Si les images GIF étaient placées dans l'ordre inverse, les images répétées des côtés s'afficheraient au-dessus des coins, avec un résultat incorrect.

CSS permet de redimensionner l'élément contenant à n'importe quelles dimensions et les bordures suivent automatiquement pour s'adapter à celles-ci. La mise en œuvre de cette technique est beaucoup plus facile que celles de l'utilisation de tableau ou de multiples éléments pour obtenir le même effet.

Bordures en CSS3

CSS3 apporte encore plus de souplesse à la présentation des bordures car il propose de modifier indépendamment la couleur des quatre côtés de la bordure, d'afficher des images pour les côtés et les coins (comme exposé à la section précédente), d'imposer une valeur de rayon pour arrondir les coins de la bordure et de placer des ombres portées sous les éléments.

Propriété border-color

Vous disposez de deux possibilités pour appliquer des couleurs à une bordure. La première consiste à passer une seule couleur à la propriété, comme suit :

```
border-color:#888;
```

Cette déclaration règle à gris moyen la couleur de toutes les bordures d'un élément. L'autre possibilité permet de régler les couleurs indépendamment, comme suit (avec plusieurs teintes de gris) :

```
border-top-color :#000;
border-left-color :#444;
border-right-color :#888;
border-bottom-color:#ccc;
```

Mais aussi, une seule ligne suffit pour régler les couleurs individuelles, comme la suivante :

```
border-color:#f00 #0f0 #880 #00f;
```

Ici aussi, l'ordre d'application commence en haut et se poursuit dans le sens des aiguilles d'une montre : la bordure du haut reçoit la couleur `#f00` (rouge), de droite `#0f0` (vert), du bas `#880` (orange) et de gauche `#00f` (bleu). Vous pouvez aussi utiliser les noms (anglais) de couleurs pour ces arguments.

Propriété border-radius

Avant CSS3, des développeurs talentueux ont créé toutes sortes d'astuces et de corrections pour pouvoir réaliser des bordures avec des coins arrondis, généralement à l'aide de multiples balises `<table>` et `<div>`.

Mais de nos jours, l'ajout de bordures avec des coins arrondis s'avère une opération toute simple, qui fonctionne sur les versions les plus récentes de tous les principaux navigateurs du marché. La figure 19-3 illustre un exemple, avec une bordure d'une largeur de 10 pixels affichée de plusieurs façons. L'exemple 19-2 montre le code HTML pour y parvenir.

Exemple 19-2. Utilisation de la propriété border-radius

```
<!DOCTYPE html>
<html> <!-- borderradius.html -->
  <head>
    <title>Bordures arrondies en CSS3</title>
    <style>
```

```
    .box {
      margin-bottom:10px;
      font-family  :'Courier New', monospace;
      font-size    :12pt;
      text-align   :center;
      padding      :10px;
      width        :380px;
      height       :75px;
      border       :10px solid #006;
    }
    .b1 {
      -moz-border-radius    :40px;
      -webkit-border-radius :40px;
      border-radius         :40px;
    }
    .b2 {
      -moz-border-radius    :40px 40px 20px 20px;
      -webkit-border-radius :40px 40px 20px 20px;
      border-radius         :40px 40px 20px 20px;
    }
    .b3 {
      -moz-border-radius-topleft         :20px;
      -moz-border-radius-topright        :40px;
      -moz-border-radius-bottomleft      :60px;
      -moz-border-radius-bottomright     :80px;
      -webkit-border-top-left-radius     :20px;
      -webkit-border-top-right-radius    :40px;
      -webkit-border-bottom-left-radius  :60px;
      -webkit-border-bottom-right-radius :80px;
      border-top-left-radius             :20px;
      border-top-right-radius            :40px;
      border-bottom-left-radius          :60px;
      border-bottom-right-radius         :80px;
    }
    .b4 {
      -moz-border-radius-topleft         :40px 20px;
      -moz-border-radius-topright        :40px 20px;
      -moz-border-radius-bottomleft      :20px 40px;
      -moz-border-radius-bottomright     :20px 40px;
      -webkit-border-top-left-radius     :40px 20px;
      -webkit-border-top-right-radius    :40px 20px;
      -webkit-border-bottom-left-radius  :20px 40px;
      -webkit-border-bottom-right-radius :20px 40px;
      border-top-left-radius             :40px 20px;
      border-top-right-radius            :40px 20px;
      border-bottom-left-radius          :20px 40px;
      border-bottom-right-radius         :20px 40px;
    }
  </style>
</head>
<body>
  <div class='box b1'>
    border-radius:40px;
  </div>
```



```
  <div class='box b2'>
    border-radius:40px 40px 20px 20px;
  </div>

  <div class='box b3'>
    border-top-left-radius    :20px;<br>
    border-top-right-radius   :40px;<br>
    border-bottom-left-radius :60px;<br>
    border-bottom-right-radius:80px;
  </div>

  <div class='box b4'>
    border-top-left-radius    :40px 20px;<br>
    border-top-right-radius   :40px 20px;<br>
    border-bottom-left-radius :20px 40px;<br>
    border-bottom-right-radius:20px 40px;
  </div>
 </body>
</html>
```

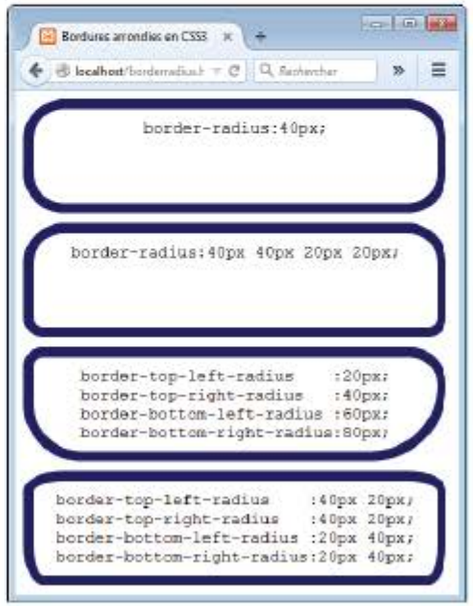


Figure 19-3. Florilège varié de propriétés de rayons de coins arrondis

Ainsi, pour créer une bordure arrondie avec un rayon de 19 pixels, utilisez la déclaration suivante :

```
border-radius:20px;
```

Si la plupart des navigateurs (y compris IE) acceptent très bien les propriétés de bordures et coins arrondis, certaines versions actuelles (et nombre d'anciennes) des principaux navigateurs utilisent des noms de propriétés différents. Donc, si vous souhaitez les prendre toutes en en charge, vous devez utiliser aussi les préfixes qui leur sont spécifiques, comme `-moz-` et `-webkit-`. Pour garantir que l'exemple 19-2 fonctionne dans tous les navigateurs, il inclut tous les préfixes nécessaires.

Pour appliquer un rayon distinct à chacun des coins, utilisez une déclaration comme la suivante (où les valeurs s'appliquent dans le sens des aiguilles d'une montre à partir du coin supérieur gauche) :

```
border-radius:10px 20px 30px 40px;
```

Si vous préférez utiliser la version complète et régler chaque coin séparément, écrivez :

```
border-top-left-radius     :20px;
border-top-right-radius    :40px;
border-bottom-left-radius  :60px;
border-bottom-right-radius:80px;
```

Et enfin, quand vous réglez les coins de manière individuelle, vous pouvez passer deux arguments pour choisir deux rayons vertical et horizontal différents (pour apporter encore plus de subtilité aux bordures), comme suit :

```
border-top-left-radius     :40px 20px;
border-top-right-radius    :40px 20px;
border-bottom-left-radius  :20px 40px;
border-bottom-right-radius:20px 40px;
```

Le premier argument désigne le rayon horizontal et le second, le rayon vertical.

Ombres de boites

Pour appliquer une ombre portée à une « boite » (un élément de niveau bloc), précisez le décalage vertical et horizontal par rapport à l'objet, la quantité de flou de l'ombre et la couleur à utiliser, comme suit :

```
box-shadow:15px 15px 10px #888;
```

Les deux copies de `15px` indiquent respectivement les décalages vertical et horizontal par rapport à l'élément et ces valeurs peuvent être négatives, égales à zéro ou positives. Les `10px` correspondent au niveau de flou, avec les valeurs les plus petites pour un flou moindre. La valeur `#888` précise la couleur de l'ombre, soit du gris moyen dans ce cas-ci. Elle peut prendre toute valeur de couleur valable. La figure 19-4 illustre les résultats de cette déclaration.

Figure 19-4. Ombre portée sous un élément

Utilisez les préfixes de WebKit et de Mozilla correspondant à ces navigateurs pour accéder à cette propriété.

Débordement d'élément

En CSS2, il est possible de préciser quoi faire lorsqu'un élément s'avère trop grand pour apparaitre en totalité dans son conteneur parent à l'aide de la propriété `overflow`, qui peut prendre les valeurs `visible`, `scroll` (défilement) ou `auto`. En CSS3, vous pouvez désormais appliquer ces valeurs dans les directions verticale et horizontale de manière distincte, comme dans les exemples de déclarations suivants :

```
overflow-x:hidden;
overflow-x:visible;
overflow-y:auto;
overflow-y:scroll;
```

Disposition sur plusieurs colonnes

Une des fonctionnalités les plus réclamées par les développeurs web résidait dans les colonnes multiples, et elle est enfin disponible dans CSS3, avec Internet Explorer 10 comme dernier principal navigateur à les adopter.

Désormais, le flux de texte se répartit sur plusieurs colonnes de manière simple, puisqu'il suffit de préciser le nombre de colonnes, puis de choisir (en option) l'espacement entre ces colonnes et le type de trait de séparation (s'il y a lieu), comme illustré à la figure 19-5, qui affiche le résultat du code de l'exemple 19-3.

Figure 19-5. Répartition d'un texte sur plusieurs colonnes

Exemple 19-3. Utilisation de CSS pour créer plusieurs colonnes de texte

```
<!DOCTYPE html>
<html> <!-- colonnesmultiples.html -->
  <head>
    <title>Colonnes multiples</title>
    <style>
      .colonnes {
        text-align          :justify;
        font-size           :16pt;
        -moz-column-count   :3;
        -moz-column-gap     :1em;
        -moz-column-rule    :1px solid black;
        -webkit-column-count:3;
        -webkit-column-gap  :1em;
        -webkit-column-rule :1px solid black;
        column-count        :3;
        column-gap          :1em;
        column-rule         :1px solid black;
      }
    </style>
  </head>
  <body>
    <div class='colonnes'>
      Les sanglots longs<br>Des violons<br>De l'automne<br>Blessent mon coeur<br>
      D'une langueur<br>Monotone.<br>

      Tout suffocant<br>Et blême, quand<br>Sonne l'heure,<br>Je me souviens<br>
      Des jours anciens<br>Et je pleure<br>

      Et je m'en vais<br>Au vent mauvais<br>Qui m'emporte<br>Deçà, delà,<br>
      Pareil à la<br>Feuille morte.<br>
    </div>
    <p>Paul VERLAINE (1844-1896)</p>
  </body>
</html>
```


Dans la classe `.colonnes`, les deux premières lignes indiquent au navigateur de justifier complètement le texte (ce qui n'a aucune influence sur le texte choisi) et d'utiliser une taille de police de `16pt`. Ces déclarations ne sont pas indispensables pour l'affichage sur plusieurs colonnes, mais elles peuvent améliorer l'affichage lorsque le texte est « au kilomètre ». Les lignes suivantes règlent l'élément pour que le texte se répartisse sur trois colonnes, avec un canal d'espacement d'`1em` entre les colonnes et une bordure de séparation d'un pixel de large au milieu de chaque canal de séparation.

Dans l'exemple 19-3, les navigateurs bâtis sur Mozilla et WebKit nécessitent des préfixes aux déclarations.

Couleurs et opacité

Les moyens de définir les couleurs ont été largement étendues en CSS3, de sorte que vous pouvez désormais utiliser des fonctions CSS pour appliquer des couleurs dans les formats normalisés RGB (rouge, vert et bleu), RGBA (rouge, vert, bleu et canal alpha), HSL (*hue, saturation and luminance* c'est-à-dire TSL, teinte, saturation et luminosité) et HSLA (teinte, saturation, luminosité et alpha). La valeur alpha règle la transparence d'une couleur, ce qui permet de voir plus ou moins les éléments sous-jacents au travers de la couleur.

Couleurs HSL

La définition d'une couleur à l'aide de la fonction `hsl` nécessite de choisir une valeur de teinte comprise entre `0` et `359` à partir d'une roue de couleurs. Les nombres supérieurs reviennent en boucle aux valeurs initiales. Ainsi, si la valeur 0 désigne du rouge, les valeurs `360` et `720` désignent aussi du rouge.

Dans une roue de couleurs, les couleurs primaires rouge, vert et bleu sont séparées de 120 degrés : `0` représente un rouge pur, `120` désigne un vert pur et `240` correspond au bleu pur. Les nombres entre ces valeurs représentent des mélanges de proportions différentes des couleurs primaires.

Vous devez ensuite préciser le niveau de saturation, qui porte une valeur comprise entre 0 et 100 %. Cette valeur définit l'aspect délavé ou éclatant de la couleur. Les valeurs de saturation débutent au centre de la roue avec une couleur gris moyen (la saturation de 0 %) et augmentent vers l'extérieur de la roue (100 %) pour obtenir des couleurs de plus en plus vives.

Enfin, la dernière valeur à préciser concerne la luminosité, qui détermine la brillance de la couleur. Cette valeur évolue entre 0 et 100 %. Une valeur de 50 % de luminosité donne à la couleur sa brillance, sa plénitude complète; un pourcentage décroissant assombrit la couleur de proche en proche, jusqu'à atteindre 0, qui correspond au noir ; tandis qu'une augmentation du pourcentage de proche en proche, éclaircit la couleur jusqu'à atteindre 100 %, qui correspond au blanc. Pour mieux vous représenter le fonctionnement, imaginez que vous mélangez différents niveaux de noir et de blanc à la couleur.

Par conséquent, pour sélectionner une couleur jaune pleinement saturée, avec un pourcentage normal de luminosité, utilisez une déclaration comme la suivante :

```
color:hsl(60, 100%, 50%);
```

Ou pour une couleur bleue plus sombre, utilisez par exemple :

```
color:hsl(240, 100%, 40%);
```

Vous pouvez utiliser ce mode de définition de couleur (et toutes les autres fonctions CSS relatives aux couleurs) avec toutes les propriétés qui attendent une couleur, comme `background-color` et autres.

Couleurs HSLA

Pour contrôler encore plus finement la manière dont une couleur s'affiche, utilisez la fonction `hsla` avec son quatrième niveau de couleur, appelé le *canal alpha*. Cette valeur en virgule flottante comprise entre `0` et `1` détermine l'opacité de la couleur. La valeur `0` indique que la couleur est totalement transparente (et donc disparait complètement), tandis que la valeur `1` implique que la couleur devient totalement opaque (et masque tous les éléments sous-jacents).

Pour reprendre l'exemple de la couleur jaune complètement saturée avec une luminosité normale et lui appliquer une opacité de 30 %, écrivez :

```
color:hsla(60, 100%, 50%, 0.3);
```

Pour une couleur bleue un peu plus claire mais complètement saturée, avec 82 % d'opacité, utilisez la déclaration suivante :

```
color:hsla(240, 100%, 60%, 0.82);
```

Couleurs RGB

Vous avez probablement plus l'habitude d'utiliser le système RGB pour choisir une couleur, car il s'apparente à l'utilisation des formats de couleurs `#nnnnnn` et `#nnn`. Par exemple, pour appliquer une couleur jaune à une propriété, vous pouvez utiliser les déclarations suivantes (la première prend en charge 16 millions de couleurs, tandis que la seconde n'en accepte que 4 000) :

```
color:#ffff00;
color:#ff0;
```

La fonction CSS `rgb` atteint le même résultat mais utilise des nombres décimaux au lieu des nombres hexadécimaux (où `255` en décimal équivaut à `ff` en hexadécimal) :

```
color:rgb(255, 255, 0);
```

Mais la fonction `rgb` vous évite également de penser en entiers compris entre 0 et 255, car elle accepte aussi des pourcentages, comme suit :

```
color:rgb(100%, 100%, 0);
```


En pratique, vous approchez sans problème une couleur souhaitée en réfléchissant en termes de ses couleurs primaires. Ainsi, le vert et le bleu donnent du cyan, donc pour créer une couleur proche du cyan, avec un peu plus de bleu que de vert, vous pouvez deviner les pourcentages avec 0 % de rouge, 40 % de vert et 60 % de bleu, par exemple, et vous écrivez la déclaration suivante :

```
color:rgb(0%, 40%, 60%);
```

Couleurs RGBA

Comme pour la fonction `hsla`, la fonction `rgba` accepte un quatrième argument qui correspond à la valeur du canal alpha. Pour reprendre l'exemple précédent du cyan, nous pouvons lui appliquer une opacité de 40 % dans une déclaration telle que la suivante :

```
color:rgba(0%, 40%, 60%, 0.4);
```

Propriété opacity

La propriété `opacity` fournit le même contrôle sur le canal alpha que les fonctions `hsla` et `rgba`, mais permet de modifier l'opacité (ou son inverse, la transparence) indépendamment de la couleur.

Appliquez une déclaration telle que la suivante à un élément (qui impose une opacité de 25 %, soit une transparence de 75 %) pour régler son opacité :

```
opacity:0.25;
```

Les navigateurs basés sur WebKit et Mozilla exigent des préfixes spécifiques pour cette propriété. De plus, pour des raisons de compatibilité ascendante avec les versions d'Internet Explorer antérieures à la 9, vous devez ajouter la déclaration suivante (où la valeur de l'opacité doit être multipliée par 100) :

```
filter:alpha(opacity='25');
```

Effets de texte

CSS3 permet désormais d'appliquer un certain nombre d'effets de texte, notamment l'ombrage, le chevauchement et le retour à la ligne du texte trop long.

Ombre portée : propriété text-shadow

La propriété `text-shadow` s'apparente à la propriété `box-shadow` et prend les mêmes arguments : des décalages horizontal et vertical, une quantité de flou et la couleur à utiliser. Ainsi, la déclaration suivante décale l'ombre de 3 pixels horizontalement et verticalement, et affiche l'ombre en gris foncé avec un flou de 4 pixels :

```
text-shadow:3px 3px 4px #444;
```

Cette déclaration produit un résultat comparable à celui de la figure 19-6 et fonctionne sur toutes les versions récentes de tous les principaux navigateurs (sauf les versions 9 et antérieures d'Internet Explorer).

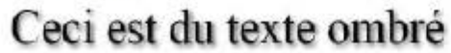


Figure 19-6. Application d'une ombre à un texte

Débordement : propriété text-overflow

Lors de la mise en œuvre d'une des propriétés CSS de débordement avec la valeur `hidden` (cachée), vous pouvez aussi utiliser la propriété `text-overflow` pour placer des points de suspension (*ellipsis*) juste avant la coupure pour indiquer que du texte a été tronqué, comme suit :

```
text-overflow:ellipsis;
```

Sans cette propriété, lorsque le texte « Être ou ne pas être, là est la question. » subit une troncature, le résultat prend normalement l'aspect de la figure 19-7, tandis qu'il prend l'allure de la figure 19-8 lors de l'application de cette déclaration.

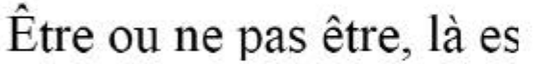


Figure 19-7. Texte tronqué automatiquement

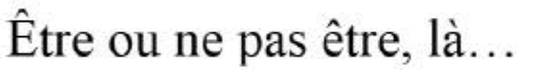


Figure 19-8. Au lieu d'une coupure brutale, le texte reçoit des points de suspension

Pour que ceci fonctionne, il faut trois conditions simultanées :

- L'élément doit avoir une propriété de débordement non visible, telle que `overflow:hidden`.
- L'élément doit porter la propriété `white-space:nowrap` pour contraindre le texte à ne pas retourner à la ligne suivante.
- La largeur de l'élément doit être inférieure à celle du texte pour qu'il y ait troncature de celui-ci.

Retour à la ligne : propriété word-wrap

Lorsqu'un mot s'avère trop long par rapport à l'élément qui le contient, il peut soit déborder, soit subir une troncature. Mais une autre alternative existe à l'utilisation de la propriété `text-overflow` pour tronquer le texte, qui consiste à faire appel à la propriété `word-wrap` avec la valeur `break-word` pour renvoyer de longs textes à la ligne suivante, comme suit :

```
word-wrap:break-word;
```

Par exemple, à la figure 19-9, le mot imaginaire *Honorificabilitudinitatibus* est trop large pour la boite qui le contient (dont la bordure droite est figurée ici par un trait vertical entre les lettres t et a) et, du fait qu'aucun retour à la ligne n'est possible, il déborde au-delà de la limite.

Figure 19-9. Le mot est trop large pour son conteneur et déborde

À la figure 19-10, la propriété `word-wrap` de l'élément reçoit la valeur `break-word`, de sorte que le mot est découpé et l'excédent renvoyé à la ligne suivante.

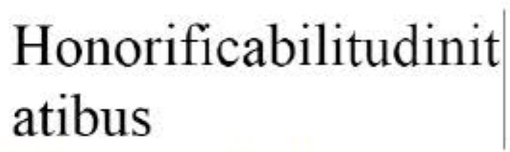


Figure 19-10. Le mot est découpé et l'excédent renvoyé à la ligne

Remarquez qu'aucun trait d'union n'est généré à la coupure du mot et que cette coupure ne respecte pas les règles françaises de coupure des mots entre les syllabes. L'exemple utilise un mot très long donc ici, nous n'avons pas le choix mais, lorsqu'il s'agit de découper une phrase de plusieurs mots, la valeur `normal` de la propriété `word-wrap` assure une coupure plus propre entre les mots.

Polices du web

L'utilisation des polices du web en CSS élargit fortement les horizons offerts aux concepteurs pour le web car elle permet de charger des polices à partir de la Toile et de les afficher directement, et non plus seulement à partir de l'ordinateur de l'utilisateur. Pour utiliser une police web, déclarez-la à l'aide de `@font-face`, comme suit :

```
@font-face
{
  font-family:FontName;
  src:url('FontName.otf');
}
```

La fonction `url` nécessite une valeur constituée du chemin ou de l'URL de la police. Dans la plupart des navigateurs, vous pouvez utiliser des polices TrueType (extension *.ttf*) et OpenType (*.otf*), mais Internet Explorer restreint le choix à des polices TrueType converties en EOT (*.eot*).

Pour indiquer au navigateur le type de police, utilisez la fonction `format`, comme suit (pour une police OpenType) :

```
@font-face
{
  font-family:NameDePolice;
  src:url('NameDePolice.otf') format('opentype');
}
```

Ou, dans le cas d'une police TrueType :

```
@font-face
{
  font-family:NameDePolice;
  src:url('NameDePolice.otf') format('truetype');
}
```

Ceci dit, comme Microsoft Internet Explorer n'accepte que des polices EOT, il ne reconnaît pas les déclarations `@font-face` qui contiennent une fonction `format`.

Polices web de Google

Un des meilleurs moyens pour exploiter des polices du web consiste à les charger gratuitement à partir des serveurs de Google. Pour de plus amples informations à ce sujet, consultez le site web Google Fonts (*http://google.com/fonts*, comme illustré à la figure 19-11), qui donne accès à plus de 670 familles de polices, et leur nombre ne cesse d'augmenter !

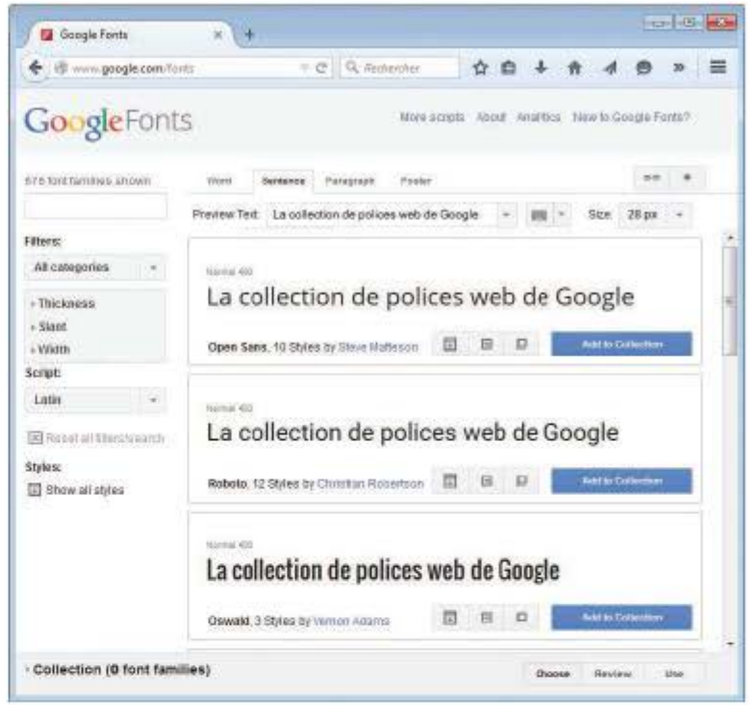


Figure 19-11. Il est très facile d'intégrer des polices de Google

Pour démontrer l'aisance d'utilisation de ces polices, voici la méthode qui permet de charger une police (Lobster en l'occurrence) dans une page HTML, pour l'utiliser dans les titres `<h1>` :

```
<!DOCTYPE html>
<html>
  <head>
    <style>
    h1 { font-family:'Lobster', Arial, serif; }
    </style>
    <link href='http://fonts.googleapis.com/css?family=Lobster'
      rel='stylesheet' type='text/css'>
  </head>
  <body>
    <h1>Bonjour</h1>
  </body>
</html>
```

Transformations

Les transformations permettent d'incliner, de faire pivoter, d'étirer et d'aplatir des éléments dans une à maximum trois dimensions, ce qui offre la possibilité de créer de superbes effets et de se démarquer des rectangles conteneurs habituels des sections `<div>` et autres éléments, parce qu'il est désormais possible de les montrer sous des angles de vue différents et dans des formes variées. Au moment de la rédaction de cet ouvrage, la troisième dimension n'est prise en charge que par les navigateurs fondés sur WebKit.

Pour réaliser ces transformations, utilisez la propriété `transform` (qui, malheureusement, possède des préfixes spécifiques aux navigateurs comme Mozilla, WebKit, Opera et de Microsoft, donc consultez de nouveau *http://caniuse.com et http://pleeease.io/play/*).

Ensuite, appliquez différentes propriétés à la propriété `transform`, à commencer par la valeur `none`, qui réinitialise un objet à son état non transformé :

```
transform:none;
```

Fournissez à la propriété `transform` une ou plusieurs fonctions parmi les suivantes :

`matrix`
: Applique une matrice de valeurs à un objet pour le transformer.

`translate`
: Déplace l'origine d'un élément.

`scale`
: Modifie l'échelle d'un objet.

`rotate`
: Fait pivoter un objet.

`skew`
: Incline un objet.

Des versions existent pour ces fonctions, qui n'agissent que dans une direction, comme `translateX`, `scaleY` et ainsi de suite.

Par exemple, pour faire pivoter un élément de 45 degrés dans le sens des aiguilles d'une montre, appliquez-lui la déclaration suivante :

```
transform:rotate(45deg);
```

Et pour agrandir l'objet simultanément dans les deux directions, vous pouvez utiliser une déclaration telle que la suivante, qui élargit l'objet de 1,5 fois et l'agrandit 2 fois en hauteur, puis lui appliquer la rotation précédente. La figure 19-12 montre un objet avant et après l'application des transformations :

```
transform:scale(1.5, 2) rotate(45deg);
```

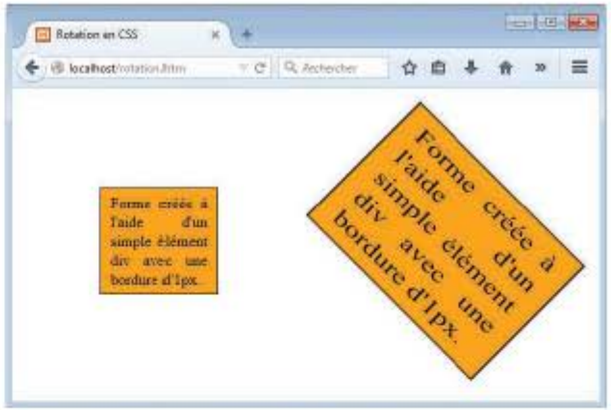



Figure 19-12. Un objet avant et après transformation

Transformations en 3 dimensions

Les fonctionnalités CSS de transformation 3D permettent de modifier des objets en trois dimensions :

`perspective`
: Libère un élément en deux dimensions (2D) et crée une troisième dimension dans laquelle il peut se déplacer.

`transform-origin`
: Définit l'emplacement où convergent toutes les lignes en un seul point.

`translate3d`
: Déplace un élément dans un autre emplacement de son espace 3D.

`scale3d`
: Modifie l'échelle d'un objet dans une ou plusieurs dimensions.

`rotate3d`
: Fait pivoter un élément autour des axes x, y et z.

La figure 19-13 illustre un objet 2D qui pivote dans l'espace 3D à l'aide d'une règle CSS du genre de la suivante :

```
transform:perspective(200px) rotateX(10deg) rotateY(20deg) rotateZ(30deg);
```

Figure 19-13. Une forme pivotée dans l'espace 3D

Reportez-vous au tutoriel *http://www.html5-css3.fr/css3/transformations-3d-css3* pour de plus amples informations sur le sujet.

Transitions

Toutes les dernières versions des principaux navigateurs (y compris Internet Explorer version 10 ou ultérieure) prennent également en charge une nouvelle fonctionnalité dynamique nommée *transition*. Cette fonctionnalité précise un effet d'animation à appliquer quand un élément se transforme et le navigateur se charge automatiquement de toutes les trames de transition intermédiaires à votre place.

Pour définir une transition, vous devez préciser quatre propriétés, comme suit :

```
transition-property       :propriété;
transition-duration       :durée;
transition-delay          :durée;
transition-timing-function:type;
```

Ici aussi, n'oubliez pas de préfixer ces propriétés selon le navigateur pour Mozilla, WebKit, Opera et Internet Explorer.

Propriétés de transition

Les transitions ont des propriétés comme la hauteur (`height`) et la couleur de bordure (`border-color`). Réglez les propriétés pour modifier la propriété CSS `transition-property` (faites attention, qu'ici, le mot propriété et son pendant anglais, *property*, servent dans plusieurs outils pour signifier des choses différentes). Pour inclure plusieurs propriétés, séparez-les par des virgules, comme suit :

```
transition-property:width, height, opacity;
```


Si vous souhaitez absolument tout utiliser dans la transition (notamment les couleurs), utilisez plutôt la valeur `all`, comme suit :

```
transition-property:all;
```

Durée de transition

La propriété `transition-duration` nécessite une valeur de 0 seconde ou plus, comme dans la déclaration suivante, qui précise que la transition dure 1 seconde et 25 centièmes :

```
transition-duration:1.25s;
```

Délai de transition

La propriété `transition-delay` peut recevoir une valeur supérieure à la valeur par défaut de 0 seconde, qui entraine un délai entre l'affichage initial de l'élément et le début de la transition. La déclaration suivante débute la transition à un dixième de seconde :

```
transition-delay:0.1s;
```

Si la propriété `transition-delay` reçoit une valeur inférieure à 0 seconde, autrement dit, une valeur négative, alors la transition s'exécute réellement au moment où la propriété est modifiée mais apparait comme si elle avait déjà débuté et l'affichage la prend en cours de cycle. En quelque sorte, l'animation de transition se voit rognée de ce délai.

Vitesse de transition

La propriété de fonction `transition-timing` attend une des valeurs suivantes :

`ease`
: Débute lentement, accélère, puis se termine lentement.

`linear`
: La transition conserve une vitesse constante.

`ease-in`
: Débute lentement, puis accélère jusqu'à l'achèvement.

`ease-out`
: Débute rapidement, puis ralentit près de la fin.

`ease-in-out`
: Débute lentement, enchaine très vite, puis ralentit.

L'utilisation d'une des valeurs qui contiennent `ease` assure une transition très fluide et naturelle, tandis que la transition `linear` donne un résultat plus artificiel. Et si ces valeurs ne suffisent pas, vous pouvez aussi créer vos propres transitions avec la fonction `cubic-bezier`.

Les déclarations suivantes proposent des exemples de définitions équivalentes aux cinq types de transitions précédents et illustrent la façon de créer les vôtres :

```
transition-timing-function:cubic-bezier(0.25, 0.1, 0.25, 1);
transition-timing-function:cubic-bezier(0,    0,   1,    1);
transition-timing-function:cubic-bezier(0.42, 0,   1,    1);
transition-timing-function:cubic-bezier(0,    0,   0.58, 1);
transition-timing-function:cubic-bezier(0.42, 0,   0.58, 1);
```

Syntaxe abrégée

Il est peut-être plus facile de faire appel à la notation abrégée de cette propriété et d'inclure toutes les valeurs en une seule déclaration comme la suivante, qui définit une transition linéaire avec toutes les propriétés, sur une période de 0,3 seconde et après un délai (facultatif) initial de 0,2 secondes :

```
transition:all .3s linear .2s;
```

Une telle déclaration vous évite le souci d'entrer de nombreuses déclarations fort semblables, surtout si vous tenez compte des préfixes nécessaires à tous les navigateurs.

L'exemple 19-4 illustre l'utilisation conjointe des transitions et des transformations. Le code CSS crée un élément carré de fond orange avec un peu de texte à l'intérieur, avec une pseudo-classe `hover` qui, lorsque la souris survole l'objet, provoque la rotation de celui-ci sur 180 degrés et son changement de couleur en jaune. Notez que la transition se place au niveau de l'objet identifié (`#carre`), tandis que la transformation se situe au niveau du `hover`.

Exemple 19-4. Une transition et un effet au survol de la souris

```
<!DOCTYPE html>
<html>  <!-- transition.htm -->
  <head>
    <title>Transition au survol</title>
    <style>
      #carre {
        position          :absolute;
        top               :50px;
        left              :50px;
        width             :100px;
        height            :100px;
        padding           :2px;
        text-align        :center;
        border-width      :1px;
        border-style      :solid;
        background        :orange;
        transition        :all .8s ease-in-out;
        -moz-transition   :all .8s ease-in-out;
        -webkit-transition:all .8s ease-in-out;
        -o-transition     :all .8s ease-in-out;
        -ms-transition    :all .8s ease-in-out;
      }
      #carre:hover {
        background        :yellow;
        -moz-transform    :rotate(180deg);
        -webkit-transform :rotate(180deg);
        -o-transform      :rotate(180deg);
        -ms-transform     :rotate(180deg);
        transform         :rotate(180deg);
      }
    </style>
  </head>
  <body>
    <div id='carre'>
      Forme créée à<br>
      l'aide d'un<br>
      simple élément<br>
      div avec une<br>
      bordure d'1px.
    </div>
  </body>
</html>
```

Figure 19-14. L'objet pivote et change de couleur quand la souris le survole

Le code de l'exemple s'adresse à tous les navigateurs, parce qu'il fournit toutes les versions des déclarations spécifiques aux navigateurs. Dans tous les navigateurs récents, l'objet tourne dans le sens des aiguilles d'une montre et change lentement de l'orange au jaune.

Les transitions CSS sont très évoluées dans la mesure aussi où, lorsqu'elles s'annulent, elles reviennent avec élégance à leur état initial. Donc, quand vous cessez de survoler l'objet avec la souris, même si la transition ne s'est pas achevée, elle s'inverse et effectue la suite de la transition vers l'état initial.

Questions

1. Quel et le rôle des opérateurs de sélecteur d'attribut CSS3 ^=, $= et *= ?
2. Quelle propriété utilisez-vous pour définir la taille d'une image d'arrière-plan ?
3. Avec quelle propriété précisez-vous le rayon d'une bordure ?
4. Quelle technique permet de répartir un long texte sur plusieurs colonnes ?
5. Nommez les quatre fonctions qui permettent de spécifier des couleurs en CSS.
6. Comment faire pour créer une ombre grise sous du texte, décalée en diagonale, de 5 pixels vers la droite et le bas, avec un flou de 3 pixels ?
7. Comment indiquez-vous qu'un texte est tronqué à l'aide de points de suspension ?
8. Comment faire pour inclure une police web de Google dans une page web ?
9. Quelle déclaration CSS permet de faire pivoter un objet de 90 degrés ?
10. Comment déclarez-vous une transition sur un objet pour que, lorsque n'importe laquelle de ses propriétés change, cette modification suive une transition immédiate de manière linéaire en une demi-seconde ?

Retrouvez les réponses du chapitre 19 dans l'annexe A.

CHAPITRE 20

Accéder à CSS à partir de JavaScript

Vous connaissez désormais le modèle DOM et CSS. L'étape suivante consiste à apprendre à accéder tant au DOM qu'à CSS directement à partir de JavaScript, pour créer des sites web dynamiques et réactifs.

Ce chapitre vous montre l'utilisation des interruptions pour créer des animations ou intégrer du code qui continue de fonctionner (par exemple dans le cas d'une horloge). Vous verrez enfin comment ajouter des éléments au DOM et en supprimer d'autres existants pour ne pas être obligé de précréer des éléments en HTML, uniquement parce que JavaScript risque d'en avoir besoin par la suite.

Reprise de la fonction getElementById

Pour faciliter la rédaction et les explications des exemples de la suite de ce livre, nous allons utiliser une version améliorée de la fonction `getElementById` pour manipuler rapidement et efficacement les éléments DOM et les styles CSS, sans nous obliger à inclure un environnement (*framework*) tel que jQuery.

Cependant, pour éviter tout conflit avec les environnements qui empruntent le caractère `$`, nous utiliserons la lettre `O` en capitale, tout simplement parce que c'est la première lettre du mot *objet*, qui correspond à ce que retourne la fonction quand nous l'appelons, c'est-à-dire l'objet représenté par l'identifiant passé à la fonction.

Fonction O

Voici à quoi ressemble le squelette de la fonction `O` :

```
function O(i)
{
  return document.getElementById(i)
}
```

Cette précaution nous épargne la frappe de 22 caractères chaque fois que nous appelons la fonction. Cependant, nous allons étendre un peu cette fonction pour nous permettre de lui passer soit un nom d'identifiant, soit un objet, comme illustré par l'exemple 20-1, qui reprend la version complète de cette fonction.

Exemple 20-1. La fonction O()

```
function O(i)
{
  return typeof i == 'object' ? i : document.getElementById(i)
}
```

Quand la fonction reçoit un objet en argument, elle le retourne simplement tel quel, tandis que lorsqu'elle reçoit un identifiant, elle retourne l'objet correspondant à cet identifiant.

Mais au fait, pourquoi prendre la peine d'ajouter cette première instruction qui renvoie simplement l'objet passé en argument?

Fonction S

La réponse à cette question apparait lorsque vous examinez une fonction partenaire dénommée `S`, qui offre un accès aisé aux propriétés de `style` (ou CSS) d'un objet, illustrée dans l'exemple 20-2.

Exemple 20-2. La fonction S()

```
function S(i)
{
  return O(i).style
}
```

Le `S` de cette fonction correspond à la première lettre de *style*, et la fonction assure le renvoi de la propriété `style` (ou un sous-objet) de l'élément référencé. Comme la fonction intégrée `O` accepte aussi bien un identifiant qu'un objet, vous pouvez également passer un identifiant ou un objet à `S`.

Voyons à présent ce qui se produit lorsque nous prenons un élément, par exemple `<div>`, d'identifiant `monobj` et que nous réglons à vert la couleur de son texte, comme suit :

```
<div id='monobj'>Du texte</div>

<script>
  O('monobj').style.color = 'green'
</script>
```

Ce code-ci fait tout ce qu'il faut mais comme la fonction `S` existe, nous pouvons abréger le code, comme suit :

```
S('monobj').color = 'green'
```


Maintenant, examinons le cas où l'objet retourné par `O` est affecté à un objet (variable) appelé, par exemple, `diva` :

```
diva = O('monobj')
```

De la manière dont la fonction `S` travaille, nous pouvons encore l'appeler pour changer la couleur du texte en vert, comme suit :

```
S(diva).color = 'green'
```

Autrement dit, que vous souhaitiez accéder directement à l'objet (`diva`) ou indirectement par son identifiant (`'monobj'`), vous avez la possibilité de le faire aussi bien par les fonctions `O` ou `S`, selon les nécessités. Rappelez-vous simplement que, lorsque vous passez un objet, celui-ci ne porte pas d'apostrophes, contrairement à un identifiant.

Fonction C

À ce stade, nous avons donc deux fonctions simples et abrégées pour accéder facilement, l'une à n'importe quel élément d'une page web, et l'autre à toute propriété de `style` d'un élément. Or, il est parfois nécessaire d'accéder à plusieurs éléments à la fois et pour y parvenir, vous pouvez affecter un nom de classe CSS à chacun de ces éléments qui, comme dans les deux exemples suivants, empruntent la classe commune `maclasse` :

```
<div class='maclasse'>Contenu d'une div</fiv>
<p class='maclasse'>Contenu d'un paragraphe</p>
```

Pour ensuite accéder à tous les éléments d'une page qui utilisent une classe déterminée, faites appel à la fonction `C` (comme la première lettre de classe) illustrée dans l'exemple 20-3, pour obtenir un tableau contenant tous les objets qui correspondent au nom de classe fourni.

Exemple 20-3. La fonction C()

```
function C(i)
{
  return document.getElementsByClassname(i)
}
```

L'utilisation de cette fonction se résume ensuite à l'appeler comme suit et à récupérer le tableau renvoyé dans un tableau, pour accéder individuellement aux éléments qu'il contient de manière systématique dans une boucle :

```
montableau = C('maclasse')
```

À partir de là, vous pouvez faire ce que vous voulez des objets renvoyés, comme par exemple les souligner, c'est-à-dire régler à `underline` leur propriété de style `textDecoration`, comme suit :

```
for (i = 0 ; i < montableau.length ; ++i)
  S(montableau[i]).textDecoration = 'underline'
```

Ces instructions itèrent parmi les objets de `montableau[]` puis utilisent la fonction `S` pour faire référence aux propriétés de style de chacun d'eux et définir la propriété `textDecoration` à `underline`.

Inclure les fonctions

À partir d'ici et jusqu'à la fin de ce chapitre, nous utiliserons systématiquement les fonctions `O` et `S`, parce qu'elles abrègent le code et qu'elles sont très simples à suivre. Par conséquent, vous les trouverez dans le fichier *OSC.js* (avec la fonction `C`, à laquelle vous reconnaitrez certainement une utilité) dans le dossier du chapitre 20 des fichiers d'exemples, dans l'archive téléchargeable librement sur le site d'accompagnement du livre.

Pour inclure ces fonctions dans une page web, utilisez l'instruction suivante, de préférence dans la section `<head>` et avant tout script qui fait appel à ces fonctions :

```
<script src='OSC.js'></script>
```

L'exemple 20-4 reprend le contenu du fichier *OSC.js*, où tout est proprement rassemblé sur trois lignes :

Exemple 20-4. Le fichier OSC.js

```
function O(i) { return typeof i == 'object' ? i : document.getElementById(i) }
function S(i) { return O(i).style                                             }
function C(i) { return document.getElementsByClassname(i)                    }
```

Accéder aux propriétés CSS à partir de JavaScript

Si vous avez été attentif, lorsque nous avons fait appel à la propriété `textDecoration` pour souligner le contenu d'éléments de la classe CSS `maclasse`, la propriété CSS correspondante s'écrivait avec un tiret, comme ceci : `text-decoration`. Mais comme JavaScript réserve le tiret à un opérateur mathématique, chaque fois que vous accédez à une propriété CSS avec un tiret, vous devez supprimer ce tiret et placer en capitale le caractère qui le suit immédiatement. Et c'est la même chose s'il y a plusieurs tirets dans la propriété.

La propriété CSS `font-size` est un autre exemple car elle devient `fontSize` en JavaScript lorsqu'elle est placée derrière l'opérateur point, comme suit :

```
monobjet.fontSize = '16pt'
```

Une alternative existe, un peu plus longue à écrire, qui fait appel à la fonction `setAttribute`. Celle-ci prend en charge les noms de propriétés CSS standard :

```
monobjet.setAttribute('style', 'font-size:16pt')
```

Dans ses versions plus anciennes, le navigateur Internet Explorer s'avère un peu pointilleux dans certains cas à propos de l'utilisation des noms de propriétés CSS en JavaScript lors de l'application des versions des règles spécifiques au navigateur, préfixées par `-ms-`. Si vous rencontrez ce problème, n'hésitez pas à emprunter la fonction `setAttribute`, qui devrait le résoudre.

Quelques propriétés usuelles

JavaScript permet de modifier n'importe quelle propriété de tout élément d'un document web, d'une manière semblable à l'utilisation de CSS. Comme vous venez de voir comment accéder aux propriétés CSS tant par la forme abrégée en JavaScript que par la fonction `SetAttribute` pour emprunter les propriétés exactes en CSS pur, nous n'allons pas vous lasser en détaillant à nouveau ces centaines de propriétés. Nous allons plutôt voir comment accéder à quelques-unes des propriétés CSS et examiner certaines choses que vous pouvez réaliser.

Pour commencer, examinons la modification de quelques propriétés CSS à partir de JavaScript avec l'exemple 20-5, qui charge les trois fonctions précitées, crée un élément `<div>` et applique des instructions JavaScript dans la section `<script>` du code HTML, pour modifier quelques-uns de ses attributs (figure 20-1).

Exemple 20-5. Accès à des propriétés CSS avec JavaScript

```
<!DOCTYPE html>
<html>
  <head>
    <title>Accès à des propriétés CSS</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <div id='objet'>Objet div</div>

    <script>
      S('objet').border     = 'solid 1px red'
      S('objet').width      = '100px'
      S('objet').height     = '100px'
      S('objet').background = '#eee'
      S('objet').color      = 'blue'
      S('objet').fontSize   = '15pt'
      S('objet').fontFamily = 'Helvetica'
      S('objet').fontStyle  = 'italic'
    </script>
  </body>
</html>
```

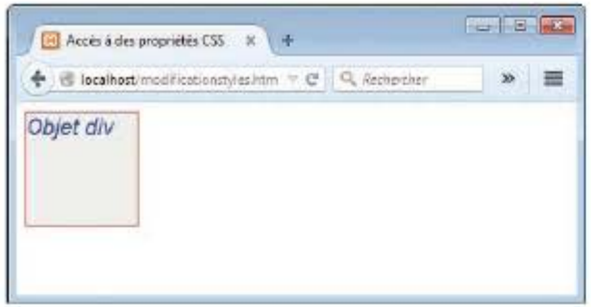


Figure 20-1. Modification de styles avec JavaScript

Vous n'avez aucun intérêt particulier à appliquer des styles de cette façon, alors que vous pouvez les modifier directement en CSS dans le code HTML mais, sous peu, vous devrez modifier des propriétés en réaction à des interactions de l'utilisateur et c'est là que vous percevrez la réelle puissance de la combinaison de JavaScript et de CSS.

Autres propriétés

JavaScript ouvre également la porte à une grande variété d'autres propriétés, telles que la largeur et la hauteur du navigateur, à l'ouverture de fenêtres contextuelles au sein des fenêtres ou des cadres du navigateur, à des informations notamment sur la fenêtre parent (s'il y en a une) et à l'historique des URL de navigation au cours de la session.

Toutes ces propriétés sont accessibles à partir de l'objet `window` par l'entremise de l'opérateur point (par exemple `window.name`). Le tableau 20-1 les énumère, avec leur description.

Tableau 20-1. Propriétés usuelles d'une fenêtre (objet window)

Propriété	Définit et (ou) renvoie
`closed`	Renvoie une valeur booléenne qui indique si une fenêtre a été ou non fermée
`defaultStatus`	Définit ou renvoie le texte par défaut de la barre d'état d'une fenêtre
`document`	Renvoie l'objet `document` correspondant à la fenêtre
`frames`	Renvoie un tableau avec tous les cadres (*frame*) et *iframes* de la fenêtre
`history`	Renvoie l'objet `history` de la fenêtre
`innerHeight`	Définit ou renvoie la hauteur intérieure de la zone de contenu d'une fenêtre
`innerWidth`	Définit ou renvoie la largeur intérieure de la zone de contenu d'une fenêtre
`length`	Renvoie le nombre de cadres et d'*iframes* d'une fenêtre
`location`	Renvoie l'objet `location` (l'adresse URL) de la fenêtre
`name`	Définit ou renvoie le nom d'une fenêtre

Propriété	Définit et (ou) renvoie
`navigator`	Renvoie l'objet `navigator` de la fenêtre
`opener`	Renvoie une référence à la fenêtre qui a créé la fenêtre
`outerHeight`	Définit ou renvoie la hauteur extérieure d'une fenêtre, y compris ses barres d'outils et de défilement
`outerWidth`	Définit ou renvoie la largeur extérieure d'une fenêtre, y compris ses barres d'outils et de défilement
`pageXOffset`	Renvoie le nombre de pixels dont le document a été déplacé dans un défilement, horizontalement à partir du bord gauche de la fenêtre
`pageYOffset`	Renvoie le nombre de pixels dont le document a été déplacé dans un défilement, verticalement à partir du bord haut de la fenêtre
`parent`	Renvoie la fenêtre parent d'une fenêtre
`screen`	Renvoie l'objet `screen` de la fenêtre
`screenLeft`	Renvoie la coordonnée en *x* de la fenêtre relativement à l'écran (*screen*) dans tous les navigateurs récents, sauf Mozilla Firefox (pour lequel vous devez utiliser `screenX`)
`screenTop`	Renvoie la coordonnée en *y* de la fenêtre relativement à l'écran dans tous les navigateurs récents, sauf Mozilla Firefox (pour lequel vous devez utiliser `screenY`)
`screenX`	Renvoie la coordonnée en *x* de la fenêtre relativement à l'écran dans tous les navigateurs récents, sauf Opera, qui renvoie des valeurs incorrectes; non pris en charge par les versions d'IE antérieures à la 9
`screenY`	Renvoie la coordonnée en *y* de la fenêtre relativement à l'écran dans tous les navigateurs récents, sauf Opera, qui renvoie des valeurs incorrectes ; non pris en charge par les versions d'IE antérieures à la 9
`self`	Renvoie la fenêtre courante
`status`	Définit ou renvoie le texte de la barre d'état d'une fenêtre
`top`	Renvoie la fenêtre de niveau le plus haut du navigateur

Quelques remarques méritent d'être précisées à propos de certaines de ces propriétés :

- Les propriétés `defaultStatus` and `status` ne peuvent être modifiées que si l'utilisateur a modifié son navigateur pour le permettre (ce qui est très improbable).
- Il n'est pas possible de lire le contenu de l'objet `history`, donc vous ne pouvez savoir où vos visiteurs ont surfé, mais il autorise l'accès à sa propriété `length` qui détermine la longueur de l'historique, ainsi qu'aux méthodes `back` (page précédente), `forward` (suivante) et `go` (aller à), pour naviguer vers des pages spécifiques de l'historique.
- Lorsque vous devez connaitre la quantité d'espace disponible dans la fenêtre courante du navigateur, lisez les valeurs de `window.innerHeight` et `window.innerWidth`. Celles-ci permettent notamment de calculer le centrage d'une fenêtre contextuelle surgissante (*pop-up window*) d'alerte ou d'une boite de dialogue de demande de confirmation, par rapport à la fenêtre du navigateur.
- L'objet `screen` prend en charge les propriétés en lecture `availHeight`, `availWidth`, `colorDepth`, `height`, `pixelDepth` et `width`, qui permettent de déduire des informations relatives à l'affichage, c'est-à-dire l'écran, dont dispose l'utilisateur.

Nombre de ces propriétés revêtent un intérêt inestimable lorsque vous ciblez des tablettes et des téléphones mobiles, parce qu'elles indiquent exactement la quantité d'espace dont vous disposez pour l'affichage, le type de navigateur utilisé et ainsi de suite.

Ces quelques bribes d'informations vous permettent déjà de débuter et vous offrent de nombreux éléments nouveaux et intéressants, accessibles en JavaScript. En réalité, il existe encore de nombreuses propriétés et méthodes, que nous ne pouvons examiner dans ce chapitre. Cependant, alors que vous savez comment accéder à ces propriétés et les utiliser, tout ce qu'il vous reste à faire est de trouver des ressources qui les énumèrent et les décrivent toutes. En guise de bon point de départ, consultez par exemple (en anglais) *http://www.javascriptkit.com/domref/documentproperties.shtml*.

JavaScript à la volée

Les balises `<script>` ne constituent pas le seul endroit où rédiger des instructions en JavaScript car vous pouvez aussi accéder à du code JavaScript au sein même de balises HTML, ce qui apporte une grande interactivité dynamique. Ainsi, pour ajouter un effet rapide lorsque la souris survole un objet, utilisez du code comme celui dans la balise `<img>` de l'exemple 20-6, qui affiche par défaut une pomme et la remplace par une orange quand la souris la survole (`onmouseover`), pour restaurer la pomme quand la souris s'éloigne (`onmouseout`).

Exemple 20-6. Utilisation de JavaScript à la volée

```
<!DOCTYPE html>
<html>
  <head>
    <title>JavaScript à la volée</title>
  </head>
  <body>
    <img src='apple.png'
      onmouseover="this.src='orange.png'"
      onmouseout="this.src='apple.png'">
  </body>
</html>
```

Mot clé this

L'exemple précédent montre l'utilisation du mot clé `this`. Il indique à JavaScript d'opérer sur l'objet appelant, soit en pratique la balise `<img>`. La figure 20-2 illustre le résultat avant que la souris ne survole la pomme.

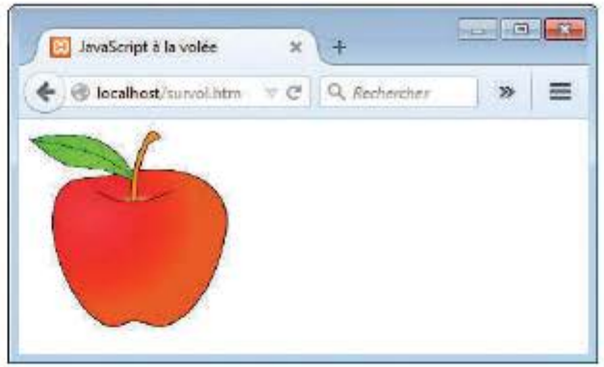


Figure 20-2. Exemple de survol de souris et de code JavaScript à la volée

Lorsqu'il intervient dans un appel JavaScript à la volée, le mot clé `this` représente l'objet appelant. Lorsqu'il apparait dans des méthodes de classe, il représente un objet auquel s'applique la méthode.

Associer des évènements à des objets dans un script

Le code précédent revient à fournir un identifiant à la balise `<img>`, puis à associer des actions aux évènements de souris de la balise, comme dans l'exemple 20-7.

Exemple 20-7. Du JavaScript mais pas à la volée

```
<!DOCTYPE html>
<html>
  <head>
    <title>JavaScript, pas à la volée</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <img id='objet' src='apple.png'>

    <script>
      O('objet').onmouseover = function() { this.src = 'orange.png' }
      O('objet').onmouseout  = function() { this.src = 'apple.png'  }
    </script>
  </body>
</html>
```

Dans la section HTML `<body>`, cet exemple attribue à l'élément `<img>` l'identifiant `objet`, puis traite la manipulation dans une section JavaScript `<script>` indépendante et associe une fonction anonyme à chaque évènement.

Associer d'autres évènements

Que vous utilisiez du JavaScript à la volée ou séparé, vous disposez de nombreux évènements pour les associer à des actions et fournir ainsi une pléthore de fonctionnalités à vos utilisateurs. Le tableau 20-2 énumère ces évènements et le moment de leur déclenchement.

Tableau 20-2. Évènements et leurs moments de déclenchement

Évènement	Se produit
onabort	Lorsque le chargement d'une image en cours est interrompu avant son achèvement
onblur	Quand un élément perd la cible de saisie (*focus*)
onchange	Quand une partie quelconque d'un formulaire a changé
onclick	Quand un objet reçoit un clic de souris
ondblclick	Quand un objet reçoit un double-clic
onerror	Lors de l'apparition d'une erreur JavaScript
onfocus	Quand un élément obtient la cible de saisie
onkeydown	Lors de la pression d'une touche (y compris [Maj.], [Alt], [Ctrl] et [Échap.])
onkeypress	Lors de la pression d'une touche (sauf [Maj.], [Alt], [Ctrl] et [Échap.])
onkeyup	Lors du relâchement d'une touche
onload	Quand le chargement d'un objet est terminé
onmousedown	Lors de la pression du bouton de la souris sur un élément
onmousemove	Lors du déplacement de la souris alors qu'elle survole un élément
onmouseout	Lorsque la souris quitte un élément
onmouseover	Lorsque la souris survole un élément alors qu'elle vient de l'extérieur
onmouseup	Lors du relâchement du bouton de la souris
onsubmit	Lors de la réinitialisation d'un formulaire
onreset	Lors du redimensionnement du navigateur
onresize	Lors d'un défilement dans le document
onscroll	Lors de la sélection de texte
onselect	Lors de l'envoi en soumission d'un formulaire
onunload	Lorsqu'un document est retiré du navigateur

Faites preuve de bon sens lorsque vous associez des évènements à des objets. Par exemple, un objet qui n'est pas un formulaire ne verra jamais l'évènement `onsubmit`.

Ajouter de nouveaux éléments

JavaScript ne vous restreint pas à la seule manipulation d'éléments ou d'objets existants dans le HTML d'un document. Vous pouvez aussi créer des objets et les insérer dans le DOM, et les supprimer.

Ainsi, s'il vous faut un nouvel élément `<div>`, l'exemple 20-8 montre une manière de l'ajouter dans la page web.

Exemple 20-8. Insertion d'un élément dans le DOM

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ajout d'élément</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    Ce document ne comporte initialement que ce texte.<br><br>

    <script>
      alert('Cliquez sur OK pour ajouter un élément')

      newdiv    = document.createElement('div') // Crée l'élément
      newdiv.id = 'NewDiv'
      document.body.appendChild(newdiv)         // Ajoute l'élément enfant au DOM

      S(newdiv).border = 'solid 1px red'        // Définit son style
      S(newdiv).width  = '100px'
      S(newdiv).height = '100px'
      newdiv.innerHTML = "Je suis un nouvel objet inséré dans le DOM."
      tmp              = newdiv.offsetTop       // Force l'actualisation du DOM

      alert('Cliquez sur OK pour supprimer l\'élément')
      pnode = newdiv.parentNode                 // Capture le nœud parent
      pnode.removeChild(newdiv)                 // Supprime l'élément enfant
      tmp = pnode.offsetTop                     // Force l'actualisation du DOM
    </script>
  </body>
</html>
```

La figure 20-3 montre le résultat produit par ce code après avoir ajouté le nouvel élément `<div>` dans le document web. La première étape consiste à créer l'élément avec `createElement`, puis d'appeler la fonction `appendChild` pour insérer l'élément dans le DOM et, enfin, d'imposer son affichage dans le document.

Figure 20-3. Insertion d'un nouvel élément dans le DOM

Ensuite, nous affectons quelques propriétés à l'élément, dont le texte à inclure dans son HTML intérieur (`innerHTML`). Mais, pour être certain que le nouvel élément soit révélé immédiatement, nous lisons sa propriété `offsetTop` et l'affectons à la variable jetable `tmp`. Cette précaution impose une actualisation du DOM et permet l'affichage de l'élément dans tout navigateur qui, sinon, tarderait à le faire, en particulier Internet Explorer.

Ce nouvel élément apparait exactement comme s'il avait été ajouté au code HTML initial et met à disposition les mêmes propriétés et méthodes.

Cette technique qui consiste à créer de nouveaux éléments s'avère utile quelquefois, lorsqu'il s'agit de créer une fenêtre contextuelle surgissante, parce qu'elle n'impose pas de prévoir un élément `<div>` de réserve dans le DOM.

Supprimer des éléments

La suppression d'éléments du DOM est également permise, y compris de ceux qui n'ont pas été ajoutés en JavaScript. La suppression est même plus aisée que l'ajout. Elle fonctionne comme suit, en supposant que l'élément à supprimer soit l'objet `element` :

```
element.parentNode.removeChild(element)
```

Cette instruction accède au nœud (autrement dit à l'objet) parent `parentNode` de l'élément et lui indique de supprimer cet élément de son propre nœud. L'appel à la méthode `removeChild` sur ce dernier exige le passage de l'objet `element` à supprimer. Cependant, pour garantir l'actualisation immédiate du DOM dans tous les navigateurs, il est préférable de remplacer l'instruction précédente par les suivantes (à adapter) :

```
pnode = element.parentNode
pnode.removeChild(element)
tmp   = pnode.offsetTop
```


La première instruction crée une copie d'`element.parentNode` (l'objet parent de l'élément) dans `pnode`, dont, après la suppression de l'élément enfant, nous examinons la propriété `offsetTop` (affectée à la variable jetable `tmp`), pour garantir l'actualisation complète du DOM.

Autres moyens d'ajouter et de supprimer des éléments

L'insertion d'éléments est conçue pour ajouter des objets totalement nouveaux à une page web. Or, si vous ne souhaitez que masquer des objets pour les révéler lors d'un évènement `onmouseover` ou autre, gardez à l'esprit que CSS possède des propriétés disponibles pour ces usages, qui n'exigent pas de mesures aussi drastiques que la création et la suppression d'éléments du DOM.

Ainsi, lorsque vous souhaitez rendre invisible un élément mais le laisser en place (tous les éléments environnants conservent leur emplacement), réglez sa propriété `visibility` à `'hidden'`, comme ceci :

```
monobjet.visibility = 'hidden'
```

Ensuite, pour faire apparaitre l'objet, écrivez ce qui suit :

```
monobjet.visibility = 'visible'
```

Une autre possibilité consiste à replier des éléments, de sorte qu'ils n'occupent plus qu'une largeur zéro et une hauteur zéro (tous les objets environnants occupent l'espace libéré), comme suit :

```
monobjet.display = 'none'
```

Ensuite, pour restaurer ses dimensions initiales à l'élément, écrivez :

```
monobjet.display = 'block'
```

Bien entendu, vous disposez toujours de la propriété `innerHTML`, qui permet de modifier le contenu HTML appliqué à un élément, comme suit :

```
monelement.innerHTML = '<b>Texte HTML de remplacement</b>'
```

Ou, en empruntant la fonction `O` que nous avons définie précédemment, écrivez :

```
O('unid').innerHTML = 'Nouveau contenu'
```

Et dans le même ordre d'idée, pour donner à l'élément l'impression de disparaitre, écrivez :

```
O('unid').innerHTML = ''
```

JavaScript propose encore bien d'autres possibilités d'action sur les propriétés CSS, comme `opacity`, pour régler la visibilité d'un objet entre visibilité et transparence complète, ou `width` et `height`, pour redimensionner l'objet. De plus, la propriété `position` permet, selon ses valeurs `'absolute'`, `'static'` ou `'relative'`, de déplacer un objet n'importe où dans (ou en dehors de) la fenêtre du navigateur, comme vous le souhaitez.

Exploiter les interruptions

JavaScript donne accès à un mécanisme d'*interruptions*, une méthode qui permet de demander au navigateur d'appeler du code déterminé à la fin d'une certaine durée (ou délai d'interruption, *timeout*), et même d'appeler du code à la fin d'un intervalle (*interval*) de temps répété. Ceci offre un moyen de gérer des tâches en coulisses telles que des communications Ajax ou même l'animation d'éléments web.

Pour ce genre d'opérations, vous disposez de deux types d'interruptions : `setTimeout` et `setInterval`. Elles possèdent respectivement les fonctions `clearTimeout` et `clearInterval` pour arrêter ces interruptions.

Utiliser setTimeout

Lors de l'appel de `setTimeout`, passez-lui du code JavaScript ou le nom d'une fonction, ainsi qu'une valeur correspondant à la durée d'attente avant l'exécution du code, comme suit :

```
setTimeout(fairececi, 5000)
```

Dans la définition de la fonction `fairececi`, écrivez par exemple :

```
function fairececi()
{
  alert('Il est temps de vous lever!');
}
```

Attention : Dans ce cas-ci, vous ne pouvez pas directement utiliser la fonction `alert()` avec ses parenthèses en guise de fonction à appeler à l'aide de `setTimeout`, parce qu'elle s'exécuterait immédiatement. Ce n'est que si vous passez un nom de fonction sans ses parenthèses (par exemple, `alert` tout seul) que son code ne s'exécute qu'à l'expiration du délai d'attente.

Passer une chaine

Pour fournir à la fonction d'appel de code un argument et garantir que la fonction ne s'exécute qu'au terme du délai d'attente, donnez à la fonction `setTimeout` une valeur de chaine en plus, comme suit :

```
setTimeout("alert('Bonjour!')", 5000)
```


En réalité, vous pouvez fournir autant de lignes de code JavaScript que vous le souhaitez, à condition d'entourer l'ensemble de guillemets verticaux et de les séparer par des points-virgules, comme ceci :

```
setTimeout("document.write('Démarrage'); alert('Bonjour!')", 5000)
```

Répéter les délais d'interruption

Une technique utilisée par certains programmeurs pour répéter des délais d'interruptions consiste à utiliser la fonction `setTimeout` pour appeler la fonction `setTimeout` dans le code appelé par la première, comme dans ce qui suit, qui amorce une boucle infinie de fenêtres d'alerte :

```
setTimeout(fairececi, 5000)

function fairececi()
{
  setTimeout(dothis, 5000)
  alert('Je m\'ennuie!')
}
```

Ce code provoque l'affichage répété de l'alerte toutes les cinq secondes.

Annuler un délai d'interruption

Pour annuler une interruption en cours, vous devez au préalable mémoriser la valeur renvoyée par l'appel initial à `setTimeout`, comme suit :

```
handle = setTimeout(fairececi, 5000)
```

Le terme *handle*, littéralement « poignée », sert fréquemment dans ce genre de contexte. Il représente une référence de gestion, comme les descripteurs de fichiers évoqués au chapitre 7.

Cette variable connue, vous pouvez annuler l'interruption à tout moment avant l'expiration de son délai avec l'instruction suivante :

```
clearTimeout(handle)
```

Celle-ci provoque l'oubli total de l'interruption et le code associé n'est pas exécuté.

Utiliser setInterval

Au lieu de définir des interruptions répétées avec un appel imbriqué à `setTimeout`, un moyen plus simple de les définir réside dans l'utilisation de la fonction `setInterval`. Elle fonctionne de la même manière, sauf qu'elle déclenche son action à chaque intervalle de temps spécifié en millisecondes, à l'infini tant que vous ne l'annulez pas.

L'exemple 20-9 exploite cette fonction pour afficher une horloge élémentaire dans le navigateur (figure 20-4).

Exemple 20-9. Une horloge gérée par des interruptions

```
<!DOCTYPE html>
<html>
  <head>
    <title>Utilisation de setInterval</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    Il est : <span id='heure'>00:00:00</span><br>

    <script>
      setInterval("afficheheure(O('heure'))", 1000)

      function afficheheure(objet)
      {
        var date = new Date()
        objet.innerHTML = date.toTimeString().substr(0,8)
      }
    </script>
  </body>
</html>
```

Figure 20-4. Affichage de l'heure à l'aide d'interruptions

Lors de chaque appel à `afficheheure`, celle-ci récupère la date et l'heure courantes et les dépose dans l'objet `date` de classe `Date` :

```
var date = new Date()
```

Ensuite, elle règle la propriété `innerHTML` de l'`objet` passé à `afficheheure` avec l'heure courante en heures, minutes et secondes, par l'entremise de l'appel à la fonction `toTimeString`. Celle-ci renvoie une chaine du genre `11:55:47 UTC+0200`, tronquée ensuite aux seuls huit premiers caractères à l'aide de la fonction `substr` :

```
objet.innerHTML = date.toTimeString().substr(0,8)
```

Utiliser la fonction

Pour activer cette fonction `afficheheure`, vous devez d'abord créer un objet en HTML d'identifiant `'heure'`, dont la propriété `innerHTML` permettra d'afficher l'heure :

```
Il est : <span id='heure'>00:00:00</span>
```


Ensuite, dans la section de code `<script>`, placez un appel à la fonction `setInterval`, comme suit :

```
setInterval("afficheheure(O('heure'))", 1000)
```

Cet appel transmet une chaine à `setInterval` avec l'instruction suivante et définit l'intervalle d'interruption à une seconde (toutes les 1 000 millisecondes) :

```
afficheheure(O('heure'))
```

La liaison avec l'élément HTML d'identifiant `'heure'` est assurée et la fonction `afficheheure` reçoit bel et bien l'objet correspondant. Dans le cas rare où l'utilisateur a désactivé JavaScript dans son navigateur (ce que les gens font parfois pour des raisons de sécurité), ce code JavaScript ne s'exécute pas et l'utilisateur ne voit que l'heure initiale, `00:00:00`.

Annuler un intervalle

Pour arrêter une interruption par intervalle lorsque vous la définissez par l'appel à la fonction `setInterval`, vous devez mémoriser le descripteur de l'intervalle, comme suit :

```
handle = setInterval("afficheheure(O('heure'))", 1000)
```

Ensuite seulement, vous pouvez arrêter l'horloge à l'aide d'une instruction telle que la suivante :

```
clearInterval(handle)
```

Vous pouvez même définir une minuterie pour arrêter l'horloge au terme d'un certain délai, comme suit :

```
setTimeout("clearInterval(handle)", 10000)
```

Cette instruction déclenche une interruption après 10 secondes qui efface l'interruption à intervalle répétitif.

Exploiter les interruptions dans les animations

La combinaison de quelques propriétés CSS et d'une interruption répétée permet de produire toutes sortes d'animations et d'effets.

Ainsi, le code de l'exemple 20-10 déplace un carré le long du bord supérieur dans un navigateur, fait gonfler le carré tout le long, comme illustré à la figure 20-5, avant de recommencer lors de la remise à zéro de `GAUCHE`.

Exemple 20-10. Une animation simple

```
<!DOCTYPE html>
<html>
  <head>
    <title>Animation simple</title>
    <script src='OSC.js'></script>
    <style>
      #carre {
```

```
        position  :absolute;
        background:orange;
        border    :1px solid red;
      }
    </style>
  </head>
  <body>
    <div id='carre'></div>

    <script>
      TAILLE = GAUCHE = 0

      setInterval(animer, 30)

      function animer()
      {
        TAILLE += 10
        GAUCHE += 3

        if (TAILLE == 200) TAILLE = 0
        if (GAUCHE == 600) GAUCHE = 0

        S('carre').width  = TAILLE + 'px'
        S('carre').height = TAILLE + 'px'
        S('carre').left   = GAUCHE + 'px'
      }
    </script>
  </body>
</html>
```

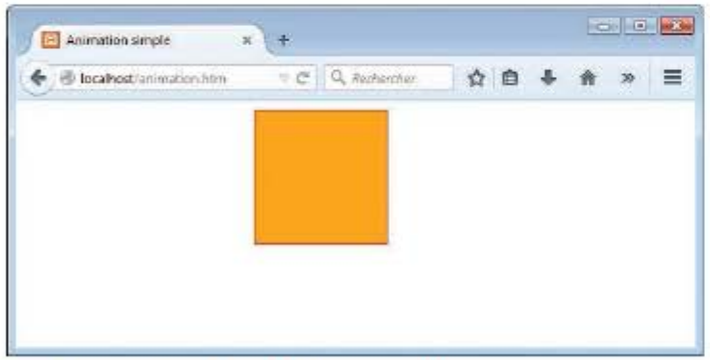


Figure 20-5. L'objet glisse de gauche à droite et grandit

La section `<head>` du document définit l'objet `carre` avec une couleur de fond `orange`, une bordure rouge d'un pixel et la propriété `position` réglée à `absolute` pour qu'il puisse se déplacer dans le navigateur.

Ensuite, dans la fonction `animer`, les variables globales `TAILLE` et `GAUCHE` subissent des mises à jour répétées pour modifier les attributs de style `width`, `height` et `left` de l'objet `carre` (l'ajout de la chaine `'px'` indique l'expression des valeurs en pixels), ce qui génère une animation à une fréquence d'une toutes les 30 millisecondes, ce qui génère un taux de rafraichissement de 33,33 images par seconde (1 000/30 millisecondes).

Questions

1. À quoi servent les fonctions `O`, `S` et `C` définies dans ce chapitre ?
2. Nommez deux manières de modifier un attribut CSS d'un objet.
3. Quelles propriétés fournissent la largeur et la hauteur disponibles dans la fenêtre d'un navigateur ?
4. Comment faire en sorte que quelque chose se produise lorsque la souris passe au-dessus d'un objet et le quitte ?
5. Quelle fonction JavaScript crée de nouveaux éléments et les ajoute au DOM ?
6. Comment (a) rendre un élément invisible et (b) le réduire à des dimensions nulles ?
7. Quelle fonction crée un évènement unique qui se produit à un moment de l'avenir ?
8. Quelle fonction définit des évènements répétitifs à des intervalles réguliers et déterminés à l'avenir ?
9. Comment libérer un élément de son emplacement dans une page web pour lui permettre de se déplacer dans celle-ci ?
10. Quel délai (en millisecondes) entre des évènements devez-vous régler pour réaliser une animation au taux de 50 images par seconde ?

Retrouvez les réponses du chapitre 20 dans l'annexe A.

CHAPITRE 21

Introduction à jQuery

Aussi puissant et souple que soit JavaScript, avec sa pléthore de fonctions intégrées, vous avez souvent besoin de couches supplémentaires de code pour effectuer des choses simples qui ne peuvent être réalisées en natif ou avec CSS, comme des animations, la gestion des évènements et de l'Ajax.

De plus, les différentes guerres entre les navigateurs au fil des ans ont eu, et ont encore, comme conséquences des incompatibilités qui surgissent et disparaissent de temps à autre, pour s'ériger à des moments différents sur différentes plateformes et programmes.

Par conséquent, garantir que vos pages web se ressemblent sur tous les appareils ne peut souvent se faire qu'à coups de code JavaScript fastidieux à mettre en place, pour tenir compte de tous les écarts parmi la gamme des navigateurs et des versions publiées au cours des dernières années. Autrement dit, c'est un véritable cauchemar.

Au moment de l'écriture de ces lignes, en 2014, il semble que le jeu se calme un peu, notamment parce que Microsoft Internet Explorer a embrassé les standards dans de nombreux domaines (mais pas tous), Opera a opté pour se fonder sur WebKit (la même technologie que celle utilisée par Google) pour édifier son navigateur, et Apple a tiré sa révérence sur le marché des navigateurs pour PC.

Quoi qu'il en soit, les anciennes incohérences persistent et plein de gens utilisent encore toujours d'anciens navigateurs. En outre, demeure toujours le souci de devoir écrire beaucoup de code JavaScript lorsqu'il s'agit de créer quoi que ce soit qui aille au-delà des effets spéciaux élémentaires.

Pour combler les manques, un certain nombre de bibliothèques de fonctions ont été développées pour réduire les différences entre les navigateurs, dont nombre parmi elles fournissent aussi des points d'insertion (*hooks*) pour accéder au DOM, le modèle objet de document, ainsi que pour Ajax, la gestion des évènements et des animations. Ces bibliothèques s'appellent *AngularJS*, *jQuery*, *MooTools*, *Prototype*, *script.aculo.us* et *YUI*, mais il y en a bien d'autres encore.

Pourquoi jQuery

Il n'y a de place dans ce livre que pour étudier une seule bibliothèque et nous allons opter pour la plus largement utilisée, jQuery, qui est actuellement installée sur plus de 60 % de tous les sites web, selon *http://w3techs.com*, et (si l'on en croit les graphiques qu'ils ont établis) elle est plus utilisée que tous ses concurrents rassemblés. Par ailleurs, si vous avez la curiosité de voir comment les différentes bibliothèques se positionnent les unes par rapport aux autres, de jour en jour, recherchez *javascript* sur *similartech.com*.

Grâce à jQuery, non seulement vous obtenez un niveau très élevé de compatibilité entre les navigateurs (oui, même avec Internet Explorer), mais vous disposez d'un accès aisé et rapide aux manipulations HTML et du DOM, de fonctions spéciales pour interagir directement avec CSS, de la possibilité de contrôler des évènements, d'outils puissants pour créer des effets et animations professionnels et de fonctions pour piloter des communications Ajax avec le serveur web. jQuery forme également la base d'une vaste gamme de plugiciels et autres utilitaires.

Vous n'êtes cependant pas *obligé* d'utiliser jQuery et certains puristes de la programmation vont jusqu'à refuser de toucher à une bibliothèque, pour préférer créer leurs propres collections de fonctions construites sur mesure. Il peut y avoir de bonnes raisons à cela, notamment d'éviter de devoir attendre des autres des corrections de bogues que vous découvrez, de mettre en œuvre vos propres fonctions personnalisées de sécurité et ainsi de suite. Cependant, jQuery a certainement résisté à l'épreuve du temps et si vous souhaitez profiter de sa courbe d'apprentissage en pente douce pour pouvoir développer des pages web de qualité le plus rapidement possible, ce chapitre explique comment vous pouvez commencer à l'utiliser.

Inclure jQuery

Vous disposez de deux possibilités pour inclure jQuery dans vos pages web. Vous allez sur le site web de jQuery, choisissez la version dont vous avez besoin, la téléchargez sur votre site web et l'utilisez à partir de là. Ou alors, vous tirez parti d'un réseau de diffusion de contenu (RDC, ou *Content delivery network*, *CDN*) gratuit et liez simplement la version dont vous avez besoin.

Du fait que jQuery est diffusé sous les termes de la licence MIT, qui n'impose quasiment aucune restriction sur ce que vous pouvez en faire, vous avez la liberté d'utiliser n'importe quel projet jQuery dans n'importe quel autre projet (même à des fins commerciales), tant que vous conservez intact l'entête de droits d'auteurs.

Choisir la bonne version

Avant de décider de télécharger et d'héberger jQuery directement ou de passer par un RDC, vous devez sélectionner une version de jQuery. Dans la plupart des cas, il est évident que vous opterez pour la dernière version. Cependant, si vous ciblez des navigateurs particuliers ou si vous maintenez un site web hérité, ancien, qui se fonde sur une version déterminée de jQuery, alors il se peut que la dernière version ne soit celle la mieux adaptée à ces cas.

Contrairement à la plupart des logiciels pour lesquels il suffit de télécharger et d'installer la nouvelle version disponible, jQuery a évolué au fil du temps pour tenir compte de la dynamique changeante du marché des différentes versions de navigateurs, avec des caractéristiques différentes et des bogues spécifiques.

Dans le même temps, des améliorations ont été apportées à jQuery qui peuvent entrainer un fonctionnement différent des versions plus récentes sur les sites qui ont été spécialement adaptés à une version particulière (et les bizarreries qui l'entourent), au moment où elle a été diffusée.

Bien entendu, chaque nouvelle version constitue une amélioration par rapport à la précédente, et elle est de plus en plus susceptible de couvrir toutes les bases. Mais lorsqu'un fonctionnement identique est essentiel pour un site web, tant que vous n'avez pas entièrement testé une nouvelle version, il est souvent préférable de s'en tenir à une précédente.

À propos d'Internet Explorer

Par-dessus tout cela, en plus de la série 1.x de jQuery, une série 2.x existe désormais, qui ne prend plus en charge les versions d'Internet Explorer antérieures à la version 9. Les deux séries suivent des développements parallèles. Si vous êtes absolument certain que tous vos utilisateurs disposent d'une version 9 ou supérieure d'Internet Explorer, par exemple parce que vous rédigez une application web mobile, alors n'hésitez pas à préférer la dernière version 2.x pour profiter d'un code plus compact, rapide et efficace. En revanche, si au moins un de vos utilisateurs utilise encore une ancienne version d'IE, alors préférez une version 1.x de jQuery.

Compressé ou modifiable

Vous devez également décider si vous préférez utiliser une version de jQuery minimisée (réduite) en taille pour limiter le trafic nécessaire et améliorer la durée de chargement, ou si vous préférez exploiter la version décompressée (notamment si vous souhaitez y apporter des modifications, ce que vous êtes parfaitement en droit de faire). Généralement, la version minimisée est préférable mais la plupart des serveurs web prennent en charge gzip pour une compression et décompression à la volée, donc cela devient moins capital. Cela dit, la minimisation supprime tous les commentaires de code.

Téléchargement

Chaque version de jQuery est proposée sous les deux formes, compressée ou minimisée, sur *jquery.com/download/*.

Tout ce qu'il vous reste à faire, c'est de choisir la version dont vous avez besoin, de cliquer du bouton droit sur le lien qui figure à côté et de l'enregistrer sur votre disque dur. À partir de là, déposez-le sur votre serveur web et incluez-le entre des balises `<script>`, comme suit (pour la forme minimisée de la version 1.11.1) :

```
<script src='http://nonserveur.com/jquery-1.11.2.min.js'></script>
```

Si vous n'avez jamais utilisé jQuery auparavant (et n'avez pas vraiment d'exigences précises), alors téléchargez la dernière version minimisée ou créez un lien vers elle à partir d'un RDC, comme décrit ci-dessous.

Utiliser un réseau de diffusion de contenu

Plusieurs réseaux de diffusion de contenu prennent en charge jQuery. Si vous empruntez l'un d'eux, vous vous épargnez les tracas de devoir télécharger de nouvelles versions, pour ensuite les téléverser sur votre serveur. Il vous suffit en effet de lier directement l'URL fournie par ces réseaux.

De plus, ces réseaux proposent ce service gratuitement, alors qu'ils disposent de dorsales de très grande capacité qui figurent parmi les plus rapides au monde. Les RDC détiennent leurs contenus dans un nombre de sites géographiques différents, pour fournir un fichier de leur serveur le plus près possible du surfeur, ce qui garantit la diffusion et la livraison les plus rapides possibles.

Globalement, si vous ne devez pas modifier le code source de jQuery (ce qui vous obligerait alors à l'héberger sur vos propres serveurs web) et si vos utilisateurs sont certains de disposer d'une connexion internet permanente, la solution du RDC est certainement la plus appropriée. Et elle est simple. Tout ce que vous devez connaître, c'est le nom du fichier que vous ciblez et le dossier racine qu'utilise le RDC. Ainsi, toutes les versions courantes et précédentes sont accessibles par l'entremise du RDC qu'utilise jQuery, comme suit :

```
<script src='http://code.jquery.com/jquery-1.11.2.min.js'></script>
```

Le dossier de base est *http://code.jquery.com/* et il suffit de le faire suivre sur le nom du fichier à inclure, soit *jquery-1.11.2.min.js* ici.

Tant Microsoft que Google proposent jQuery sur leurs propres réseaux donc vous pouvez l'inclure d'une des manières suivantes :

```
<script src='http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.11.2.min.js'></script>
<script src='http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js'>
</script>
```

Dans le cas du RDC de Microsoft (aspnetcdn.com), commencez l'URL par le dossier de base *ajax.aspnetcdn.com/ajax/jQuery/*, puis ajoutez le nom du fichier ciblé.

En revanche, pour Google, vous devez scinder le nom du fichier, par exemple *jquery-1.11.2.min.js*, en un dossier et un nom de fichier, comme ceci : *1.11.2/jquery.min.js*. Faites précéder ceci de *ajax.googleapis.com/ajax/libs/jquery/*.

Un autre avantage de l'utilisation d'un RDC se situe dans le fait que, comme la plupart des autres sites web le font aussi, il est fort probable que jQuery ait déjà été chargé dans la mémoire cache du navigateur de l'utilisateur et qu'il ne nécessite donc plus de transit par l'internet. Avec les quelque 60 % et plus de sites qui utilisent jQuery, ceci peut s'avérer un gain de trafic et de temps.

Utiliser toujours la dernière version

Un autre avantage des RDC est qu'ils offrent la possibilité de toujours utiliser la dernière version de jQuery, sans que n'ayez à vous en soucier : vous enregistrez votre page web et la version liée sera toujours la dernière diffusée.

Pour inclure la dernière version (de la série 1.x) du RDC de jQuery ou de Google, utilisez une des formes suivantes de la balise `<script>` :

```
<script src='http://code.jquery.com/jquery-latest.min.js'></script>
<script src='http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js'></script>
```

Soyez toutefois très prudent avec cette solution car, si quelque chose commence à présenter un comportement bizarre ou si vous constatez de la casse dans votre site, soyez conscient que le problème peut venir d'une mise à niveau inattendue de jQuery.

Cela vaut la peine de savoir que les RDC de jQuery et Google prennent en charge les protocoles tant HTTP que HTTPS, accessibles par les préfixes `http://` et `https://`. Dans les exemples de ce chapitre, disponibles au téléchargement sur le site d'accompagnement du livre, j'ai cependant opté pour le téléchargement et la diffusion de jQuery en local, pour que vous puissiez tester tous les exemples, même si vous êtes hors ligne, dans un train, une auto ou un avion, par exemple.

Personnaliser jQuery

Si vous vous trouvez dans une situation qui impose de réduire au strict minimum la quantité de données téléchargées par une page web, alors vous pouvez toujours utiliser jQuery à condition d'en créer une version spéciale qui ne contienne que les fonctionnalités requises par votre site web. Vous ne pouvez plus vous reposer sur un RDC pour la fournir mais, dans de telles circonstances, vous n'envisagiez probablement déjà pas d'y faire appel.

Pour créer votre propre version de jQuery, consultez *projects.jga.me/jquery-builder*, cochez les cases des options que vous voulez et décochez celles que vous ne souhaitez pas. Ladite version de jQuery se charge en fin d'opération dans un onglet ou une fenêtre, dont il vous suffit de copier et coller le code.

Syntaxe de jQuery

La principale pierre d'achoppement pour les gens qui découvrent jQuery se situe au niveau du symbole `$`, qui agit en tant que méthode de fabrication (*factory method*) de jQuery.

Il a été choisi ainsi parce qu'il est autorisé en JavaScript, qu'il est court et qu'il se distingue de tous les noms des variables, objets, fonctions et méthodes.

Il prend la place d'un appel à la fonction `jQuery` (tout aussi permis, d'ailleurs, si vous préférez). L'idée est de conserver un code court et fluide, et d'éviter les frappes excessives au clavier chaque fois que vous accédez à jQuery. Il indique aussi immédiatement aux autres développeurs qui découvrent votre code que vous utilisez jQuery (ou une bibliothèque semblable).

Exemple simple

Dans la situation la plus simple, pour accéder à jQuery, tapez le symbole `$`, suivi d'un sélecteur entre parenthèses, puis d'un point et d'une méthode à appliquer à l'élément ou aux éléments sélectionnés.

Par exemple, pour modifier la famille de police de tous les paragraphes en `monospace`, écrivez l'instruction suivante :

```
$('p').css('font-family', 'monospace')
```

Ou, pour ajouter une bordure à l'élément `<code>`, écrivez ceci :

```
$('code').css('border', '1px solid #aaa')
```

Examinons cela dans un exemple plus complet, comme l'exemple 21-1, où la partie jQuery est mise en évidence en gras.

Exemple 21-1. Exemple simple de jQuery

```
<!DOCTYPE html>
<html>
  <head>
    <title>Premier exemple de jQuery</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    La bibliothèque jQuery utilise les noms de fonctions soit <code>$()</code>,
      soit <code>jQuery()</code>.
    <script>
      $('code').css('border', '1px solid #aaa')
    </script>
  </body>
</html>
```

Au chargement de cet exemple dans un navigateur, le résultat prend l'apparence illustrée par la figure 20-1. Bien entendu, cette instruction déterminée imite ce que vous pouvez réaliser à l'aide de CSS normal mais l'idée consiste à illustrer la syntaxe de jQuery et, pour l'instant, nous ne voyons que des choses simples.

L'autre manière d'envoyer cette commande fait appel à la fonction `jQuery`, qui fonctionne de la même façon que `$`, comme suit :

```
jQuery('code').css('border', '1px solid #aaa')
```

Figure 21-1. Modification d'éléments avec jQuery

Éviter les conflits entre bibliothèques

Si vous utilisez d'autres bibliothèques aux côtés de jQuery, vous risquez de constater qu'elles définissent leur propre fonction `$`. Pour résoudre ce conflit, appelez la méthode `noConflict` sur le symbole, qui libère le contrôle pour que l'autre bibliothèque puisse prendre la main, comme suit :

```
$.noConflict()
```

Ceci fait, pour accéder de nouveau à jQuery par la suite, vous devez appeler la fonction `jQuery`. Vous pouvez aussi remplacer l'utilisation du symbole `$` par un nom d'objet de votre choix, comme ceci :

```
jq = $.noConflict()
```

À partir de là, vous pouvez utiliser `jq`, partout où vous auriez utilisé précédemment `$`.

Pour distinguer les objets de jQuery et les suivre séparément des objets d'éléments standards, certains développeurs préfixent d'un `$` tout nom d'objet créé avec jQuery (de sorte qu'ils finissent par ressembler à des variables de PHP).

Sélecteurs

Maintenant que vous avez vu la simplicité de l'inclusion de jQuery dans une page web et l'accès à ses fonctionnalités, poursuivons avec l'examen de ses sélecteurs qui, vous apprécierez ce fait, fonctionnent exactement de la même manière qu'en CSS. En fait, c'est là le fondement même du fonctionnement de jQuery.

Tout ce que vous devez faire est de réfléchir à la manière dont vous voulez appliquer du style à un ou plusieurs éléments à l'aide de CSS, puis vous pouvez utiliser les mêmes sélecteurs pour appliquer des opérations jQuery aux éléments sélectionnés.

Méthode css

Pour expliquer l'usage des sélecteurs en jQuery, examinons une des méthodes les plus fondamentales de jQuery : `css`. Celle-ci permet de modifier dynamiquement toute propriété CSS. Elle prend deux arguments : le nom de la propriété à laquelle accéder et une valeur à lui appliquer, comme suit :

```
css('font-family', 'Arial')
```

Ainsi que vous le verrez dans les prochaines sections, il n'est pas permis d'utiliser uniquement cette méthode car vous devez l'ajouter à un sélecteur de jQuery, qui permet de sélectionner un ou plusieurs éléments dont la méthode doit modifier les propriétés. Ce qui suit en est un exemple, qui impose la justification complète de tous les éléments `<p>` :

```
$('p').css('text-align', 'justify')
```

La méthode `css` permet aussi de renvoyer (au lieu de définir) une valeur calculée, lorsque vous ne lui donnez comme argument qu'un nom de propriété, sans le second argument. Dans ce cas, elle renvoie la valeur de la propriété du premier élément qui correspond au sélecteur. Ainsi, l'instruction suivante renvoie la couleur du texte de l'élément d'identifiant `elem`, sous la forme d'une méthode `rgb` :

```
couleur = $('#elem').css('color')
```

Retenez que la valeur renvoyée est la valeur calculée. Autrement dit, jQuery calcule et renvoie la valeur utilisée par le navigateur au moment de l'appel de la méthode, et non la valeur originale affectée éventuellement à la propriété par une feuille de style ou tout autre moyen.

Par exemple, si la couleur du texte est bleue, la valeur affectée à la variable couleur dans l'instruction précédente est `rgb(0, 0, 255)`, même si la couleur a été initialement réglée au nom de couleur `blue`, ou aux chaines hexadécimales `#ff` ou `#0000ff`. Cette valeur calculée est toutefois toujours sous une forme que vous pouvez réaffecter par la suite à l'élément (ou tout autre élément), en guise de second argument de la méthode `css`.

Soyez attentif à toute dimension renvoyée par cette méthode parce que, selon le réglage de `box-sizing` (voir chapitre 19), elle peut ne pas être ce que vous attendez. Lorsque vous devez obtenir ou définir des largeurs et des hauteurs sans tenir compte de `box-sizing`, préférez les méthodes `width` et `height` (et leurs consœurs), comme décrit à la section «Modifier des dimensions », à la page 535.

Sélecteur d'élément

Pour sélectionner un élément à manipuler par jQuery, indiquez son nom entre les parenthèses qui suivent le symbole `$` (ou le nom de fonction `jQuery`). Ainsi, pour changer la couleur de fond de tous les éléments `<blockquote>` (citation en bloc), utilisez une instruction telle que la suivante :

```
$('blockquote').css('background', 'lime')
```


Sélecteur d'identifiant

Vous pouvez aussi faire référence à des éléments par leurs identifiants, en plaçant un caractère `#` devant le nom d'identifiant. Ainsi, pour ajouter une bordure à l'élément d'identifiant `publicite`, par exemple, utilisez ceci :

```
$('#publicite').css('border', '3px dashed red')
```

Sélecteur de classe

Et vous pouvez manipuler des groupes d'élément en fonction de la classe qu'ils utilisent. Par exemple, pour souligner tous les éléments qui utilisent la classe `nouveau`, écrivez :

```
$('.nouveau').css('text-decoration', 'underline')
```

Combiner les sélecteurs

Comme en CSS, vous pouvez combiner les sélecteurs dans une seule sélection jQuery à l'aide de virgules, comme suit :

```
$('blockquote, #publicite, .nouveau').css('font-weight', 'bold')
```

L'exemple 21-2 rassemble tous ces types de sélecteurs en un seul code (avec les instructions jQuery surlignées en gras), dont la figure 21-2 illustre le résultat.

Exemple 21-2. Utilisation de différents sélecteurs avec jQuery

```
<!DOCTYPE html>
<html>
  <head>
    <title>Deuxième exemple de jQuery</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <blockquote>Aussi puissant et souple que soit JavaScript, avec sa pléthore
    de fonctions intégrées, vous avez souvent besoin de couches supplémentaires
    de code pour effectuer des choses simples qui ne peuvent être réalisées en
    natif ou avec CSS, comme des animations, la gestion des évènements et
    de l'Ajax.</blockquote>
    <div id='publicite'>Ceci est une publicité</div>
    <p>Ceci est mon <span class='nouveau'>nouveau</span> site web</p>
    <script>
      $('blockquote').css('background', 'lime')
      $('#publicite').css('border', '3px dashed red')
      $('.nouveau').css('text-decoration', 'underline')
      $('blockquote, #publicite, .nouveau').css('font-weight', 'bold')
    </script>
  </body>
</html>
```

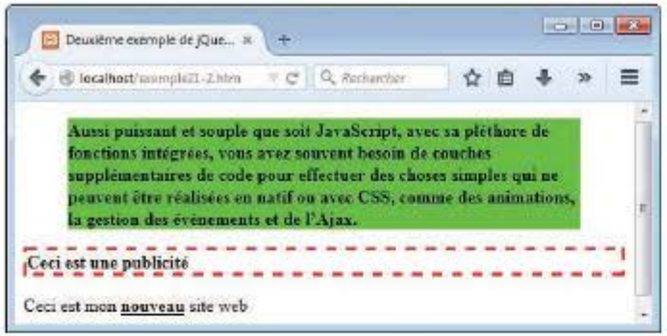


Figure 21-2. Manipulation de plusieurs éléments

Gérer les évènements

Si jQuery ne pouvait servir qu'à modifier des styles CSS, il ne serait pas d'une grande utilisé mais, bien tendu, il permet bien plus que cela. Voyons par exemple comment il gère les évènements.

Rappelez-vous que la plupart des évènements sont déclenchés par les interactions avec l'utilisateur: lorsque la souris survole un élément, que l'utilisateur clique avec le bouton de sa souris ou qu'il presse une touche. Mais il est possible aussi de déclencher d'autres évènements, comme lorsqu'un document a terminé son chargement.

Avec jQuery, il est simple d'associer votre code à ces évènements de manière sécuritaire, qui n'empêche pas d'autres programmes d'accéder aussi à ces évènements. Par exemple, voici comment indiquer à jQuery de répondre au clic sur un élément:

```
$('#cliquezsurmoi').click(function()
{
  $('#resultat').html('Vous avez cliqué avec la souris!')
})
```

Lorsque l'élément d'identifiant `cliquezsurmoi` reçoit un clic de souris, la propriété `innerHTML` de l'élément d'identifiant `resultat` est mise à jour à l'aide de la fonction jQuery `html`.

Les objets jQuery (créés tant avec les méthodes `$` que `jQuery`) ne sont pas les mêmes que les objets créés en JavaScript à l'aide de `getElementById`. En pur JavaScript, vous pouvez écrire une instruction telle que `objet = document.getElementById('resultat')`, suivie de (par exemple) `objet.innerHTML = 'quelquechose'`. Mais, dans l'exemple précédent, `$('#result').innerHTML` ne fonctionne pas, car innerHTML ne fait pas partie des propriétés d'un objet jQuery. D'où l'utilisation de la méthode jQuery `html` pour obtenir le résultat attendu.

L'exemple 21-3 concrétise le concept et la figure 21-3 illustre l'affichage obtenu.

Exemple 21-3. Traitement d'un évènement

```
<!DOCTYPE html>
<html>
  <head>
    <title>Évènements en jQuery</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <button id='cliquezsurmoi'>Cliquez sur moi</button>
    <p id='resultat'>Je suis un paragraphe</p>
    <script>
      $('#cliquezsurmoi').click(function()
      {
        $('#resultat').html('Vous avez cliqué sur le bouton!')
      })
    </script>
  </body>
</html>
```

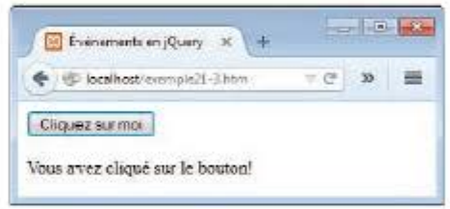


Figure 21-3. Traitement d'un évènement de clic

Lorsque vous accédez à un évènement à l'aide de jQuery, oubliez le préfixe `on` qu'utilise JavaScript en standard. Ainsi, le nom d'évènement JavaScript `onmouseover` devient la fonction `mouseover` en jQuery, `onclick` devient `click` et ainsi de suite.

Attendre que le document soit prêt

Comme jQuery est si intimement associé au DOM dans ce qu'il permet de réaliser, vous devrez très souvent attendre qu'une page web soit chargée avant de commencer à en manipuler les éléments. Sans jQuery, le chargement de la page est répercuté dans l'évènement `onload` mais jQuery offre une méthode plus efficace et valable pour tous les navigateurs, appelée `ready`, qui permet d'intervenir le plus tôt possible, même avant `onload`. Cela signifie que jQuery peut commencer à agir sur une page très rapidement, pour réduire au minimum des délais peu conviviaux.

Pour exploiter cette fonctionnalité, placez votre code jQuery au sein de la structure suivante :

```
$('document').ready(function()
{
  // Placez votre code ici.
})
```

Ainsi, le code attend que le document soit prêt et ce n'est qu'alors que la méthode `ready` l'appelle. En pratique, il existe une version abrégée qui évite des excès de frappe au clavier, illustrée dans l'exemple 21-4.

Exemple 21-4. La fonction d'encapsulation de code « ready » la plus courte en jQuery

```
$(function()
{
  // Placez votre code ici.
})
```

Si vous prenez la bonne habitude d'encapsuler vos instructions jQuery dans une de ces deux structures, vous ne risquez pas de devoir faire face aux types d'erreurs générés par des tentatives trop précoces d'accès au DOM.

Une autre approche consiste à placer tout le code JavaScript à la fin de chaque page HTML, pour qu'il ne puisse s'exécuter qu'à la fin du chargement de la totalité du document. Cette approche offre l'avantage également de garantir que le contenu de la page web reçoive la priorité au chargement, dont vous mesurerez l'amélioration au niveau de la qualité d'expérience de l'utilisateur.

Le seul cas où le fait de placer les scripts en fin de page présente des failles se situe lorsqu'un document semble prêt alors qu'il ne l'est pas en réalité ou quand ses feuilles de styles externes ne sont pas encore chargées (ce que vous ne pourrez vérifier que par des tests). L'utilisateur pense qu'il peut commencer à interagir avec la page alors que le script n'est pas prêt. Dans ce genre de cas, implantez la fonction `ready` et tout se passera bien. En pratique, dans le doute, placez vos scripts à la fin de la page et utilisez la fonction `ready`, et vous bénéficierez des deux avantages.

Fonctions et propriétés d'évènements

Si vous venez de voir la méthode d'évènement `ready`, sachez que jQuery met à votre disposition des dizaines de méthodes d'évènements et de propriétés associées (beaucoup trop pour les détailler toutes ici). Les suivantes sont les plus utilisées et elles vous accompagneront dans la plupart de vos projets. Pour un résumé de tous les évènements disponibles, consultez *api.jquery.com/category/events*.

Évènements blur et focus

Le déclenchement de l'évènement `blur` a lieu lorsqu'un élément perd la cible de saisie (*focus*), ce qui provoque son estompage (traduction littérale de *blur*), ce qui en fait le partenaire privilégié de l'évènement `focus`. Tous deux servent à ajouter un gestionnaire de l'évènement, ou à déclencher l'évènement si vous oubliez un des arguments attendu entre les parenthèses de la méthode.

L'exemple 21-5 comporte quatre champs de saisie. Deux gestionnaires sont ajoutés à tous les éléments `input`. Le gestionnaire `focus` règle leur fond à jaune lorsqu'ils reçoivent le focus, tandis que le gestionnaire `blur` règle à gris leur fond lorsqu'ils perdent le focus (et donc s'estompent). Ensuite, le premier champ reçoit le focus à l'ouverture par un appel à la méthode `focus`, pour l'appliquer à l'élément d'identifiant `premier`.

Exemple 21-5. Exploitation des évènements focus et blur

```
<!DOCTYPE html>
<html>
  <head>
    <title>Évènements blur et focus</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <h2>Cliquez dans et hors de ces champs</h2>
    <input id='premier'> <input> <input> <input>
    <script>
      $('input').focus(function() { $(this).css('background', '#ff0') } )
      $('input') .blur(function() { $(this).css('background', '#aaa') } )
      $('#premier').focus()
    </script>
  </body>
</html>
```

Vous avez le droit de placer des espaces entre les parenthèses d'une méthode et l'opérateur point utilisé pour lui associer une autre méthode (et même après le point, si vous préférez), comme dans l'exemple précédent qui justifie à droite les noms d'évènements `focus` et `blur`, pour afficher la suite des instructions en colonnes.

La figure 25-4 montre que le code donne une couleur d'arrière-plan gris à tous les champs qui ont reçu puis perdu le focus. Si l'un d'eux a le focus, sa couleur de fond devient jaune, tandis que les champs non visités conservent la couleur de fond blanche.

Figure 21-4. Gestion des évènements blur et focus

Mot clé this

Cet exemple permet également d'illustrer l'utilisation du mot clé `this`. Lors de l'appel d'un évènement, l'élément au départ duquel il a été déclenché est transmis à l'objet `this`, que l'on peut ensuite passer à la méthode `$` pour un traitement. Ou, comme `this` est un objet standard de JavaScript (et non un objet jQuery), il peut servir en tant que tel. Donc, si vous préférez, vous pouvez remplacer ceci (jQuery) :

```
$(this).css('background', '#ff0')
par cela (JavaScript) :
this.style.background = '#ff0'
```

Évènements click et dblclick

Tout au début de la section Gérer les évènements, nous avons vu l'évènement `click` mais il existe aussi un évènement pour gérer les doubles clics. Pour les utiliser, associez la méthode de l'évènement à une sélection jQuery et en argument, placez une méthode jQuery à appeler lors du déclenchement de l'évènement, comme suit :

```
$('.maclasse')    .click( function() { $(this).slideUp() })
$('.maclasse').dblclick( function() { $(this).hide()    })
```

Cette solution emprunte des fonctions anonymes, mais vous pouvez utiliser des fonctions nommées si vous préférez. Rappelez-vous toutefois de fournir les noms de ces fonctions sans parenthèses pour éviter qu'elles ne soient appelées à un moment inapproprié. L'objet `this` est automatiquement transmis et mis à la disposition de la fonction nommée, comme ceci :

```
$('.maclasse').click(enrouler)

function enrouler()
{
  $(this).slideUp()
}
```

Les détails des méthodes `slideUp` et `hide` feront l'objet de la section Effets spéciaux, page 521. Pour l'instant, exécutez l'exemple 21-6 et cliquez ou double-cliquez sur les boutons pour en voir certains disparaitre avec une animation (à l'aide de `slideUp`) ou d'autres simplement s'évanouir à l'écran (à l'aide de hide), comme illustré à la figure 21-5.

Exemple 21-6. Association des évènements click et dblclick

```
<!DOCTYPE html>
<html>
  <head>
    <title>Évènenents click et dblclick</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <h2>Cliquez et double-cliquez sur les boutons</h2>
    <button class='maclasse'>Button 1</button>
    <button class='maclasse'>Button 2</button>
    <button class='maclasse'>Button 3</button>
    <button class='maclasse'>Button 4</button>
    <button class='maclasse'>Button 5</button>
    <script>
      $('.maclasse')    .click( function() { $(this).slideUp() })
      $('.maclasse').dblclick( function() { $(this).hide()    })
    </script>
  </body>
</html>
```

Figure 21-5. Un clic sur le bouton 3 le fait s'enrouler vers le haut

Évènement keypress

De temps à autre, il s'avère nécessaire de disposer d'une interaction plus approfondie avec le clavier, notamment lors de la gestion de formulaires complexes ou encore l'écriture de jeux. Dans de telles situations, utilisez la méthode `keypress`, qui s'associe à n'importe quel élément acceptant des entrées au clavier, comme un champ d'entrée (`<input>`) ou le document lui-même.

Dans l'exemple 21-7, la méthode est associée au document pour intercepter toutes les pressions de touches (figure 21-6).

Exemple 21-7. Interception de pressions de touches

```
<!DOCTYPE html>
<html>
  <head>
    <title>Évènement keypress</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <h2>Appuyez sur une touche du clavier</h2>
    <div id='resultat'></div>
    <script>
      $(document).keypress(function(event)
      {
        touche = String.fromCharCode(event.which)

        if (touche >= 'a' && touche <= 'z' ||
            touche >= 'A' && touche <= 'Z' ||
            touche >= '0' && touche <= '9')
        {
          $('#resultat').html('Vous avez appuyé sur : ' + touche)
          event.preventDefault()
        }
      })
    </script>
  </body>
</html>
```

Figure 21-6. Gestion des pressions de touches au clavier

Cet exemple comporte quelques petites choses que vous devez conserver à l'esprit lorsque vous rédigez vos propres gestionnaires de touches de clavier. Par exemple, comme les navigateurs renvoient des valeurs divergentes pour cet évènement, la propriété `which` (laquelle) de l'objet `event` (évènement) est normalisée par jQuery pour renvoyer toujours le même caractère, quel que soit le navigateur. Donc c'est cette valeur que vous devez inspecter pour connaitre la touche pressée.

Ensuite, comme la valeur renvoyée par `which` est un code de caractère, donc un nombre, passez-le à `String.fronCharCode` pour le convertir en une chaine d'une seule lettre. Si vous êtes en mesure de traiter directement les codes ASCII, il n'est pas nécessaire de passer par là mais, au moins, vous connaissez désormais cette méthode si vous devez manipuler des caractères.

Dès qu'une pression de touche est détectée, une simple instruction permet d'insérer un texte adéquat dans la propriété `innerHTML` de l'élément `div` de l'identifiant `resultat`.

Ceci est un bon exemple où il ne faut pas utiliser la fonction `document.write`, parce que le document a déjà été chargé au moment de la pression d'une touche par l'utilisateur. Si nous avions utilisé `document.write` pour afficher l'information à ce moment, elle aurait écrasé et effacé le document courant. Par conséquent, dans ce cas-ci, l'écriture dans la partie HTML d'un élément constitue le moyen parfait, non destructif, de fournir à l'utilisateur un retour d'information, comme expliqué au chapitre 13, dans la section « À propos de document.write », page 328.

Programmation respectueuse

Lorsque vous anticipez des entrées de l'utilisateur, décidez des valeurs auxquelles vous souhaitez répondre et réagir, mais ignorez toutes les autres au cas où un gestionnaire d'évènement aurait besoin d'y accéder. Ceci constitue une pratique respectueuse des autres utilitaires (et du navigateur principal lui-même) qui peuvent s'exécuter en parallèle. Ainsi, dans l'exemple précédent, nous n'acceptons que les caractères alphanumériques a à z, A à Z et 0 à 9, pour ignorer tous les autres.

Vous avez deux possibilités pour transférer des interruptions du clavier (ou les intercepter pour les retenir) aux autres gestionnaires. La première consiste à ne rien faire du tout et, quand votre code termine son exécution, les autres gestionnaires peuvent voir de nouveau les pressions de touches pour y réagir. Ceci entraine cependant de la confusion si une même pression de touche correspond à plusieurs actions différentes.

La seconde, si vous préférez éviter que l'évènement ne déclenche d'autres gestionnaires, consiste à appeler la méthode `preventDefault` d'`event`, qui permet d'empêcher l'évènement de remonter, tel une bulle à la surface de l'eau, vers d'autres gestionnaires.

Soyez prudent lorsque vous placez votre appel à `preventDefault`, parce que s'il apparait en dehors de la portion de code où vous gérez les pressions de touches, alors il empêche tous les autres évènements de clavier de remonter et vous risquez ainsi de verrouiller l'utilisateur hors du navigateur (ou en tout cas de l'empêcher d'utiliser certaines fonctionnalités).

Évènement mousemove

Parmi les évènements les plus communément interceptés figurent ceux liés à la souris. Nous avons déjà vu les clics sur le bouton de la souris, donc nous examinons ici les évènements de déplacement (move) de la souris.

C'est l'occasion de nous pencher sur un exemple de code un peu plus intéressant. L'exemple 21-8 rassemble un programme de dessin rudimentaire en jQuery et un canevas HTML5. Nous ne verrons le canevas (canvas) en détail qu'au chapitre 23, mais ne vous inquiétez pas car le code demeure assez simple.

Exemple 21-8. Interception des évènements de mouvements et de bouton de la souris

```
<!DOCTYPE html>
<html>
  <head>
    <title>Évènements de gestion de la souris</title>
    <script src='jquery-1.11.2.min.js'></script>
    <style>
      #pad {                                          /* zone de dessin */
        background:#def;
        border    :1px solid #aaa;
      }
    </style>
  </head>
  <body>
    <canvas id='pad' width='480' height='320'></canvas>
    <script>
      canvas  = $('#pad')[0]                          // Canevas
      context = canvas.getContext("2d")               // Contexte
      pendown = false                                 // Stylo abaissé?

      $('#pad').mousemove(function(event)
      {
        var xpos = event.pageX - canvas.offsetLeft
        var ypos = event.pageY - canvas.offsetTop

        if (pendown) context.lineTo(xpos, ypos)
        else         context.moveTo(xpos, ypos)

        context.stroke()
      })

      $('#pad').mousedown(function() { pendown = true  } )
      $('#pad')  .mouseup(function() { pendown = false } )
    </script>
  </body>
</html>
```

La figure 21-7 montre que cet ensemble très simple d'instructions permet de créer des dessins à base de traits, dont la qualité dépend de vos talents artistiques. Voyons comment le tout fonctionne. D'abord, nous créons un objet `canvas` par référence au premier élément (d'indice zéro) du sélecteur jQuery, comme suit :

```
canvas = $('#pad')[0]
```

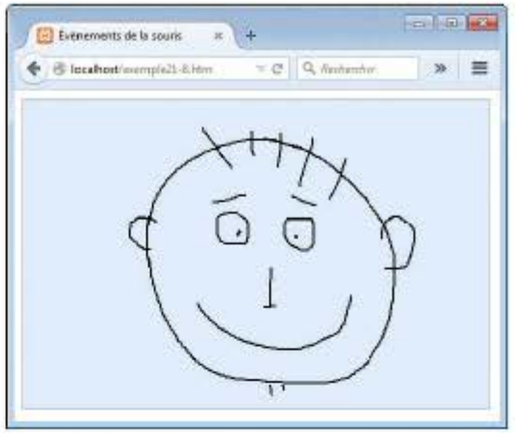



Figure 21-7. Interception des évènements de mouvements et de bouton de la souris

Ceci représente une manière de prendre rapidement un objet de jQuery et de l'extraire comme un objet d'élément standard de JavaScript. Une autre manière existe, qui emprunte la méthode `get`, comme suit :

```
canvas = $('#pad').get(0)
```

Ces deux manières d'écrire sont interchangeables mais celle de `get` présente l'avantage de, sans argument, renvoyer tous les objets de nœuds d'éléments d'un objet jQuery sous forme d'un tableau.

Quoi qu'il en soit, comme l'explique le chapitre 23, pour pouvoir écrire dans le canevas, celui-ci nécessite un objet spécial `context`, créé comme suit :

```
context = canvas.getContext("2d")
```

Il reste une chose à initialiser : une variable booléenne nommée `pendown` qui détermine si le stylo est abaissé sur la zone de dessin, pour suivre l'état pressé ou non du bouton de la souris (initialisée à `false` parce que le stylo est relevé) :

```
pendown = false
```

Ensuite, l'évènement `mousemove` du canevas est intercepté par la fonction anonyme suivante, au sein de laquelle trois séries de choses ont lieu :

```
$('#pad').mousemove(function(event)
{
  ...
})
```

D'abord, les variables locales `xpos` et `ypos` (locales de par les mots-clés `var`) reçoivent des valeurs qui représentent l'emplacement de la souris dans la zone du canevas.

Ces valeurs sont obtenues à partir des propriétés jQuery `pageX` et `pageY`, qui font référence au décalage du pointeur de la souris par rapport au coin supérieur gauche du document conteneur. Par conséquent, comme le canevas est lui-même légèrement décalé par rapport à cet emplacement, les valeurs de ces décalages (dans `offsetLeft` et `offsetTop`) viennent se soustraire respectivement de `pageX` et `pageY` :

```
var xpos = event.pageX - canvas.offsetLeft
var ypos = event.pageY - canvas.offsetTop
```

Maintenant que nous connaissons la position du pointeur de souris par rapport au canevas, les deux lignes suivantes testent la valeur de `pendown`. Si elle vaut `true`, le bouton de la souris est pressé, donc un appel à `lineTo` permet de tracer un trait jusqu'à l'emplacement courant. Sinon, le stylo est relevé et un appel à `moveTo` met simplement à jour l'emplacement courant du pointeur :

```
if (pendown) context.lineTo(xpos, ypos)
else context.moveTo(xpos, ypos)
```

Ensuite, un appel à la méthode `stroke` applique immédiatement toutes les commandes de dessin apportées au canevas. Ces cinq lignes de code se chargent de la gestion du dessin, mais il reste à suivre l'état du bouton de la souris, dont se chargent les deux lignes finales. Elles interceptent les évènements `mousedown` et `mouseup`, pour régler `pendown` à `true` quand le bouton de la souris est pressé ou à `false` s'il est relâché :

Cet exemple illustre la combinaison de trois gestionnaires d'évènements différents qui œuvrent ensemble pour créer un utilitaire simple avec l'utilisation de variables locales pour des expressions internes et des variables globales lorsqu'un objet ou l'état de quelque chose doit demeurer à la disposition de plusieurs fonctions.

Autres évènements de la souris

Les évènements `mouseenter` et `mouseleave` se déclenchent lorsque la souris passe dans un élément ou le quitte. Aucune valeur de position n'est fournie car vous êtes supposé prendre une décision booléenne à propos de ces évènements.

Dans l'exemple 21-9, deux fonctions anonymes viennent s'associer à ces évènements pour modifier le HTML d'un élément, comme illustré à la figure 21-8.

Exemple 21-9. Détection de l'entrée et de la sortie de la souris dans un élément

```
<!DOCTYPE html>
<html>
  <head>
    <title>Évènements souris autres</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <h2 id='test'>Passez la souris sur moi</h2>
    <script>
      $('#test').mouseenter(function() { $(this).html('Hé, arrêtez de me chatouiller!')
} )
      $('#test').mouseleave(function() { $(this).html('Où allez-vous?')
} )    </script>
  </body>
</html>
```

Figure 21-8. Détection de l'entrée et de la sortie de la souris d'un élément

Lorsque la souris pénètre parmi les limites de l'élément sélectionné, un appel à `html` provoque la modification de la propriété `innerHTML` de l'élément. Ensuite, quand la souris quitte l'élément, un nouvel appel à `html` modifie à nouveau le contenu HTML de l'élément.

Méthodes alternatives de la souris

Nombre d'autres fonctions d'évènements de souris existent en jQuery pour couvrir une vaste plage de circonstances particulières, toutes détaillées (en anglais) sur *api.jquery.com/category/events/mouseevents*.

Ainsi, vous pouvez aussi exploiter les méthodes alternatives `mouseover` et `mouseout` pour obtenir des résultats comparables :

```
$('#test').mouseover(function() { $(this).html('Découpez-le!')              } )
$('#test') .mouseout(function() { $(this).html('Essayez ceci cette fois...') } )
```

Mais vous pouvez aussi exploiter la méthode `hover` et lui associer deux gestionnaires en un seul appel de fonction, comme suit :

```
$('#test').hover(function() { $(this).html('Découpez-le!')              },
                 function() { $(this).html('Essayez ceci cette fois...') } )
```

Si vous envisagez de créer des effets combinés avec `mouseover` et `mouseout`, la méthode `hover` se distingue clairement comme la fonction la plus logique à utiliser mais il existe encore un autre moyen pour atteindre le même résultat, qui s'appelle le chainage (expliqué plus loin, à la section « Chainage de méthodes », page 529), à l'aide de code comme celui-ci :

```
$('#test').mouseover(function() { $(this).html('Découpez-le!')              } )
          .mouseout(function() { $(this).html('Essayez ceci cette fois...') } )
```

Dans ce cas-ci, l'opérateur point au début de la deuxième partie de l'instruction assure l'enchainement de celle-ci à la première, de sorte qu'elles forment une chaine de méthodes.

Les exemples précédents montrent comment capturer un clic et un mouvement de souris, ainsi que les évènements de clavier. Ils s'adaptent donc plutôt aux environnements de bureau, ce à quoi jQuery se destine essentiellement. Cependant, une version de jQuery existe pour les appareils mobiles, qui fournit tout le contrôle des gestionnaires de gestes et de mouvements tactiles que vous pouvez imaginer. Elle est disponible sur *jquerymobile.com*.

Évènement submit

Lors de la soumission d'un formulaire, il est parfois nécessaire d'effectuer des vérifications d'erreurs sur les données entrées avant de les soumettre au serveur. Une possibilité consiste à intercepter l'évènement `submit` du formulaire comme dans l'exemple 21-10. La figure 21-9 montre le résultat du chargement de ce document, puis de son envoi en soumission avec un ou plusieurs champs laissés vides :

Exemple 21-10. Interception de l'évènement submit d'un formulaire

```
<!DOCTYPE html>
<html>
  <head>
    <title>Évènement submit</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <form id='form'>
      Prénom : <input id='prenom' type='text' name='prenom'><br>
      Nom :    <input id='nomfam' type='text' name='nomfam'><br>
      <input type='submit'>
    </form>
    <script>
      $('#form').submit(function()
      {
        if ($('#prenom').val() == '' ||
            $('#nomfam').val() == '')
        {
          alert('Entrez les deux noms!')
          return false
        }
      })
    </script>
  </body>
</html>
```

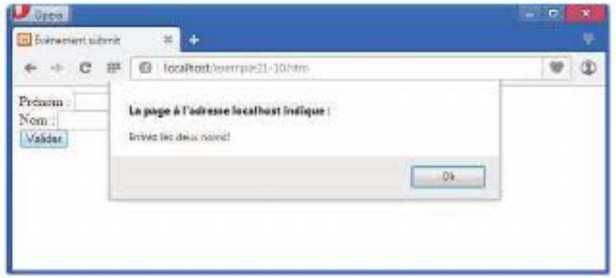



Figure 21-9. Vérification d'entrées de l'utilisateur lors de la soumission

Les nouveautés clés de cet exemple se situent à l'association de l'évènement à une fonction anonyme, comme suit :

```
$('#form').submit(function()
```

Et dans le test des valeurs des deux champs d'entrée pour vérifier s'ils sont vides :

```
if ($('#prenom').val() == '' ||
    $('#nomfam').val() == '')
```

La méthode jQuery `val` permet de récupérer la valeur de la propriété `value` de chaque champ. Ceci est beaucoup plus propre que d'utiliser `$('#prenom')[0]` (comme dans l'exemple 21-8) pour accéder à l'objet du DOM et d'ensuite lui ajouter `value` pour lire la valeur du champ, comme dans : `$('#prenom')[0].value`.

Dans cet exemple, le retour de la valeur `false` quand un les deux champs sont vides provoque l'annulation de la soumission du formulaire. Pour autoriser la soumission, renvoyez `true` ou tout simplement rien du tout.

Effets spéciaux

jQuery commence réellement à montrer tout son potentiel lorsqu'il s'agit de produire des effets spéciaux. Bien que vous puissiez exploiter les transitions de CSS3, leur exploitation n'est pas si facile à gérer de manière dynamique en JavaScript, tandis que jQuery simplifie fortement les opérations puisqu'il suffit de sélectionner un ou plusieurs éléments et de leur appliquer ensuite un ou plusieurs effets.

Les principaux effets disponibles sont le masquage et l'affichage, le fondu du ou vers le néant, le repli et les animations, et ces effets peuvent s'utiliser indépendamment ou en pleine synchronisation ou encore en séquence. Ils prennent également en charge des rappels, c'est-à-dire des fonctions ou méthodes appelées seulement lorsqu'une opération s'achève.

La section suivante énumère quelques-uns des effets de jQuery les plus utiles, qui acceptent chacun jusqu'à trois arguments, comme suit :

Aucun argument
: Lorsqu'aucun argument n'est fourni, la méthode est appelée immédiatement et n'est pas placée dans la file d'attente d'animation.

Durée
: Si vous fournissez une valeur de durée, l'effet se déroule pendant la durée attribuée, qui porte une valeur établie en millisecondes, ou encore les chaine `fast` (rapide) ou `slow` (lent).

Transition
: Seulement deux options de transition (***easing***) sont possibles dans la bibliothèque jQuery, `swing` (variation) et `linear` (linéaire). La valeur par défaut `swing` donne un effet plus naturel que `linear`. Pour obtenir plus d'options de transition, cherchez du côté de plugiciels comme les options de transition de jQuery UI sur *jqueryui.com/easing*.

Rappel
: Si vous fournissez une fonction de rappel (*callback*), elle est appelée à l'achèvement de la méthode d'effet.

Par exemple, vous pouvez appeler la méthode `hide` (masquer) de plusieurs manières, comme celles-ci :

```
$('#objet').hide()
$('#objet').hide(1000)
$('#objet').hide('fast')
$('#objet').hide('linear')
$('#objet').hide('slow', 'linear')
$('#objet').hide(mafonction)
$('#objet').hide(333, mafonction)
$('#objet').hide(200, 'linear', function() { alert('Terminé!') } )
```

La section « Chainage de méthodes » de la page 529 vous montrera que vous pouvez associer des appels de fonctions (qui fournissent les arguments) les unes aux autres et elles s'enchainent tour à tour, comme dans cet exemple qui masque puis réaffiche un élément :

```
$('#objet').hide(1000).show(1000)
```

Nombre de ces méthodes prennent en charge encore d'autres arguments, moins usités. Pour de plus amples informations sur ceux-ci (et sur toutes les méthodes d'effets prises en charge), consultez *api.jquery.com/category/effects*.

Masquer et montrer

L'effet le plus simple reste encore le masquage et l'affichage d'éléments en réaction à des interactions de l'utilisateur. Comme l'indique la section précédente, vous pouvez fournir aucun ou plusieurs arguments aux méthodes `hide` et `show`. En cas d'absence d'argument, le masquage ou l'affichage se produit immédiatement.

Le mode de fonctionnement de ces deux méthodes lorsqu'elles reçoivent des arguments consiste à modifier simultanément les propriétés `width`, `heigth` et `opacity` d'un élément,

jusqu'à aboutir à 0 pour `hide` ou leurs valeurs initiales pour `show`. Lors du masquage, la propriété `display` de l'élément est réglée à `none` au moment du masquage total, puis, après l'appel à `show`, ses valeurs précédentes sont réaffectées à l'élément au moment de sa restauration complète.

L'exemple 21-11 vous permet d'essayer `hide` et `show` par vous-même (figure 21-10).

Exemple 21-11. Masquage et affichage d'un élément

```
<!DOCTYPE html>
<html>
  <head>
    <title>Effets masquage et affichage</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <button id='hide'>Masquer</button>
    <button id='show'>Afficher</button>
    <p id='text'>Cliquez sur les boutons Masquer et Afficher</p>
    <script>
      $('#hide').click(function() { $('#text').hide('slow', 'linear') })
      $('#show').click(function() { $('#text').show('slow', 'linear') })
    </script>
  </body>
</html>
```

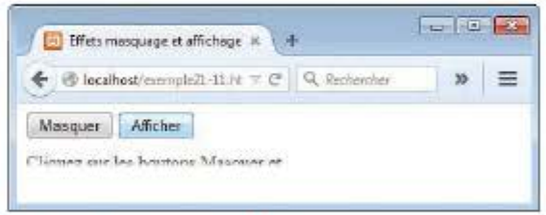


Figure 21-10. L'élément en cours de réaffichage

Méthode toggle (basculement)

Au lieu d'utiliser successivement les méthodes `hide` et `show`, vous disposez de la méthode `toggle`, qui permet de faire évoluer l'exemple précédent pour aboutir à l'exemple 21-12.

Exemple 21-12. Utilisation de la méthode toggle

```
<!DOCTYPE html>
<html>
  <head>
    <title>Effets toggle</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
```

```
  <body>
    <button id='toggle'>Basculer</button>
    <p id='text'>Cliquez sur le bouton Basculer</p>
    <script>
      $('#toggle').click(function() { $('#text').toggle('slow', 'linear') })
    </script>
  </body>
</html>
```

La méthode `toggle` prend les mêmes arguments que `hide` et `show` mais conserve en interne l'état de l'élément pour savoir s'il faut l'afficher ou le masquer.

Quatre méthodes jQuery existent qui définissent tel ou tel état et qui proposent des versions de basculement pour simplifier le code. En plus de `toggle`, vous disposez de `fadeToggle` (basculement avec fondu), `slideToggle` (avec repli) et la classe `toggle`, toutes décrites dans ce chapitre.

Fondu enchainé

Quatre méthodes permettent de gérer les fondus : `fadeIn` (fondu vers l'élément), `fadeOut` (fondu vers la disparition de l'élément), `fadeToggle` (tour à tour de et vers l'élément) et `fadeTo` (vers une opacité ou transparence). Maintenant que vous avez une idée du fonctionnement de jQuery, vous comprenez sans peine que les trois premières sont comparables à `show`, `hide` et `toggle`. La dernière diffère cependant un peu dans la mesure où elle permet de préciser une valeur d'opacité vers laquelle un élément (ou plusieurs) doit se fondre, entre 0 et 1.

L'exemple 21-13 propose quatre boutons pour tester ces méthodes (figure 21-11).

Exemple 21-13. Les quatre méthodes de fondu

```
<!DOCTYPE html>
<html>
  <head>
    <title>Effets de fondu</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <button id='fadeout'>fadeOut</button>
    <button id='fadein'>fadeIn</button>
    <button id='fadetoggle'>fadeToggle</button>
    <button id='fadeto'>fadeTo</button>
    <p id='text'>Cliquez sur les boutons ci-dessus</p>
    <script>
      $('#fadeout')   .click(function() { $('#text').fadeOut(   'slow'     ) })
      $('#fadein')    .click(function() { $('#text').fadeIn(    'slow'     ) })
      $('#fadetoggle').click(function() { $('#text').fadeToggle('slow'     ) })
      $('#fadeto')    .click(function() { $('#text').fadeTo(    'slow', 0.5) })
    </script>
  </body>
</html>
```


Figure 21-11. Le texte a fondu à 50 % d'opacité

Faire glisser des éléments vers le haut et vers le bas

Une autre manière de faire disparaitre et réapparaitre des éléments consiste à modifier progressivement leur hauteur pour les replier vers le haut et les déplier vers le bas. Trois méthodes vous aident en cela : `slideDown`, `slideUp` et `slideToggle`. Elles fonctionnent de manières comparables aux méthodes précédentes. L'exemple 21-14 en offre une démonstration (figure 21-12).

Exemple 21-14. Utilisation des méthodes de glissement

```
<!DOCTYPE html>
<html>
  <head>
    <title>Effets de glissement</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <button id='slideup'>slideUp</button>
    <button id='slidedown'>slideDown</button>
    <button id='slidetoggle'>slideToggle</button>
    <div id='para' style='background:#def'>
      <h2>Extrait du Conte de deux cités - de Charles Dickens</h2>
      <p>Je vois une ville splendide et un peuple glorieux surgir de cet abime;
         et dans ses luttes pour devenir vraiment libre, dans ses triomphes et
         ses défaites, je le vois expier peu à peu les forfaits de cette époque
         et de celle qui l'a précédée et engendrée, et les effacer à tout jamais.
      </p>
    </div>
    <script>
      $('#slideup')    .click(function() { $('#para').slideUp(    'slow') })
      $('#slidedown')  .click(function() { $('#para').slideDown(  'slow') })
      $('#slidetoggle').click(function() { $('#para').slideToggle('slow') })
    </script>
  </body>
</html>
```

Figure 21-12. Le paragraphe glisse vers le haut (slideUp)

Ces méthodes s'avèrent particulièrement utiles lorsque vous avez des menus et sous-menus à ouvrir et fermer de manière dynamique, selon les entrées sélectionnées par l'utilisateur.

Animations

À partir de là, nous pouvons commencer à nous amuser un peu avec le déplacement réel d'éléments dans le navigateur. Pour cela, comme la valeur par défaut `static` ne leur permet pas de bouger, nous devons nous souvenir de donner d'abord à la propriété `position` des éléments une des valeurs `relative`, `fixed` ou `absolute`.

Pour animer un élément, vous fournissez une liste de propriétés CSS (à l'exception des couleurs) à la méthode `animate`. Au contraire des précédentes méthodes d'effets, `animate` exige cette liste de propriétés en premier lieu et ensuite seulement, vous pouvez fournir en arguments la durée, la transition et le rappel que vous souhaitez.

Ainsi, pour animer un ballon rebondissant, utilisez par exemple du code comme celui de l'exemple 21-15, dont la figure 21-13 illustre les résultats.

Exemple 21-15. Création de l'animation d'un ballon rebondissant

```
<!DOCTYPE html>
<html>
  <head>
    <title>Effets d'animation</title>
    <script src='jquery-1.11.2.min.js'></script>
    <style>
      #ballon {
        position  :relative;
      }
      #box {
        width     :640px;
        height    :480px;
        background:green;
        border    :1px solid #444;
      }
    </style>
  </head>
  <body>
    <div id='box'>
      <img id='ballon' src='ballon.png'>
    </div>
    <script>
      rebondir()

      function rebondir()
      {
        $('#balle')
          .animate( { left:'270px', top :'380px' }, 'slow', 'linear')
          .animate( { left:'540px', top :'190px' }, 'slow', 'linear')
          .animate( { left:'270px', top :'0px'   }, 'slow', 'linear')
          .animate( { left:'0px',   top :'190px' }, 'slow', 'linear', rebondir)
      }
    </script>
  </body>
</html>
```

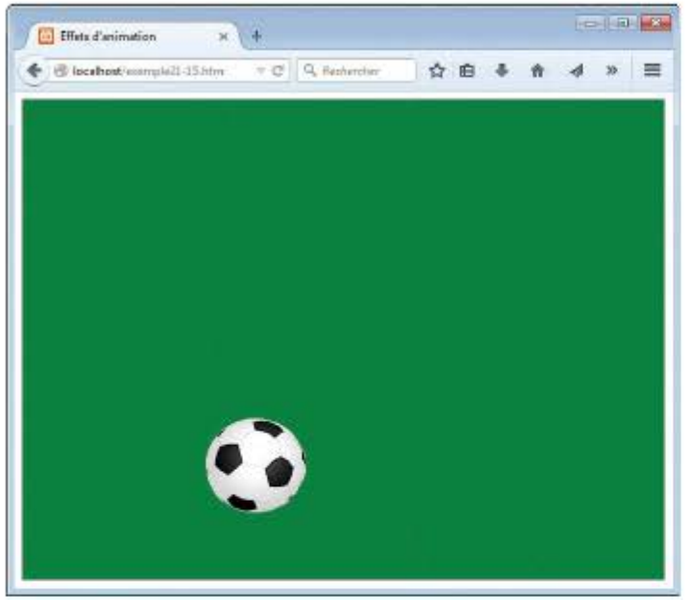


Figure 21-13. Le ballon rebondit dans le navigateur

La section `<style>` de cet exemple règle la propriété `position` du ballon de façon relative à son conteneur. Le conteneur est un élément `<div>` avec une bordure grise et un fond vert.

Ensuite, la section `<script>` définit une fonction `rebondir` où quatre appels à `animate` viennent se placer les uns à la suite des autres.

Remarquez que les noms des propriétés à animer sont fournis sans apostrophe et que des deux-points les séparent des valeurs, autrement dit, sous la forme habituelle des tableaux associatifs.

Vous pouvez aussi fournir des valeurs relatives à la place des valeurs absolues à l'aide des opérateurs `+=` et `-=`. Ainsi, la ligne suivante anime le ballon vers la droite et le haut par étapes de 50 pixels relatifs à sa position courante :

```
.animate( { left:'+=50px', top:'-=50px' }, 'slow', 'linear')
```

Vous pouvez même utiliser les valeurs chaines `hide`, `show` et `toggle` pour modifier une propriété, comme suit :

```
.animate( { height:'hide', width:'toggle' }, 'slow', 'linear')
```


Lorsque vous devez modifier une propriété CSS qui comporte des tirets et devez la passer sans apostrophe, comme `height` et `width` dans l'instruction précédente, vous devez d'abord la convertir en chatMot (*camelCase*), c'est-à-dire supprimer les tirets et placer en capitale la première lettre qui les suit. Ainsi, pour animer la propriété `left-margin` d'un élément, remplacez-la par `leftMargin`. En revanche, lorsque vous fournissez les propriétés sous forme de chaines entourées d'apostrophes, comme `css('font-weight', 'bold')`, vous ne devez pas les convertir en chatMot.

Chainage de méthodes

De la manière dont le chainage des méthodes fonctionne, quand des méthodes jQuery ont reçu leurs arguments, elles s'exécutent en séquence. Par conséquent, chacune de ces méthodes n'est appelée qu'après que la précédente a terminé son exécution. Par contre, toute méthode appelée sans argument s'exécute immédiatement et rapidement, sans animation.

Lors du chargement du code de l'exemple dans un navigateur, le déclenchement de l'animation est obtenu par un appel de la fonction `rebondir`, qui provoque le rebondissement du ballon sur les bords inférieur, droit et supérieur de son conteneur, puis le retour au milieu du bord gauche.

Utiliser les fonctions de rappel

Si nous en restons là, l'animation se termine dans l'exemple précédent après quatre animations. Une fonction de rappel permet de la relancer à la fin de chaque cycle. Cette fois, cette fonction de rappel est une fonction nommée et séparée.

Et comme l'animation se situe dans une fonction nommée `rebondir`, il suffit de rappeler cette même fonction en guise de fonction de rappel dans la quatrième instruction :

```
.animate( { left:'0px',   top :'190px' }, 'slow', 'linear', rebondir)
```

L'utilisation de la méthode `animate` permet d'animer de nombreuses propriétés CSS, à l'exception remarquable des couleurs. Cependant, il est même possible de créer des animations de couleurs à l'aide du module d'extension jQuery UI, qui ajoute la possibilité de créer des effets de changement de couleurs (et bien d'autres) très agréables pour l'œil. Pour de plus amples informations, reportez-vous à *jqueryui.com*.

Arrêter des animations

Plusieurs méthodes de jQuery permettent d'arrêter des animations en cours ou de terminer une chaine d'animations. Par exemple, `clearQueue` vide toutes les animations mémorisées dans leur file d'attente. La méthode `stop` arrête toute animation en cours et la méthode `finish` arrête l'animation en cours et vide toute la file d'attente des animations. Autrement dit, `finish` équivaut à `stop`, suivi de `clearQueue`.

Sachant ceci, pour prolonger l'exemple précédent et lui donner un peu plus d'interactivité avec l'utilisateur, nous pouvons par exemple faire en sorte que le ballon attende un clic de souris et, au déclenchement de l'évènement `click`, arrête l'animation. Pour cela, il suffit d'ajouter l'instruction en une seule ligne suivante à la suite de la fonction `rebondir` :

```
$('#ballon').click(function() { $(this).finish() })
```

Si vous réussissez à cliquer sur le ballon en mouvement, la méthode `finish` arrête l'animation en cours, vide la file d'attente et ignore toute fonction de rappel. Autrement dit, le ballon reste en place.

Pour de plus amples informations (en anglais) sur la gestion des files d'attente (*queue*) de jQuery, consultez *api.jquery.com/queue*, où vous apprendrez notamment à manipuler directement le contenu des files d'attente pour obtenir exactement les effets souhaités.

Manipuler le DOM

Du fait que jQuery est intimement lié au DOM, les exemples précédents de ce chapitre ont déjà nécessairement fait appel à certaines de ses fonctions d'accès au DOM, telles que `html` et `val`. Nous poursuivons l'examen détaillé de toutes les méthodes DOM pour savoir exactement à quoi jQuery vous donne accès et comment.

Dans l'exemple 21-3, vous avez vu comment utiliser la méthode `html` pour modifier la propriété `innerHTML` d'un élément. Cette méthode permet soit de modifier du HTML, soit de le lire dans un document HTML. L'exemple 21-16 (avec les éléments de jQuery en gras) montre comment récupérer le contenu HTML d'un élément et la figure 21-14 en illustre le résultat.

Exemple 21-16. Affichage dans une fenêtre d'alerte du contenu HTML d'un élément

```
<!DOCTYPE html>
<html>
  <head>
    <title>Le DOM - html et texte</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <h2>Exemple de document</h2>
    <p id='intro'>Ceci est un exemple de document HTML</p>
    <script>
      alert($('#intro').html())
    </script>
  </body>
</html>
```

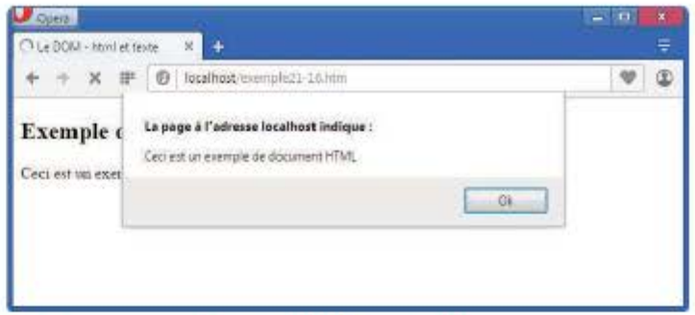



Figure 21-14. Lecture et affichage du HTML d'un élément

Si vous ne donnez aucun argument à la méthode `html`, celle-ci lit le contenu HTML de l'élément au lieu de le modifier.

Différence entre les méthodes text et html

Lorsque vous traitez des documents XML, vous ne pouvez pas utiliser la méthode `html`, tout simplement parce qu'elle ne fonctionne pas, car elle n'est conçue que pour du HTML. Utilisez plutôt la méthode `text` pour obtenir des résultats équivalents et, ceci, tant sur du HTML que sur du XML, comme suit :

```
texte = $('#intro').text()
```

La différence entre les deux se situe simplement dans le fait que `html` traite le contenu comme du HTML, tandis que `text` le traite comme du texte. Ainsi, supposons par exemple que vous souhaitez affecter la chaine suivante à un élément :

```
<a href='https://www.google.fr'>Consultez Google</a>
```

Si vous affectez cela à un élément HTML à l'aide de la méthode `html`, celle-ci modifie le DOM avec le nouvel élément `<a>` et le lien peut recevoir des clics. Par contre, si vous faites cela dans un document, tant XML que HTML, à l'aide de la méthode `text`, alors cette chaine est d'abord convertie en séquences d'échappement dans le texte (les caractères HTML tels que `<` deviennent `&lt ;` et ainsi de suite), puis insérée dans l'élément et aucun élément n'est ajouté au DOM.

Méthodes val et attr

Deux autres méthodes permettent d'interagir avec le contenu d'éléments. La première, `val`, permet de définir et de lire la valeur d'un élément d'entrée, comme illustré dans l'exemple 21-10, qui lit le contenu des champs de prénom et de nom de famille. Pour définir une valeur, fournissez-la simplement en tant qu'argument de la méthode, comme ceci :

```
$('#motdepasse').val('monpasse123')
```

La deuxième méthode, attr, permet de modifier et de lire les attributs d'un élément, comme illustré dans l'exemple 21-17, où un lien vers le site web de Google est remplacé complètement par un autre vers celui de Yahoo!

Exemple 21-17. Modification d'attributs à l'aide de la méthode attr

```
<!DOCTYPE html>
<html>
  <head>
    <title>Le DOM - attr</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <h2>Exemple de document</h2>
    <p><a id='lien' href='https://www.google.fr' title='Google'>Consultez Google</a></p>
    <script>
      $('#lien').text('Consultez Yahoo!')
      $('#lien').attr( { href :'https://qc.yahoo.com/', title:'Yahoo!' } )
      alert('Le nouveau HTML est :\n' + $('p').html())
    </script>
  </body>
</html>
```

La première instruction jQuery emprunte la méthode text pour ne modifier que le texte contenu dans l'élément <a>, tandis que la deuxième modifie les valeurs des attributs href et title en accord, et fournit les données sous forme d'un tableau associatif. La troisième instruction lit le contenu de l'élément modifié par la méthode html et l'affiche dans une fenêtre d'alerte (figure 21-15).

Figure 21-15. Le lien est complètement modifié

Pour lire la valeur d'un attribut, écrivez par exemple :

```
url = $('#lien').attr('href')
```


Ajouter et supprimer des éléments

S'il est possible d'insérer des éléments dans le DOM à l'aide de la méthode html, cette solution n'est réellement envisageable que pour la création d'éléments enfants d'un élément déterminé. Par conséquent, jQuery fournit un certain nombre de méthodes pour manipuler des portions du DOM.

Ces méthodes sont append, prepend, after, before, remove et empty. L'exemple 21-18 en propose une démonstration.

Exemple 21-18. Ajout et suppression d'éléments

```
<!DOCTYPE html>
<html>
  <head>
    <title>Modifications du DOM</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <h2>Exemple de document</h2>
    <a href='https://www.google.fr' title='Google'>Consultez Google</a>
    <code>
      Ceci est une section de code
    </code>
    <p>
      <button id='a'>Supprimer l'image</button>
      <button id='b'>Vider la citation</button>
    </p>
    <img id='ballon' src='ballon.png'>
    <blockquote id='citation' style='border:1px dotted #444; height:20px;'>
      Citation
    </blockquote>
    <script>
      $('a').prepend('Lien : ')
      $("[href^='http']").append(" <img src='lien.png'>")
      $('code').before('<hr>').after('<hr>')
      $('#a').click(function() { $('#ballon').remove() } )
      $('#b').click(function() { $('#citation').empty() } )
    </script>
  </body>
</html>
```

La figure 21-16 illustre les effets produits par les méthodes prepend, append, before et after, appliquées à quelques éléments.

Figure 21-16. Un document pour illustrer différents éléments

La méthode `prepend` insère la chaine `Lien :` avant le texte ou le HTML intérieur de tous les éléments `<a>`, comme suit :

```
$('a').prepend('Lien : ')
```

Ensuite, un sélecteur d'attribut permet de sélectionner tous les éléments qui possèdent un attribut `href` qui commence par `http` (donc aussi `https`). La chaine `http` concerne les liens qui ne sont pas relatifs (et donc sont absolus), auquel cas une icône de lien externe vient s'ajouter à la fin du texte ou du HTML intérieur de tous les éléments correspondants, comme ceci :

```
$("[href^='http']").append(" <img src='lien.png'>")
```

L'opérateur `^=` permet de n'identifier que le début d'une chaine. Si nous avions utilisé seulement l'opérateur `=`, seules les chaines entièrement identiques auraient été identifiées. Les chapitres 18 et 19 examinent en détail les sélecteurs CSS.

Ensuite, l'utilisation de méthodes chainées permet d'appeler les méthodes `before` (avant) et `after` (après) pour placer des éléments frères (de même niveau dans le DOM) respectivement devant et à l'arrière d'un autre élément. Dans ce cas-ci, nous plaçons un trait horizontal `<hr>` avant et après des éléments de la section `<code>`, comme suit :

```
$('code').before('<hr>').after('<hr>')
```


Ensuite, un peu d'interaction avec l'utilisateur emprunte deux boutons. Lors d'un clic de souris sur le premier bouton, la méthode `remove` supprime l'élément `<img>` qui contient le ballon, comme ceci :

```
$('#a').click(function() { $('#ballon').remove() } )
```

À ce moment-là, l'image n'existe plus dans le DOM, ce que vous pouvez vérifier si vous sélectionnez le contenu du navigateur, cliquez dessus et utilisez Examiner l'élément dans la plupart des navigateurs principaux ou si vous appuyez sur [F12] dans Internet Explorer.

Finalement, la méthode `empty` est appliquée à l'élément de citation `<blockquote>` lors du clic sur le second bouton pour vider le contenu de l'élément mais laisser l'élément en place dans le DOM, comme suit :

```
$('#b').click(function() { $('#citation').empty() } )
```

Appliquer des classes de manière dynamique

Le fait de pouvoir changer la classe employée par un élément s'avère parfois bien pratique, de même que d'ajouter une classe ou d'en retirer une à un élément. Ainsi, supposez que vous disposez d'une classe appelée `read` (lu) qui permet de marquer d'un style les publications d'un blogue qui ont déjà été lues. La méthode `addClass` permet d'ajouter une classe à une publication (post), comme suit :

```
$('#post23').addClass('read')
```

Vous pouvez en ajouter plusieurs à la fois en les séparant par des espaces, comme ceci :

```
$('#post23').addClass('read liked')
```

Et si l'utilisateur décide de marquer de nouveau une publication comme non lue afin de la relire plus tard, la méthode `removeClass` intervient, comme cela :

```
$('#post23').removeClass('read')
```

Les autres classes utilisées par la publication demeurent inchangées lors d'une telle suppression.

Lorsque vous devez fournir la possibilité d'ajouter et d'ôter de manière répétitive une classe à un élément, vous disposez aussi de la méthode `toggleClass`, plus simple à utiliser :

```
$('#post23').toggleClass('read')
```

Avec cette instruction, si la publication n'utilise pas la classe, elle vient s'ajouter ; sinon, elle est supprimée.

Modifier des dimensions

La manipulation des dimensions représente toujours une tâche complexe du développement web parce que les différents navigateurs ont tendance à utiliser des variantes de valeurs. Une des grandes forces de jQuery réside précisément dans son travail de

normalisation de ces types de valeurs, de sorte qu'en passant par lui, vos pages apparaissent comme vous le souhaitez dans tous les navigateurs principaux.

Trois types de dimensions existent : la largeur (*width*) et la hauteur (*height*), la largeur (*inner width*) et la hauteur intérieures (*inner height*), et la largeur (*outer width*) et hauteur (*outer height*) extérieures d'un élément. Examinons-les séparément.

Méthodes width et height

Les méthodes `width` et `height` permettent de connaître la largeur et la hauteur du premier élément identifié par un sélecteur, ou de définir la largeur et la hauteur de tous les éléments identifiés. Ainsi, pour obtenir la largeur d'un élément d'identifiant `elem`, écrivez cette instruction :

```
largeur = $('#elem').width()
```

La valeur renvoyée à la variable `largeur` est numérique, différente de celle de la valeur CSS obtenue par un appel à la méthode `css` comme la suivante, qui renvoie par exemple `230px` au lieu du nombre `230` :

```
largeur = $('#elem').css('width')
```

Vous pouvez aussi connaître la largeur de la fenêtre ou du document courants, comme suit :

```
largeur = $(window).width()
largeur = $(document).width()
```

Lorsque vous passez les objets `window` ou `document` à jQuery, vous ne pouvez pas déterminer leur largeur ou leur hauteur à l'aide de la méthode `css`. À la place, vous devez appeler les méthodes `width` et `height`.

La valeur renvoyée est indépendante du réglage de `box-sizing` (voir le chapitre 19). Si vous devez prendre en compte `box-sizing`, utilisez plutôt la méthode `css` avec l'argument `width`, comme suit (mais rappelez-vous d'ôter le `px` ajouté après la partie numérique si vous comptez travailler avec les valeurs retournées) :

```
largeur = $('#elem').css('width')
```

La définition des valeurs est tout aussi aisée. Par exemple, pour régler tous les éléments qui utilisent la classe `box` à la taille de 100 × 100 pixels, écrivez :

```
$('.box').width(100).height(100)
```

L'exemple 21-19 regroupe ces actions en un programme dont la figure 21-17 illustre les résultats.

Exemple 21-19. Lecture et écriture de dimensions d'éléments

```
<!DOCTYPE html>
<html>
  <head>
    <title>Dimensions</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <p>
      <button id='getdoc'>Lire largeur document</button>
      <button id='getwin'>Lire largeur fenêtre</button>
      <button id='getdiv'>Lire largeur div</button>
      <button id='setdiv'>Régler largeur div à 150 pixels</button>
    </p>
    <div id='resultat' style='width:300px; height:50px; background:#def;'></div>
    <script>
      $('#getdoc').click(function()
      {
        $('#resultat').html('Largeur document : ' + $(document).width())
      } )

      $('#getwin').click(function()
      {
        $('#resultat').html('Largeur fenêtre : ' + $(window).width())
      } )

      $('#getdiv').click(function()
      {
        $('#resultat').html('Largeur div : ' + $('#resultat').width())
      } )

      $('#setdiv').click(function()
      {
        $('#resultat').width(150)
        $('#resultat').html('Largeur div : ' + $('#resultat').width())
      } )
    </script>
  </body>
</html>
```

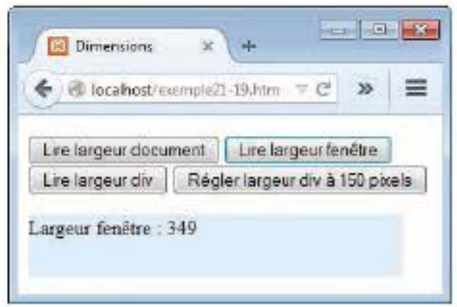


Figure 21-17. Lecture et écriture de dimensions d'éléments

Au début du corps de la page se présentent quatre boutons : trois pour obtenir les largeurs du document, de la fenêtre et d'un élément `<div>` qui apparait juste en dessous des boutons, et un quatrième pour définir la largeur de la div avec une valeur différente. La section script comporte quatre instructions jQuery dont les trois premières récupèrent la largeur des objets donnée, puis l'affichent dans le HTML de la div.

L'instruction finale comporte deux parties : la première réduit la largeur de l'élément `<div>` à 150 pixels et la seconde affiche au sein de cette div la nouvelle valeur de largeur, obtenue à l'aide de la méthode `width`, pour garantir l'affichage de la valeur calculée.

Lorsque l'utilisateur zoome dans la fenêtre tant en avant qu'en arrière, aucun des principaux navigateurs ne remarque l'évènement ou, en tout cas, d'aucune manière que JavaScript puisse détecter de manière fiable. Par conséquent, jQuery ne peut prendre en compte de zoom lorsqu'il applique ou renvoie des valeurs dimensionnelles. Dans ces circonstances, il se peut que vous obteniez des résultats inattendus.

Méthodes innerWidth et innerHeight

Lorsqu'il s'avère nécessaire de prendre également en compte les bordures, les marges intérieures et d'autres propriétés dans le traitement des dimensions, vous pouvez utiliser les méthodes `innerWidth` et `innerHeight` pour connaitre la largeur et la hauteur du premier élément identifié par le sélecteur, y compris les marges intérieures mais les bordures non comprises.

Par exemple, l'instruction suivante renvoie la largeur intérieure de l'élément d'identifiant `elem`, y compris les marges intérieures :

```
largeurint = $('#elem').innerWidth()
```

Méthodes outerWidth et outerHeight

Pour connaitre les dimensions d'un élément en tenant compte à la fois de ses marges intérieures et de ses bordures, appelez les méthodes `outerWidth` et `outerHeight`, comme suit :

```
largeurext = $('#elem').outerWidth()
```

Pour inclure aussi toutes les marges dans la valeur renvoyée, passez la valeur `true` en argument d'appel de ces méthodes :

```
largeurext = $('#elem').outerWidth(true)
```

Les valeurs renvoyées par les méthodes `inner…` et `outer…` ne sont pas nécessairement entières et peuvent être fractionnaires dans certains cas. Ces méthodes ne détectent pas le facteur de zoom et il n'est pas permis de les utiliser sur les objets `window` ou `document`. Pour ceux-ci, utilisez les méthodes `width` et `height`.

Parcours du DOM

Si vous retournez un instant à la section sur le modèle objet de document (DOM) du chapitre 13, vous vous rappelez que les pages web sont construites d'une façon qui s'apparente à des familles étendues. Il y a des objets parents et enfants, des frères (ou sœurs), des grands-parents et des petits-enfants, et même des relations entre les éléments qui évoquent des cousins, des oncles (tantes) et ainsi de suite. Ainsi, dans le petit extrait de code suivant, les éléments `<li>` sont des enfants de l'élément `<ul>`, lui-même parent des éléments `<li>` :

```
<ul>
  <li>Élément 1</li>
  <li>Élément 2</li>
  <li>Élément 3</li>
</ul>
```

Et, comme dans les familles, plusieurs façons permettent de faire référence aux éléments HTML, telles que de manière absolue, en partant du niveau de la fenêtre (`window`) et en se déplaçant vers le bas, ce qui s'appelle le parcours du DOM (*traversal*). Mais vous pouvez aussi exploiter la relation entre un élément et un autre pour faire référence aux éléments. En pratique, vous avez le choix, en fonction de ce qui revêt le plus de sens dans un projet déterminé.

Par exemple, lorsque vous construisez une page web pour qu'elle soit la plus autonome possible et vous autoriser le copier et coller d'éléments d'une page vers d'autres, il est préférable de faire référence aux éléments proches les uns des autres par un adressage relatif. Quelle que soit la solution que vous adoptez, jQuery offre une vaste gamme de fonctions pour cibler les éléments avec précision.

Éléments parents

Pour faire référence au parent direct d'un élément, utilisez la méthode `parent` comme suit :

```
mon_parent = $('#elem').parent()
```

Quel que soit le type de l'élément `elem`, l'objet `mon_parent` reçoit un objet jQuery qui permet de cibler le parent de l'élément. En fait, comme les sélecteurs peuvent faire référence à plusieurs éléments, l'appel renvoie en réalité un objet qui fait référence à une liste d'éléments parents, même si elle n'en contient qu'un, avec une entrée par élément identifié.

Comme un parent peut avoir de nombreux enfants, vous vous demandez peut-être si cette méthode est susceptible de renvoyer plus d'éléments qu'il n'y a de parents. Ainsi, dans l'extrait de code précédent, avec trois éléments `<li>`, l'instruction suivante renvoie-t-elle trois éléments parents parce que trois correspondances sont identifiables, alors qu'il n'y a qu'un seul `<ul>` parent ?

```
mon_parent = $('li').parent()
```

La réponse est non, parce que jQuery est suffisamment intelligent pour reconnaitre tous les doublons et les filtrer. Pour vérifier ce fait, la ligne suivante affiche le nombre d'éléments parents et le résultat vaut 1 :

```
alert($('li').parent().length)
```

Nous pouvons provoquer une action quand le sélecteur est identifié, comme changer en `bold` la propriété `font-weight` de l'élément parent de l'extrait de code précédent, comme suit :

```
$('li').parent().css('font-weight', 'bold')
```

Utiliser un filtre

Pour filtrer possiblement le parent auquel appliquer la modification souhaitée, vous pouvez passer un sélecteur à `parent`. Pour illustrer cette option, l'exemple 21-20 contient trois petites listes et deux instructions jQuery.

Exemple 21-20. Accès aux éléments parents

```
<!DOCTYPE html>
<html>
  <head>
    <title>Parcours du DOM : parent</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <ul>
      <li>Élément 1</li>
      <li>Élément 2</li>
      <li>Élément 3</li>
    </ul>
    <ul class='memo'>
      <li>Élément 1</li>
      <li>Élément 2</li>
      <li>Élément 3</li>
    </ul>
    <ul>
      <li>Élément 1</li>
      <li>Élément 2</li>
      <li>Élément 3</li>
    </ul>
    <script>
      $('li').parent()       .css('font-weight',     'bold')
      $('li').parent('.memo').css('list-style-type', 'circle')
    </script>
  </body>
</html>
```

Les trois listes sont identiques mais l'élément `<ul>` de la deuxième est déclaré comme étant de la classe `memo`. Dans la section de script, la première instruction applique la valeur `bold` à la propriété `font-weight` de tous les parents d'éléments `<li>` ; par conséquent, tous les éléments `<ul>` s'affichent en gras.

La deuxième instruction, fort semblable, passe en plus le nom de la classe `memo` à la méthode `parent`, de sorte que seul ce parent est concerné. La méthode `css` règle la propriété `list-style-type` de la liste sélectionnée à `circle`. La figure 21-18 montre les effets de ces deux instructions.

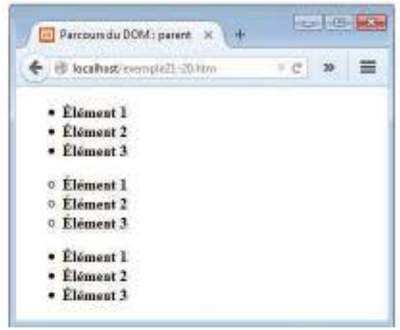


Figure 21-18. Accès aux parents avec et sans filtre

Sélectionner tous les éléments ancêtres

Nous venons de voir comment sélectionner des parents directs d'éléments mais vous pouvez également sélectionner des ancêtres de tous niveaux, jusqu'à aboutir à l'élément racine `<html>` à l'aide de la méthode `parents`. Vous vous interrogez peut-être sur l'intérêt de ceci. L'intérêt réside dans la possibilité de remonter par exemple jusqu'au premier élément `<div>` dans la généalogie pour lui appliquer un style en fonction d'une réaction dynamique à un évènement apparu plus bas dans la généalogie.

Ce genre de sélection peut s'avérer plus évolué que vous ne le pensez pour l'instant et vous n'en voyez peut-être pas l'intérêt, mais lorsque vous en aurez un jour besoin, sachez que cela existe et s'utilise comme suit :

```
$('#elem').parents('div').css('background', 'yellow')
```

En pratique, ceci ne correspond peut-être pas exactement à ce que vous souhaitez, parce que la sélection porte sur tous les éléments `<div>` de la généalogie et que vous ne souhaitez pas les modifier tous, ni ceux situés plus haut dans la généalogie. Pour ce genre de situation, ajoutez un filtrage supplémentaire à la sélection à l'aide de la méthode `parentsUntil`.

La méthode `parentsUntil` parcourt vers le haut la généalogie de la même manière que `parents`, mais s'arrête au premier élément identifié par le filtre de sélection (soit un élément `<div>` dans l'exemple suivant). Ainsi, vous l'utilisez de la même manière que dans l'instruction précédente, tout en étant certain qu'elle sélectionne seulement l'élément exact souhaité :

```
$('#elem').parentsUntil('div').css('background', 'yellow')
```

L'exemple 21-21 illustre la différence entre les deux méthodes. Il contient deux ensembles d'éléments imbriqués, chacun placé dans un élément `<div>` parent. La section script reprend un exemple d'appel de chacune des méthodes `parents` et `parentsUntil`.

Exemple 21-21. Utilisation des méthodes parents et parentsUntil

```
<!DOCTYPE html>
<html>
  <head>
    <title>Parcours du DOM : parents</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <div>
      <div>
        <section>
          <blockquote>
            <ul>
              <li>Élément 1</li>
              <li id='elem'>Élément 2</li>
              <li>Élément 3</li>
            </ul>
          </blockquote>
        </section>
      </div>
      <div>
        <section>
          <blockquote>
            <ul>
              <li>Élément 1</li>
              <li>Élément 2</li>
              <li>Élément 3</li>
            </ul>
          </blockquote>
        </section>
      </div>
    </div>
    <script>
      $('#elem').parents('div')      .css('background',      'yellow')
      $('#elem').parentsUntil('div').css('text-decoration', 'underline')
    </script>
  </body>
</html>
```

La figure 21-19 montre que la première instruction jQuery règle la couleur de fond à jaune pour tout le contenu. Le parcours de l'arborescence hiérarchique vers le haut, jusqu'à l'élément `<html>` à l'aide de la méthode `parents` rencontre deux éléments `<div>` (celui surligné de gras, qui contient la liste avec l'élément `<li>` d'identifiant `elem`, et son propre élément `<div>` parent qui contient les deux autres ensembles d'éléments imbriqués) et ceux-ci sont sélectionnés pour la modification.

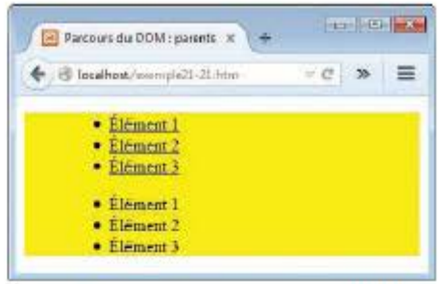


Figure 21-19. Comparaison des méthodes parents et parentsUntil

En revanche, la deuxième instruction jQuery emploie `parentsUntil` pour que la sélection s'arrête au premier élément `<div>` identifié. Par conséquent, le soulignement ne s'applique qu'au `<div>` parent le plus proche qui contient l'élément `<li>` d'identifiant `elem`. La section `<div>` extérieure n'est pas atteinte et de ce fait, comme elle n'est pas concernée, la deuxième liste n'est pas soulignée.

Éléments enfants

Pour accéder aux éléments enfants d'un élément, utilisez la méthode `children`, comme suit :

```
mes_enfants = $('#elem').children()
```

Comme la méthode `parent`, cette méthode ne descend que d'un niveau et renvoie une liste d'aucun, d'un ou de plusieurs sélections identifiées. Elle accepte aussi un argument de filtrage pour distinguer les enfants, comme ceci :

```
enfants_li = $('#elem').children('li')
```

Ceci permet de ne sélectionner que les éléments `<li>` enfants.

Pour plonger plus loin parmi les générations, utilisez la méthode `find`, qui correspond à l'inverse de `parents`, comme suit :

```
descendants_li = $('#elem').find('li')
```

Au contraire de `parents`, vous devez fournir un sélecteur de filtrage à la méthode `find` et, si vous souhaitez obtenir tous les descendants, fournissez alors le sélecteur universel, comme ceci :

```
tous_descendants = $('#elem').find('*')
```

Éléments frères

Lorsqu'il s'agit de sélectionner les éléments frères (*siblings*), vous disposez d'une gamme encore plus vaste de méthodes, à commencer par `siblings`.

La méthode `siblings` renvoie tous les éléments identifiés, enfants du parent commun, *sauf* l'élément utilisé pour la sélection. Par exemple, dans l'extrait de code suivant, si vous recherchez les frères de l'élément `<li>` d'identifiant deux, vous n'obtenez que les premier et troisième éléments `<li>`.

```
<ul>
  <li>Élément 1</li>
  <li id='deux'>Élément 2</li>
  <li>Élément 3</li>
</ul>
```

Et voici un exemple d'instruction qui permet de mettre en gras les éléments frères :

```
$('#deux').siblings().css('font-weight', 'bold')
```

Pour restreindre encore les frères à retourner, fournissez un filtre à la méthode `siblings`. Par exemple, pour ne sélectionner que les frères qui utilisent la classe `nouveau`, écrivez une instruction du genre de celle-ci :

```
$('#deux').siblings('.nouveau').css('font-weight', 'bold')
```

L'exemple 21-22 affiche une liste non numérotée de sept éléments (librement alignée en colonnes à l'aide d'espaces), dont quatre utilisent la classe `nouveau` et le deuxième possède l'identifiant `deux`.

Exemple 21-22. Sélection et filtrage d'éléments frères

```
<!DOCTYPE html>
<html>
  <head>
    <title>Parcours du DOM : siblings</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <ul>
      <li          class='nouveau'>Élément 1</li>
      <li id='deux' class='nouveau'>Élément 2</li>
      <li                          >Élément 3</li>
      <li          class='nouveau'>Élément 4</li>
      <li          class='nouveau'>Élément 5</li>
      <li                          >Élément 6</li>
      <li                          >Élément 7</li>
    </ul>
    <script>
      $('#deux').siblings('.nouveau').css('font-weight', 'bold')
    </script>
  </body>
</html>
```

La figure 21-20 illustre les résultats du chargement dans le navigateur et de l'instruction jQuery, où seuls les éléments 1, 4 et 5 apparaissent en gras, même si l'élément 2 possède aussi la classe `nouveau`. Comme l'appel de la méthode part de ce dernier élément, il est en effet exclu de la sélection.

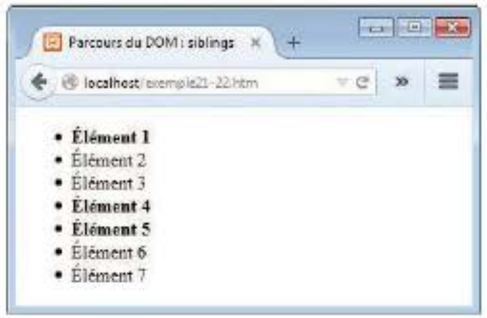


Figure 21-20. Sélection d'éléments frères

Comme la méthode `siblings` omet l'élément sur lequel elle est appelée (que nous appellerons l'*appelant*), elle ne permet pas d'elle-même de sélectionner tous les enfants d'un élément parent. Pour obtenir ceux-ci dans l'exemple précédent, vous pouvez écrire une instruction comme la suivante qui, par exemple, renvoie tous les frères (y compris l'appelant) qui utilisent la classe `nouveau` :

```
$('#deux').parent().children('.nouveau').css('font-weight',
'bold')
```

Mais vous disposez par ailleurs de la méthode `andSelf` (et soi-même) à ajouter à la sélection pour obtenir le même résultat, comme suit :

```
$('#deux').siblings('.nouveau').andSelf().css('font-weight',
'bold')
```

Sélectionner les éléments suivant et précédent

Pour atteindre un contrôle plus affiné sur la sélection des éléments frères, vous pouvez encore réduire un peu plus les éléments renvoyés à l'aide des méthodes `next` et `prev`, ainsi que leurs versions étendues. Par exemple, pour faire référence à l'élément qui suit immédiatement un sélecteur, utilisez une instruction telle que celle-ci, qui règle l'affichage en gras du ou des éléments identifiés :

```
$('#nouveau').next().css('font-weight', 'bold')
```

Dans le cas de l'extrait de code suivant, librement aligné en colonnes à l'aide d'espaces, le troisième élément porte l'identifiant `nouveau`, donc l'instruction précédente renvoie le quatrième élément :

```
<ul>
  <li           >Élément 1</li>
  <li           >Élément 2</li>
  <li id='nouveau'>Élément 3</li>
```

```
  <li             >Élément 4</li>
  <li             >Élément 5</li>
</ul>
```

Jusqu'ici, la simplicité demeure au rendez-vous. Ensuite, pour référencer tous les frères suivant un élément, utilisez la méthode `nextAll`, comme suit (où l'extrait précédent identifie les deux derniers éléments) :

```
$('#nouveau').nextAll().css('font-weight', 'bold')
```

Lors de l'appel de `nextAll`, fournissez un filtre pour effectuer une sélection à partir des éléments identifiés, comme dans l'instruction suivante, qui n'applique le style qu'aux frères suivants qui utilisent la classe `info`. Dans l'extrait précédent, comme aucun élément n'utilise cette classe, l'instruction n'apporte aucune modification :

```
$('#nouveau').nextAll('.info').css('font-weight', 'bold')
```

Examinons le cas de l'extrait suivant, où un élément porte l'identifiant `nouveau` et un autre l'identifiant `ancien` :

```
<ul>
  <li             >Élément 1</li>
  <li id='nouveau'>Élément 2</li>
  <li             >Élément 3</li>
  <li  id='ancien'>Élément 4</li>
  <li             >Élément 5</li>
</ul>
```

À partir de là, il devient possible de sélectionner seulement les frères qui suivent celui avec l'identifiant nouveau mais seulement jusqu'à celui qui porte l'identifiant ancien, celui-ci non compris, comme ceci (seul le troisième élément reçoit le style) :

```
$('#nouveau').nextUntil('#ancien').css('font-weight', 'bold')
```

Si vous ne fournissez aucun argument à `nextUntil`, cette méthode se comporte exactement comme `nextAll` et renvoie tous les frères suivants. À l'inverse, vous pouvez fournir à `nextUntil` un second argument pour constituer un filtre à partir des éléments qu'elle identifie, comme suit :

```
$('#nouveau').nextUntil('#ancien', '.info').css('font-weight', 'bold')
```

Cette instruction ne modifie le style que des éléments qui portent la classe `info`, soit aucun de ceux de l'extrait de code HTML précédent car aucun `<li  class="info">` n'existe dans la liste entre les éléments d'identifiants nouveau et ancien.

Les méthodes `prev`, `prevAll` et `prevUntil` effectuent les mêmes tâches que les précédentes, mais auscultent les éléments frères précédents.

Parcourir les sélections de jQuery

De la même manière que fonctionne le parcours du DOM, dès que vous obtenez en retour un ensemble d'éléments dans une sélection jQuery, vous pouvez ensuite parcourir ces éléments et ne choisir que ceux sur lesquels agir.

Ainsi, pour modifier le style du premier élément renvoyé par une sélection, utilisez la méthode `first`, comme suit (pour afficher en souligné le premier élément de liste de la première liste non numérotée) :

```
$('ul>li').first().css('text-decoration', 'underline')
```

Pour modifier le style du seul dernier élément, utilisez la méthode `last`, comme ceci :

```
$('ul>li').last().css('font-style', 'italic')
```

Mieux encore, pour accéder à un élément par un indice (débutant à 0), utilisez la méthode `eq`, comme suit (qui modifie le style du deuxième élément de la liste en partant de 0) :

```
$('ul>li').eq(1).css('font-weight', 'bold')
```

Vous pouvez aussi appliquer un filtre à une sélection à l'aide de la méthode `filter`, comme ceci (qui change en cyan la couleur d'un élément sur deux en commençant par le premier, l'élément 0) :

```
$('ul>li').filter(':even').css('background', 'cyan')
```

Retenez que, dans les indices des sélections jQuery, le premier élément est celui d'indice zéro. Donc, quand vous utilisez le sélecteur `:even` (qui signifie pair) de cette manière, les éléments sélectionnés sont ceux d'indices 1, 3, 5 et ainsi de suite, et non ceux d'indices 2, 4, 6, etc.

Appliquez la méthode `not` pour exclure un ou plusieurs éléments. L'exemple suivant change en bleu la couleur des éléments qui *ne* portent *pas* l'identifiant `nouveau` :

```
$('ul>li').not('#nouveau').css('color', 'blue')
```

Enfin, vous pouvez sélectionner un élément selon les descendants qu'il possède. Pour sélectionner seulement les éléments qui possèdent (*has*) un ou des éléments descendants `<ol>`, par exemple, utilisez l'instruction suivante, qui barre (*line through*) ceux identifiés :

```
$('ul>li').has('ol').css('text-decoration', 'line-through')
```

L'exemple 21-23 rassemble quelques-unes de ces instructions pour appliquer des styles à une liste non numérotée, dont une des lignes contient aussi une liste numérotée.

Exemple 21-23. Parcours d'une sélection jQuery

```
<!DOCTYPE html>
<html>
  <head>
    <title>Parcours de sélection</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <ul>
      <li>Élément 1</li>
      <li>Élément 2</li>
```

```
      <li id='nouveau'>Élément 3</li>
      <li>Élément 4
        <ol type='a'>
          <li>Élément 4a</li>
          <li>Élément 4b</li>
        </ol></li>
      <li>Élément 5</li>
    </ul>
    <script>
      $('ul>li').first()         .css('text-decoration', 'underline')
      $('ul>li').last()          .css('font-style',      'italic')
      $('ul>li').eq(1)           .css('font-weight',     'bold')
      $('ul>li').filter(':even') .css('background',      'cyan')
      $('ul>li').not('#nouveau').css('color',           'blue')
      $('ul>li').has('ol')       .css('text-decoration', 'line-through')
    </script>
  </body>
</html>
```

La figure 21-21 illustre que chaque élément de chaque liste a reçu un ou plusieurs styles de la part des instructions jQuery.

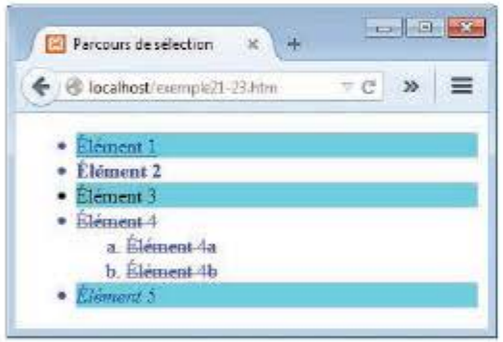


Figure 21-21. Ciblage sélectif d'éléments dans une sélection jQuery

Méthode is

Un autre moyen permet d'interroger un sélecteur jQuery pour obtenir une valeur booléenne à utiliser dans du code JavaScript pur : la méthode `is` (est-ce ?). Au contraire des autres méthodes de filtrage de jQuery, cette fonction ne crée aucun nouvel objet jQuery auquel ajouter des méthodes ou que l'on puisse ensuite filtrer plus profondément.

Elle ne renvoie que `true` ou `false`, ce qui la destine idéalement à des instructions conditionnelles. L'exemple 21-24 exploite la méthode `is` associée à un appel à `parent` dans un gestionnaire d'évènement pour un ensemble de boutons. Lorsqu'un des boutons reçoit un clic, le gestionnaire correspondant est appelé et la méthode `is` renvoie une valeur `true` ou `false`, selon que l'élément parent est une section `<div>` ou non (figure 21-22).

Exemple 21-24. Information sur l'élément parent à l'aide de la méthode is

```
<!DOCTYPE html>
<html>
  <head>
    <title>Utilisation d'is</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <div><button>Bouton dans div</button></div>
    <div><button>Bouton dans div</button></div>
    <span><button>Bouton dans span</button></span>
    <div><button>Bouton dans div</button></div>
    <span><button>Bouton dans span</button></span>
    <p id='info'></p>
    <script>
      $('button').click(function()
      {
        var elem = ''

        if ($(this).parent().is('div')) elem = 'div'
        else                            elem = 'span'

        $('#info').html('Vous avez cliqué dans une section ' + elem)
      })
    </script>
  </body>
</html>
```

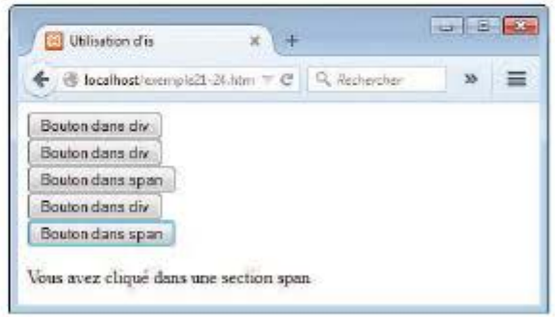


Figure 21-22. La méthode is détermine l'élément parent

Utiliser jQuery sans les sélecteurs

Deux autres méthodes de jQuery sont aussi prévues pour une utilisation avec les objets JavaScript standards afin de simplifier leur gestion. Ce sont les méthodes `$.each` et `$.map` qui s'apparentent fort l'une à l'autre, avec quelques subtiles différences.

Méthode $.each

La méthode `$.each` permet d'itérer au sein de tableaux et d'objets qui ressemblent à des tableaux, par association d'une fonction à appeler à chaque itération. L'exemple 21-25 montre un tableau de noms d'animaux de compagnie et leurs types (appelé `animaux`), dont il faut extraire un autre tableau (`cobayes`), pour ne contenir que les noms des cobayes.

Exemple 21-25. Appel de la méthode each

```
<!DOCTYPE html>
<html>
  <head>
    <title>Utilisation d'each et map</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body>
    <div id='info'></div>
    <script>
      animaux =
      {
        Gratgrat   : 'cobaye',
        Pirlouit   : 'cobaye',
        Gribouille : 'lapin',
        Atchoum    : 'lapin',
        Médor      : 'chien',
        Félix      : 'chat'
      }

      cobayes = []

      $.each(animaux, function(nom, type)
      {
        if (type == 'cobaye') cobayes.push(nom)
      })

      $('#info').html('Les noms des cobayes sont : ' + cobayes.join(' et '))
    </script>
  </body>
</html>
```

Pour y parvenir, la méthode `$.each` reçoit le tableau, ainsi qu'une fonction anonyme pour le traiter. La fonction prend deux paramètres : la clé dans le tableau (`nom`) et le contenu de chaque élément (`type`).

Dans la fonction anonyme, nous testons la valeur de `type` pour vérifier s'il s'agit d'un `cobaye` et, si c'est le cas, nous « poussons » (*push*) la valeur de `nom` dans le tableau `cobayes`. Lorsque le balayage du tableau se termine, nous affichons le contenu de `cobayes` dans l'élément `<div>` défini plus haut avec l'identifiant `info`. Pour séparer les éléments du

tableau lors de l'affichage, la méthode `join` permet de fusionner les deux entrées avec un `et` de séparation. Le résultat de cet exemple dans un navigateur produit simplement l'affichage du texte « Les noms des cobayes sont : Gratgrat et Pirlouit ».

Méthode $.map

L'autre méthode pour réaliser cela s'appelle `$.map`, qui renvoie dans un tableau toutes les valeurs que votre fonction retourne, ce qui vous épargne la nécessité de créer un tableau et d'y pousser les correspondances identifiées, comme dans le cas de l'exemple précédent.

Vous pouvez aussi créer et remplir le tableau en une seule opération. Pour cela, il suffit de lui affecter le tableau renvoyé par `$.map`, comme suit (le résultat final est identique au précédent mais avec moins de code) :

```
cobayes = $.map(animaux, function(type, nom)
{
  if (type == 'cobaye') return nom
})
```

Soyez toutefois très attentif lorsque vous échangez les méthodes `$.each` et `$.map` parce que `$.each` attend que les arguments de la fonction se suivent dans l'ordre *clé*, *valeur*, tandis que `$.map` les attend dans l'ordre inverse, *valeur*, *clé*. C'est la raison qui explique que les arguments soient intervertis dans l'exemple précédent.

Tirer parti d'Ajax

Le chapitre 17 montrait en détail comment implémenter des communications Ajax entre JavaScript dans un navigateur et PHP fonctionnant sur un serveur web. Il proposait aussi des fonctions pratiques et compactes pour vous simplifier le traitement.

Cependant, si jQuery est chargé, vous pouvez faire appel à ses fonctionnalités Ajax si vous le préférez. Elles opèrent d'une manière très semblable dans la mesure où vous choisissez de lancer une demande Post ou Get, puis vous poursuivez selon ce principe.

Utiliser la méthode Post

L'exemple 21-26 constitue l'équivalent jQuery direct à l'exemple 17-2 (qui charge le site web Amazon Mobile dans un élément `<div>`) mais, comme tout le code de traitement Ajax est relégué au fichier de bibliothèque jQuery, il s'avère bien plus court, avec un simple appel à la méthode `$.post`, ainsi que le passage des trois éléments suivants :

- l'URL du programme PHP auquel il faut accéder sur le serveur ;
- les données à passer à cette URL ;
- une fonction anonyme pour traiter les données retournées.

Exemple 21-26. Envoi d'une demande Ajax Post avec jQuery

```
<!DOCTYPE html>
<html> <!-- jqueryajaxpost.htm -->
  <head>
    <title>Ajax Post en jQuery</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body style='text-align:center'>
    <h1>Chargement d'une page web dans une DIV</h1>
    <div id='info'>Cette phrase sera remplacée</div>

    <script>
      $.post('urlpost.php', { url : 'amazon.com/gp/aw' }, function(data)
      {
        $('#info').html(data)
      } )
    </script>
  </body>
</html>
```

Le programme ***urlpost.php*** demeure identique à celui de l'exemple 17-3, parce que cet exemple et l'exemple 17-2 sont tout à fait interchangeables.

Utiliser la méthode Get

La communication en Ajax à l'aide de la méthode Get est tout aussi simple et n'exige que les deux arguments suivants, de sorte que l'exemple 21-27 est l'équivalent jQuery de l'exemple 17-4 :

- l'URL du programme PHP auquel il faut accéder sur le serveur (y compris une chaine de requête avec les données à lui transmettre) ;
- une fonction anonyme pour traiter les données retournées.

Exemple 21-27. Envoi d'une demande Ajax Get avec jQuery

```
<!DOCTYPE html>
<html> <!-- jqueryajaxget.htm -->
  <head>
    <title>Ajax Get en jQuery</title>
    <script src='jquery-1.11.2.min.js'></script>
  </head>
  <body style='text-align:center'>
    <h1>Chargement d'une page web dans une DIV</h1>
    <div id='info'>Cette phrase sera remplacée</div>

    <script>
      $.get('urlget.php?url=amazon.com/gp/aw', function(data)
      {
```



```
        $('#info').html(data)
      } )
    </script>
  </body>
</html>
```

Le programme ***urlget.php*** demeure identique à celui de l'exemple 17-5, parce que cet exemple et l'exemple 17-4 sont tout à fait interchangeables.

Rappelez-vous que les restrictions de sécurité d'Ajax imposent que les communications aient lieu avec le même serveur que celui qui a fourni le document web principal. Vous devez aussi utiliser un authentique serveur web pour les communications Ajax, parce qu'elles ne fonctionnent pas avec un système de fichiers locaux. Ces exemples fonctionnent donc mieux sur un serveur web de développement ou de production comme ceux décrits au chapitre 2.

Modules d'extension

Il n'y a de place dans ce livre que pour aborder la bibliothèque de base de jQuery (*core library*) et, si cela représente déjà plus que le nécessaire pour aider un débutant à la prendre en main, viendra un moment où vous aurez besoin de plus de fonctionnalités et de possibilités. Heureusement, d'autres projets jQuery peuvent vous aider en ce sens, parce qu'il existe désormais toute une gamme de plugiciels et modules d'extension officiels ou de tierces parties pour vous apporter à peu près toutes les possibilités que vous pouvez imaginer.

Interface utilisateur de jQuery

D'abord, il y a l'interface utilisateur de JQuery, connue sous le nom jQuery UI, qui prend le relais là où la portée de jQuery s'arrête. Elle permet d'ajouter du glisser et déposer, des redimensionnements, des méthodes de tris de vos pages web, ainsi que d'autres animations et effets, des transitions de couleurs animées, des effets de fondus entrants et sortants, en plus d'une pléthore de widgets pour créer des menus et d'autres fonctionnalités comme des accordéons (*accordions*), des boutons, des sélecteurs de dates et de couleurs, des barres de progression, des curseurs (*sliders*), des toupies (*spinners*), des onglets, des infobulles et bien d'autres encore.

Pour voir les démonstrations avant de choisir quelles options télécharger, consultez *jqueryui.com/demos*.

Le paquet complet pèse moins de 400 Ko sous forme compressée. Son téléchargement est gratuit et son utilisation est presque sans restriction (sous la licence MIT très généreuse), sur *jqueryui.com*.

Autres modules d'extension

Des plugiciels (*plug-ins*) d'une grande variété, gratuits, prêts à l'emploi sont également disponibles, proposés par de nombreux développeurs, et tous réunis sur *plugins.jquery.com*.

Parmi ces modules d'extension, vous trouverez des outils de gestion et de vérification complète de formulaires, des diaporamas, de l'interaction avec l'utilisateur, de la manipulation d'image, des animations supplémentaires et bien d'autres.

jQuery Mobile

Lorsque vous développez pour des navigateurs mobiles, examinez aussi jQuery Mobile, qui forme plutôt un environnement de développement (*framework*) qu'une bibliothèque et fournit des moyens sophistiqués et optimisés pour le toucher et les gestes, pour naviguer sur la vaste gamme de types différents de matériels et logiciels pour appareils mobiles, et offrir la meilleure expérience possible à l'utilisateur.

Le téléchargement de jQuery Mobile autorise une personnalisation complète et une taille sur mesure adaptées exactement à vos attentes, comme l'illustre la figure 21-23, qui montre l'application (ou *appli*) ThemeRoller en cours d'utilisation.

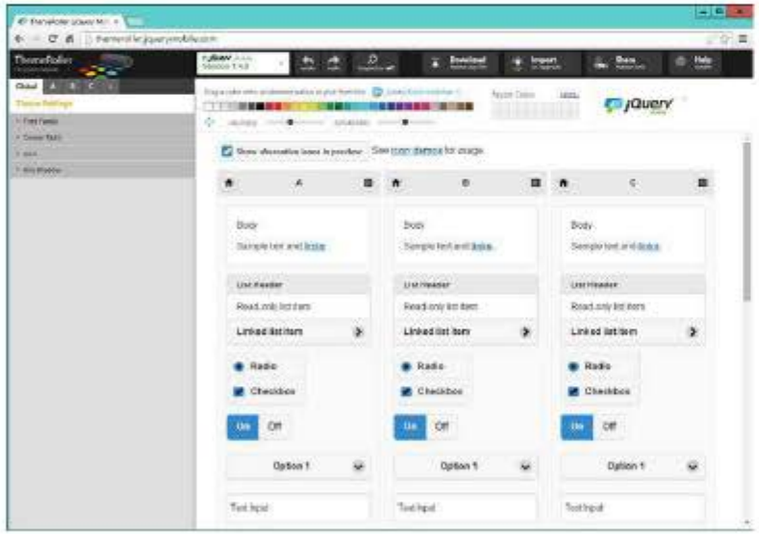


Figure 21-23. Création d'un thème pour appareil mobile à l'aide de l'appli ThemeRoller

jQuery mobile s'avère particulièrement utile pour écrire des applications web et vous trouverez encore bien d'autres outils dans ce contexte, à télécharger sur *jquerymobile.com*.

Ce chapitre vous a emmené sur une longue route, pour vous présenter des sujets qui, pour certains, méritent des livres entiers à eux seuls. J'espère avoir été suffisamment clair dans les explications et que je vous aurai convaincu que jQuery est facile à apprendre et à utiliser. Je vous propose de réserver un moment pour examiner l'annexe E, qui énumère tous les principaux objets, évènements et méthodes de jQuery, et destinée à vous servir de référence. Si vous souhaitez de plus amples informations sur ce sujet et sur d'autres que nous n'avons pas évoqués, reportez-vous à *jquery.com*.

Dans les chapitres restants, nous tournons le regard vers toutes ces nouvelles fonctionnalités disponibles en HTML5 pour ensuite rassembler tout ce que nous avons appris dans un mini-projet de réseau social.

Questions

1. Quel est le symbole communément utilisé comme méthode de fabrication (*factory*) pour créer des objets jQuery et quel est le nom de la méthode alternative ?
2. Que faut-il faire pour établir un lien dans une page web à la version minimisée 1.11.2 de jQuery à partir du RDC (ou CDN) de Google ?
3. Quels types d'arguments la méthode de fabrication de jQuery accepte-t-elle ?
4. Avec quelle méthode jQuery pouvez-vous lire ou définir la valeur d'une propriété CSS ?
5. Quelle instruction écrivez-vous pour associer une méthode à l'évènement `click` d'un élément d'identifiant `elem`, pour le masquer lentement ?
6. Quelle propriété d'un élément devez-vous modifier pour permettre son animation et quelles en sont les valeurs possibles ?
7. Comment peut-on provoquer l'exécution immédiate (ou séquentielle dans le cas des animations) de plusieurs méthodes ?
8. Comment faites-vous pour retrouver un objet nœud d'élément à partir d'un objet de sélection jQuery ?
9. Quelle instruction écrivez-vous pour imposer l'affichage en gras de l'élément frère qui précède immédiatement celui d'identifiant `nouvelles` ?
10. Quelle méthode permet d'effectuer une demande Ajax Get en jQuery ?

Retrouvez les réponses du chapitre 21 dans l'annexe A.

CHAPITRE 22

Introduction à HTML5

HTML5 représente un fabuleux pas en avant dans la conception, le design, la disposition et les possibilités d'utilisation du web. Il fournit un moyen simple de manipuler des graphismes dans un navigateur web sans dépendre de modules d'extension ou de plugiciels tels que Flash, offre des méthodes pour insérer de l'audio et de la vidéo dans des pages web, également sans modules externes, et règle le compte de plusieurs incohérences gênantes qui se sont accumulées dans HTML au cours de son évolution.

De plus, HTML5 intègre nombre d'améliorations supplémentaires, comme la gestion de la géolocalisation, les processus de traitement web pour gérer des tâches à l'arrière-plan, une gestion améliorée des formulaires, un accès à de nouveaux faisceaux de stockage local (qui dépassent de loin les capacités limitées des cookies), et même la possibilité de transformer des pages web en applications web pour les navigateurs mobiles.

Ce qu'il faut toutefois noter à propos de HTML5, c'est qu'il est en évolution permanente et que les navigateurs ont adopté des fonctionnalités différentes à des moments différents, et non tous ensemble. Heureusement, les plus populaires et les plus belles additions de HTML5 ont été finalement adoptées par tous les principaux navigateurs (ceux qui représentent plus d'un pourcent du marché, comme Chrome, Internet Explorer, Firefox, Safari, Opera et les navigateurs d'Android et d'iOS).

Mais, malgré que HTML5 ait été officiellement soumis pour approbation auprès du W3C au début 2013, un certain nombre de fonctionnalités manquent encore dans plusieurs navigateurs. Je les cite plus loin dans le livre pour que vous vous prépariez à les exploiter lorsqu'ils les auront, tous, toutes adoptées.

Quoi qu'il en soit, nous sommes aujourd'hui devant la deuxième grande étape vers l'interactivité web dynamique, la première représentant l'adoption de ce que l'on appelle désormais le Web 2.0. J'hésite cependant à appeler cette nouvelle évolution le Web 3.0, parce que le terme HTML5 dit déjà tout à la plupart des gens et, selon mon point de vue, il faudrait plutôt le considérer comme une version récente du Web 2.0, quelque chose du genre Web 2.7.

À vrai dire, j'éprouve beaucoup d'intérêt de voir ce à quoi le Web 3.0 aboutira. Si je devais me livrer à quelques hasardeuses prédictions, je dirais qu'il résulterait de l'application de l'intelligence artificielle (IA), sous la forme de versions de logiciels aux possibilités plus étendues, comme Siri d'Apple, Cortana de Microsoft, OK Google et Watson

d'IBM, combinées à des technologies portables au quotidien, qui empruntent des entrées visuelles et vocales (comme les Google Glass et la montre Galaxy Gear), plutôt qu'au clavier. Je serais ravi de couvrir ces sujets dans des éditions ultérieures de ce livre.

Pour l'heure, alors que j'ai déjà beaucoup écrit à propos de ce que HTML5 apportera sous quelques années et que de nombreuses parties des spécifications sont désormais utilisables sur à peu près tous les appareils et navigateurs, je suis ravi de les intégrer à cette édition du livre. Donc, permettez-moi de vous présenter un aperçu de ce qui est disponible dès maintenant dans HTML5.

Canevas

Introduit à l'origine par Apple dans le moteur de rendu WebKit (issu lui-même du moteur de disposition HTML de KDE) de son navigateur Safari (et désormais implémenté dans iOS, Android, Kindle, Chrome, BlackBerry, Orera et Tizen), l'élément canevas (*canvas*) permet de dessiner des graphismes dans une page web sans devoir compter sur des modules d'extension tels que Java ou Flash. Après une normalisation, le canevas a été adopté par tous les autres navigateurs et il fait aujourd'hui figure de pilier du développement web moderne.

Comme tous les autres éléments HTML, le canevas est simplement un élément présent dans une page web et affublé de dimensions déterminées, dans lequel vous utilisez du JavaScript pour dessiner des graphismes. Pour créer un canevas, insérez la balise HTML `<canvas>`, à laquelle vous devez aussi affecter un identifiant pour que JavaScript sache à quel canevas il doit accéder (puisqu'une page web peut contenir plus d'un canevas).

Dans la suite de ce chapitre, nous utiliserons indifféremment canevas et *canvas*, le nom anglais de l'élément pour le désigner, mais nous privilégierons `canvas` dans le code.

L'exemple 22-1 crée un élément `<canvas>` d'identifiant `moncanvas`, qui contient du texte que seuls affichent les navigateurs qui ne prennent pas en charge le canevas.

Exemple 22-1. Utilisation de l'élément canvas de HTML5

```
<!DOCTYPE html>
<html>
  <head>
    <title>Canevas de HTML5</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <canvas id='moncanvas' width='320' height='240'>
      Ceci est un élément canvas d'identifiant <i>moncanvas</i>.
      Ce texte n'est visible que dans les navigateurs non HTML5.
    </canvas>
    <script>
      canvas            = O('moncanvas')
      context           = canvas.getContext('2d')
      context.fillStyle = 'red'
      S(canvas).border  = '1px solid black'
      context.beginPath()
      context.moveTo(160, 120)
      context.arc(160, 120, 70, 0, Math.PI * 2, false)
      context.closePath()
      context.fill()
    </script>
  </body>
</html>
```

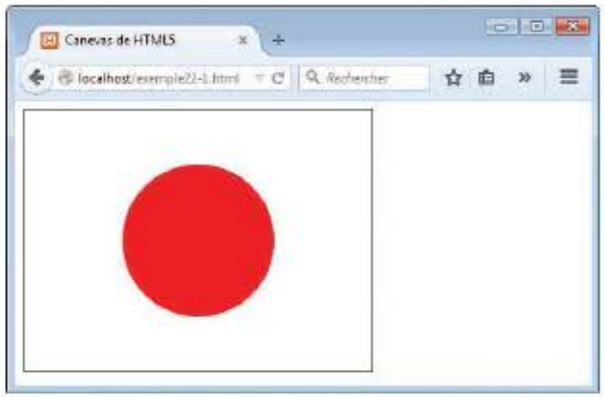


Figure 22-1. Tracé du drapeau japonais à l'aide d'un canevas HTML5

À ce stade, il n'est pas nécessaire de détailler exactement le fonctionnement du code car le chapitre suivant reprend ces détails. Au moins, vous constatez que l'utilisation du canevas n'est pas si complexe, même si elle nécessite cependant l'apprentissage de quelques fonctions JavaScript. Remarquez en passant que cet exemple emprunte l'ensemble de fonctions *OSC.js* du chapitre 20 pour conserver au code un aspect compact.

Géolocalisation

Grâce à la géolocalisation, votre navigateur peut renvoyer une information au serveur web sur l'endroit où vous êtes. Cette information provient de la puce GPS de votre ordinateur ou de votre appareil mobile, de votre adresse IP ou d'une analyse des points d'accès sans fil proches. Pour des raisons de sécurité et de respect de la vie privée, l'utilisateur a toujours le droit de refuser de diffuser cette information au cas par cas ou peut régler des paramètres pour autoriser ou interdire l'accès à cette donnée par un site web ou tous les sites.

Cette technologie offre de nombreuses possibilités, comme l'aide à la navigation routière, la fourniture de cartes géographiques locales, l'information sur les restaurants proches, les points d'accès Wifi publics, les stations-services, vos amis à proximité et bien d'autres.

L'exemple 22-2 affiche une carte Google Maps de l'emplacement de l'utilisateur, à condition que le navigateur prenne en charge la géolocalisation et que l'utilisateur autorise

la diffusion de son lieu géographique (figure 22-2). Sinon, il affiche une erreur.

Exemple 22-2. Affichage d'une carte du lieu de l'utilisateur

```
<!DOCTYPE html>
<html>
  <head>
    <title>Exemple de géolocalisation</title>
    <script src='OSC.js'></script>
    <script src="https://maps.googleapis.com/maps/api/js?sensor=false"></script>
  </head>
  <body>
    <div id='status'></div>
    <div id='map'></div>

    <script>
      if (typeof navigator.geolocation == 'undefined')
         alert("Géolocalisation non prise en charge.")
      else
        navigator.geolocation.getCurrentPosition(autorise, refuse)

      function autorise(position)
      {
        O('status').innerHTML = 'Permission accordée'
        S('map').border       = '1px solid black'
        S('map').width        = '640px'
        S('map').height       = '320px'

        var lat   = position.coords.latitude
        var long  = position.coords.longitude
        var gmap  = O('map')
        var gopts =
        {
          center: new google.maps.LatLng(lat, long),
          zoom: 9, mapTypeId: google.maps.MapTypeId.ROADMAP
        }
        var map = new google.maps.Map(gmap, gopts)
      }

      function refuse(error)
      {
        var message

        switch(error.code)
        {
          case 1: message = 'Permission refusée' ; break;
          case 2: message = 'Position non disponible' ; break;
          case 3: message = 'Dépassement de délai' ; break;
          case 4: message = 'Erreur inconnue' ; break;
        }
```



```
        O('status').innerHTML = message
      }
    </script>
  </body>
</html>
```

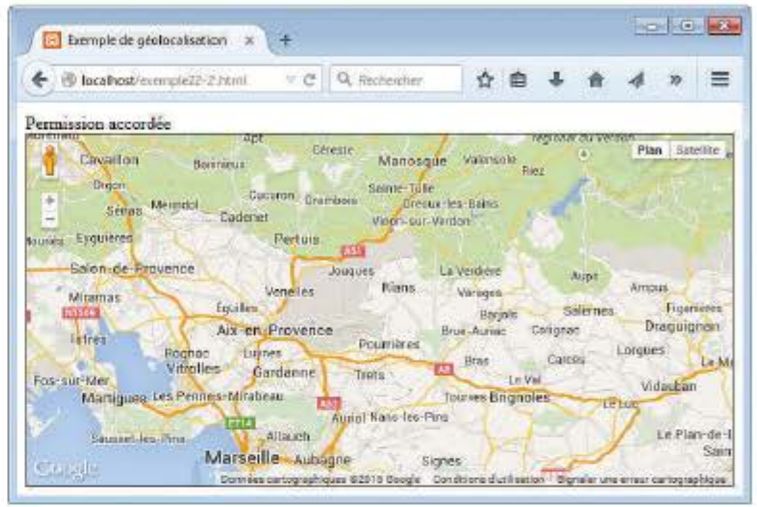


Figure 22-2. Emplacement de l'utilisateur exploité pour afficher une carte des lieux

Une fois encore, il n'y a pas de place ici pour décrire le fonctionnement dans la mesure où nous y reviendrons au chapitre 25. Pour l'instant, cet exemple montre la facilité de tirer parti de la géolocalisation, d'autant plus que la majorité du code sert à gérer les erreurs et à appeler la carte Google, donc le code de géolocalisation proprement dit est plutôt réduit.

Audio et vidéo

HTML5 apporte aussi la prise en charge d'audio et de vidéo au sein même du navigateur. Si la reproduction de ce genre de multimédia peut s'avérer un peu complexe du fait de la variété des types de codage et des licences, les éléments `<audio>` et `<video>` offrent toute la souplesse nécessaire pour afficher le multimédia dont vous disposez.

Dans l'exemple 22-3, le même fichier vidéo a été encodé sous différents formats pour permettre à tous les principaux navigateurs de l'afficher. Le navigateur sélectionne simplement le premier type qu'il reconnait et l'affiche, comme illustré à la figure 22-3.

Exemple 22-3. Affichage d'une vidéo en HTML5

```
<!DOCTYPE html>
<html>
  <head>
    <title>Vidéo en HTML5</title>
  </head>
  <body>
    <video width='560' height='320' controls>
      <source src='film.mp4'  type='video/mp4'>
      <source src='film.webm' type='video/webm'>
      <source src='film.ogv'  type='video/ogg'>
    </video>
  </body>
</html>
```

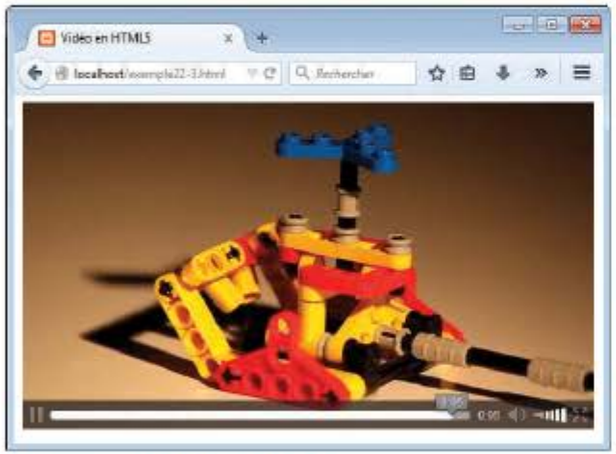


Figure 22-3. Affichage d'une vidéo en HTML5

L'insertion d'audio dans une page web est tout aussi aisée, comme vous le verrez au chapitre 24.

Formulaires

Au chapitre 12, vous avez vu que les formulaires sont en cours d'amélioration mais leur prise en charge parmi les navigateurs s'avère encore un peu parcellaire et hétérogène.

Le chapitre 12 indique les détails de ce que vous pouvez utiliser actuellement et des éditions à venir de ce livre reprendront d'autres aspects des formulaires à mesure qu'ils seront adoptés par tous. Entretemps, vous pouvez vous tenir au courant des derniers développements sur le site (en anglais) *http://tinyurl.com/h5forms*.

Stockage local

Avec le stockage local, votre capacité à enregistrer des données sur un périphérique local croît sensiblement par rapport aux maigres possibilités offertes par les cookies. Ceci ouvre la voie à toutes sortes d'applications, comme un utilitaire web pour travailler sur des documents hors ligne, puis les synchroniser avec le serveur web dès que la connexion à l'internet est rétablie. La perspective se manifeste également de stocker de petites bases de données localement pour conserver une copie de votre collection de musiques ou toutes vos statistiques personnelles dans le cadre d'un plan diététique ou de performances physiques, par exemple. Le chapitre 25 montre comment tirer le meilleur parti de cette fonctionnalité dans des projets web.

Traitement web : déléguez !

La possibilité d'exécuter en coulisses des applications pilotées par des interruptions en JavaScript existe déjà depuis des années, mais il s'agit d'un processus lourd et inefficace. Il serait de loin préférable de laisser la technologie sous-jacente du navigateur exécuter des tâches secondaires à votre place, ce que le navigateur peut réaliser beaucoup plus vite que vous, avec vos interruptions continuelles du navigateur pour vérifier comment cela se passe.

À la place, avec les traitements web (*web workers*), vous paramétrez tout et passer votre code au navigateur web, qui l'exécute ensuite. Lorsque quelque chose de notable se produit, votre code en avertit le navigateur, qui à son tour avertit votre code principal. Entretemps, votre page peut ne rien faire ou exécuter d'autres tâches, et vous oubliez totalement la tâche secondaire tant qu'elle ne se manifeste pas à votre attention.

Le chapitre 25 montre comment utiliser les traitements web pour créer une horloge simple et pour calculer des nombres premiers.

Applications web

De nos jours, les pages web ressemblent de plus en plus à des applications autonomes et, avec HTML5, elles deviennent très facilement des « *web apps* ». Tout ce que vous devez faire, c'est indiquer au navigateur web les ressources utilisées par votre application et il s'occupe de les télécharger dans un emplacement où il peut les exécuter et y accéder localement, hors ligne, et sans connexion à l'internet, si nécessaire.

Le chapitre 25 montre comment réaliser cela avec un exemple de conversion de l'horloge de la section sur les traitements web en une application web.

Microdonnées

Le chapitre 25 montre également comment marquer votre code à l'aide de microdata pour le rendre totalement compréhensible par n'importe quel navigateur ou autre technologie qui doit y accéder. Les microdonnées promettent de prendre une importance de plus en plus grande dans le cadre de l'optimisation pour les moteurs de recherche donc il est important que vous commenciez à les incorporer dans vos pages ou, tout au moins, que vous compreniez quelles informations elles peuvent apporter à vos sites web.

En résumé

Vous pouvez constater qu'il y a beaucoup à dire à propos de HTML5 et de ses fonctionnalités que de nombreuses personnes attendaient depuis longtemps. Elles sont enfin là. À commencer par le canevas, les quelques chapitres suivants expliquent ces fonctionnalités avec force de détails, pour que vous puissiez les apprendre, les exécuter et améliorer vos sites web en un minimum de temps.

Questions

1. Quel nouvel élément de HTML5 permet de dessiner des graphismes directement dans une page web ?
2. Quel langage de programmation faut-il utiliser pour accéder à nombre de fonctionnalités avancées de HTML ?
3. Quelles balises utilise-t-on pour intégrer de l'audio et de la vidéo dans une page web ?
4. Quelle fonctionnalité nouvelle de HTML5 offre de plus grandes possibilités que les cookies ?
5. Quelle technologie de HTML5 permet de prendre en charge des tâches secondaires en JavaScript ?

Retrouvez les réponses du chapitre 22 dans l'annexe A.

CHAPITRE 23

Le canevas de HTML5

Si HTML5 constitue le terme collectif donné aux nouvelles technologies du web, il ne s'agit pas simplement de balises et de propriétés HTML. C'est notamment le cas de l'élément canevas. Certes, vous créez un canevas à l'aide de la balise `<canvas>` et vous y précisez une largeur et une hauteur, puis vous le modifiez un peu en CSS, mais pour écrire ou lire réellement dans le canevas, vous devez emprunter du JavaScript.

Heureusement, le JavaScript nécessaire à apprendre demeure réduit et vraiment simple à mettre en œuvre, d'autant que nous disposons déjà d'un ensemble de trois fonctions prêtes à l'emploi depuis le chapitre 21 (dans le fichier *OSC.js*), qui rendent encore plus trivial et direct l'accès aux objets tels que le canevas. Dès lors, nous pouvons plonger au cœur du sujet et commencer à utiliser la balise `<canvas>`.

Créer un canevas et y accéder

Le chapitre précédent montrait comment dessiner un cercle simple pour afficher le drapeau japonais, comme dans l'exemple 23-1. Voyons à présent en détail comment les choses se passent.

Exemple 23-1. Dessin du drapeau japonais à l'aide d'un canevas

```
<!DOCTYPE html>
<html>
  <head>
    <title>Canevas de HTML5</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <canvas id='noncanvas' width='320' height='240'>
      Ceci est un élément canvas d'identifiant <i>noncanvas</i>.
      Ce texte n'est visible que dans les navigateurs non HTML5.
    </canvas>
```

```
    <script>
      canvas          = O('noncanvas')
      context         = canvas.getContext('2d')
      context.fillStyle = 'red'
      S(canvas).border = '1px solid black'

      context.beginPath()
      context.moveTo(160, 120)
      context.arc(160, 120, 70, 0, Math.PI * 2, false)
      context.closePath()
      context.fill()
    </script>
  </body>
</html>
```

D'abord, la déclaration `<!DOCTYPE html>` indique au navigateur que le document utilise du HTML5. Ensuite, nous définissons le titre à afficher et nous chargeons les trois fonctions du fichier *OSC.js*.

Dans le corps du document, nous définissons un élément `canvas` et lui donnons l'identifiant `noncanvas`, avec une largeur de 320 pixels et une hauteur de 240 pixels.

Une section de JavaScript suit, pour définir le style et dessiner sur le `canevas`. Pour créer l'objet `canvas`, nous appelons la fonction `O` sur l'élément canvas. Rappelez-vous que cette fonction appelle en réalité la fonction `document.getElementById` et que cela ne représente qu'une écriture abrégée pour faire référence à l'élément.

Vous avez déjà vu tout ce qui précède mais ce qui suit est nouveau :

```
context = canvas.getContext('2d')
```

Cette commande appelle la méthode `getContext` de l'objet `canvas` créé à l'instant, pour lui demander un accès en deux dimensions, par le passage de l'argument `'2d'`.

Vous aurez compris que des plans existent pour donner un contexte tridimensionnel au canevas (probablement bâti sur l'API OpenGL ES), qui autoriseront la prise en charge de l'argument `'3d'`. Pour l'instant, si vous souhaitez dessiner en trois dimensions dans un canevas, vous devez encore vous servir des mathématiques et les simuler en deux dimensions. Vous pouvez aussi investiguer du côté de WebGL (bâti sur OpenGL ES). Il n'y a malheureusement pas assez de place ici pour couvrir ce sujet, mais vous trouverez un excellent tutoriel (en anglais) sur *http://learningwebgl.com*.

Équipés de ce contexte dans l'objet `context`, nous préparons les commandes de dessin suivantes par un réglage de la propriété `fillStyle` (style de remplissage) du contexte à la valeur `'red'`, rouge :

```
context.fillStyle = 'red'
```

Un appel à la fonction `S` règle ensuite la propriété de bordure du canevas à un trait plein noir de un pixel pour entourer l'image du drapeau :


```
S(canvas).border = '1px solid black'
```

Ces préparations terminées, nous ouvrons un chemin (*path*) dans le contexte et déplaçons la position du début du dessin à l'emplacement 160,120 :

```
context.beginPath()
context.moveTo(160, 120)
```

Ensuite, nous dessinons un arc de cercle centré sur ces coordonnées, avec un rayon de 70 pixels, débutant à l'angle de 0 degré (qui correspond à l'extrémité droite du cercle lorsqu'il est dessiné) et nous poursuivons le long du cercle complet, avec un angle exprimé en radians d'une valeur déterminée par $2 \times \pi$:

```
context.arc(160, 120, 70, 0, Math.PI * 2, false)
```

La valeur `false` finale indique de tracer le cercle dans le sens des aiguilles d'une montre ; la valeur `true` indiquerait le sens contraire des aiguilles d'une montre.

Enfin, nous fermons le chemin de tracé et imposons le remplissage à l'aide de la valeur présélectionnée `'red'` de la propriété `fillStyle`, quelques lignes plus haut :

```
context.closePath()
context.fill()
```

Le résultat du chargement de ce code dans un navigateur est illustré par la figure 22-1 du chapitre précédent.

Fonction toDataURL

Lorsque vous avez créé une image dans un canevas, vous souhaitez parfois en créer une copie, par exemple pour la reproduire ailleurs dans une page web, pour l'enregistrer dans l'espace de stockage local ou pour la téléverser sur un serveur web. Ceci est d'autant plus intéressant que les utilisateurs ne peuvent emprunter le glisser-déposer pour enregistrer une image de canevas.

Pour illustrer le procédé, l'exemple 23-2 ajoute quelques lignes de code par rapport au précédent (surlignées en gras). Elles créent un élément `<img>` d'identifiant `'monimage'`, lui donnent une bordure pleine en noir et copient l'image du canevas dans l'image `<img>` (figure 23-1).

Exemple 23-2. Copie d'une image de canevas dans une image

```
<!DOCTYPE html>
<html>
  <head>
    <title>Copie d'un canevas</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <canvas id='noncanvas' width='320' height='240'>
      Ceci est un élément canvas d'identifiant <i>noncanvas</i>.
      Ce texte n'est visible que dans les navigateurs non HTML5.
    </canvas>
```

```
    <img id='monimage'>

    <script>
      canvas           = O('noncanvas')
      context          = canvas.getContext('2d')
      context.fillStyle = 'red'
      S(canvas).border = '1px solid black'

      context.beginPath()
      context.moveTo(160, 120)
      context.arc(160, 120, 70, 0, Math.PI * 2, false)
      context.closePath()
      context.fill()

      S('monimage').border = '1px solid black'
      O('monimage').src    = canvas.toDataURL()
    </script>
  </body>
</html>
```

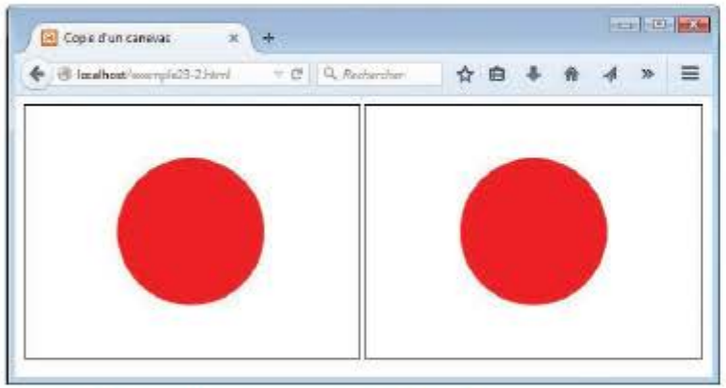


Figure 23-1. L'image de droite est une copie du canevas de gauche

Lorsque vous chargez ce code dans le navigateur, vous constatez que vous ne pouvez glisser et déposer l'image du canevas de gauche, mais que vous pouvez le faire pour l'image de droite. Ceci vous permet aussi d'enregistrer l'image dans un fichier localement ou de la déposer sur un serveur web, à l'aide du code JavaScript (et PHP sur le serveur) adéquat.

Préciser un type d'image

Lors de la création d'une image à partir d'un canevas, vous avez la possibilité de choisir le type d'image à produire, soit *.jpg*, soit *.png*. Le type par défaut est *.png* (`'image/png'`) mais si vous préférez du .jpg, modifiez légèrement l'appel à toDataURL. En même temps, vous pouvez aussi préciser le taux de compression à utiliser, compris entre `0` (la plus faible qualité) et `1` (la meilleure qualité). La ligne suivante impose une valeur de compression de `0.4`, ce qui doit produire une image d'une qualité raisonnable et d'une taille de fichier assez faible :

```
O('monimage').src = canvas.toDataURL('image/jpeg', 0.4)
```

Retenez bien que la méthode `toDataURL` s'applique à un objet `canvas` et en aucun cas à un contexte créé à partir de cet objet.

Maintenant que vous savez comment créer des images de canevas et les copier ou les exploiter, voyons les commandes de dessin disponibles, à commencer par les rectangles.

Méthode fillRect

Vous disposez de trois méthodes différentes pour tracer des rectangles, dont la première, `fillRect`. À l'appel, fournissez-lui les coordonnées du coin supérieur gauche du rectangle, suivies de la largeur et de la hauteur en pixels, comme suit :

```
context.fillRect(20, 20, 600, 200)
```

Par défaut, le rectangle se remplit de noir. Pour utiliser une autre couleur, envoyez d'abord une commande telle que la suivante, dont l'argument peut être n'importe quel nom ou valeur de couleur valide en CSS :

```
context.fillStyle = 'blue'
```

Méthode clearRect

Vous pouvez aussi tracer un rectangle dont toutes les valeurs de couleurs (rouge, vert, bleu et transparence alpha) sont réglées à `0`, comme dans la commande suivante, qui emploie le même ordre d'argument de coordonnées, de largeur et de hauteur :

```
context.clearRect(40, 40, 560, 160)
```

Lors de l'exécution d'une instruction de ce genre, le nouveau rectangle transparent récupère toutes les couleurs de la zone qu'il couvre, ne laissant que les couleurs CSS déjà appliquées à l'élément `canvas`.

Méthode strokeRect

Pour dessiner un rectangle vide, c'est-à-dire seulement délimité par son trait de bordure, utilisez une commande telle que la suivante, qui emprunte le noir par défaut ou la couleur présélectionnée pour le tracé de ses traits (*stroke*) :

```
context.strokeRect(60, 60, 520, 120)
```

Pour changer la couleur du trait, envoyez d'abord une commande telle que la suivante, avec en argument n'importe quelle couleur valide en CSS :

```
context.strokeStyle = 'green'
```

Combiner ces commandes

L'exemple 23-3 regroupe les commandes précédentes de dessin de rectangles pour obtenir l'image illustrée par la figure 23-2.

Exemple 23-3. Dessin de quelques rectangles

```
<!DOCTYPE html>
<html>
  <head>
    <title>Dessin de rectangles</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <canvas id='moncanvas' width='640' height='240'></canvas>

    <script>
      canvas              = O('moncanvas')
      context             = canvas.getContext('2d')
      S(canvas).background = 'lightblue'
      context.fillStyle   = 'blue'
      context.strokeStyle = 'green'

      context.fillRect(  20, 20, 600, 200)
      context.clearRect( 40, 40, 560, 160)
      context.strokeRect(60, 60, 520, 120)
    </script>
  </body>
</html>
```


Figure 23-2. Tracé de rectangles concentriques

Ce chapitre vous montre plus loin comment modifier les résultats produits par le changement du type de trait et sa largeur, mais poursuivons d'abord avec la modification des remplissages et l'application de dégradés.

Méthode createLinearGradient

Deux possibilités existent pour appliquer un dégradé (*gradient*) à un remplissage et la plus simple est celle de la méthode `createLinearGradient`. Vous devez préciser les coordonnées en *x* et *y* de départ et de fin du dégradé par rapport au canevas, et non par rapport à l'objet à remplir. Ceci autorise une grande subtilité. Par exemple, vous pouvez indiquer qu'un dégradé commence complètement à gauche et se termine complètement à droite du canevas, mais ne l'appliquer qu'à la zone définie par une commande de remplissage (*fill*), comme illustré dans l'exemple 23-4.

Exemple 23-4. Application d'un remplissage en dégradé

```
gradient = context.createLinearGradient(0, 80, 640,80)
gradient.addColorStop(0, 'white')
gradient.addColorStop(1, 'black')
context.fillStyle = gradient
context.fillRect(80, 80, 480,80)
```

Pour des raisons de clarté et de concision, dans cet exemple et nombre des suivants, vous ne verrez que les lignes importantes du code. Les exemples complets, avec le HTML environnant, les réglages et toutes les sections de code, sont disponibles sur le site d'accompagnement du livre.

Cet exemple crée un objet de remplissage en dégradé nommé `gradient` à l'aide d'un appel à la méthode `createLinearGradient` de l'objet `context`. L'emplacement de départ

0,80 (pour *x,y*) est à mi-chemin du bord gauche du canevas, tandis que la fin 640,80 se situe à mi-chemin du bord droit du canevas.

Ensuite, nous fournissons deux points d'arrêt de couleur (*color stops*) pour définir la première couleur du dégradé à blanc et la couleur finale à noir. Le dégradé affiche une douce transition entre ces deux couleurs sur toute la largeur du canevas, de gauche à droite.

Le dégradé ainsi préparé, nous l'appliquons ensuite à la propriété `fillStyle` de l'objet `context`, pour que l'appel final à `fillRect` puisse en tirer parti. Dans cet appel, le remplissage ne s'applique qu'à une zone rectangulaire centrale du canevas donc, bien qu'il aille du bord gauche au bord droit du canevas, la portion visible ne débute qu'à 80 pixels de la gauche et du haut, à partir du coin supérieur gauche du canevas jusqu'à la largeur de 480 et la hauteur de 80 pixels. La figure 23-3 illustre le résultat obtenu par ces lignes (ajoutées au code de l'exemple précédent).

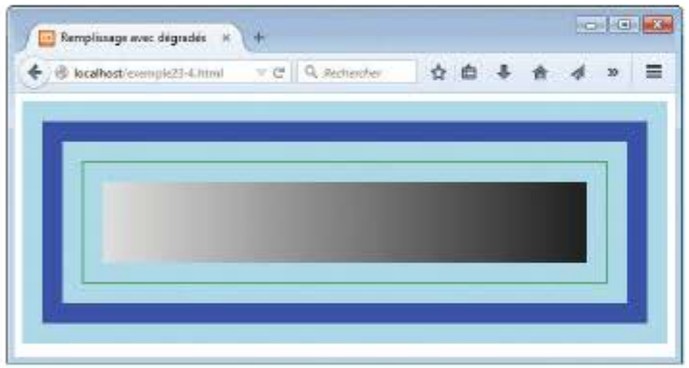


Figure 23-3. Le rectangle central possède un dégradé horizontal

Le réglage de coordonnées différentes de départ et de fin du dégradé permet de l'orienter dans n'importe quelle direction, comme l'illustrent l'exemple 23-5 et la figure 23-4.

Exemple 23-5. Différentes variantes d'angles et de couleurs de dégradés

```
gradient = context.createLinearGradient(0, 0, 160, 0)
gradient.addColorStop(0, 'white')
gradient.addColorStop(1, 'black')
context.fillStyle = gradient
context.fillRect(20, 20, 135, 200)

gradient = context.createLinearGradient(0, 0, 0, 240)
gradient.addColorStop(0, 'yellow')
gradient.addColorStop(1, 'red')
context.fillStyle = gradient
context.fillRect(175, 20, 135, 200)
```



```
gradient = context.createLinearGradient(320, 0, 480, 240)
gradient.addColorStop(0, 'green')
gradient.addColorStop(1, 'purple')
context.fillStyle = gradient
context.fillRect(330, 20, 135, 200)

gradient = context.createLinearGradient(480, 240, 640, 0)
gradient.addColorStop(0, 'orange')
gradient.addColorStop(1, 'magenta')
context.fillStyle = gradient
context.fillRect(485, 20, 135, 200)
```

Figure 23-4. Quelques exemples de dégradés linéaires différents

Cet exemple place les dégradés directement au-dessus des zones à remplir pour montrer plus clairement les variations maximales de couleurs du début à la fin.

Pour créer votre dégradé, déterminez la direction dans laquelle vous souhaitez qu'il s'effectue, puis repérez les deux points qui représentent le début et la fin. Les valeurs que vous affectez à ces points importent peu, le dégradé affichera une douce transition dans la direction donnée, même si les points se situent en dehors de la zone de remplissage.

Méthode addColorStop en détail

Vous avez le choix du nombre de points d'arrêt de couleur, bien au-delà des deux couleurs de départ et de fin utilisées dans les exemples précédents. Cette possibilité laisse place à presque tous les effets de dégradés que vous pouvez imaginer. Pour les préciser, indiquez le pourcentage du dégradé auquel se place chaque couleur à l'aide d'une valeur en virgule flottante d'emplacement le long du dégradé, comprise entre 0 et 1. Il n'est pas nécessaire de préciser l'emplacement de fin d'une couleur car celui-ci est déduit à partir de l'emplacement de début, jusqu'au point d'arrêt suivant, ou à la couleur de fin s'il n'y a pas d'autre couleur intermédiaire.

Les exemples précédents n'utilisaient que deux valeurs de départ et de fin, mais pour créer un effet d'arc-en-ciel, vous pourriez définir vos points d'arrêt de couleur comme dans l'exemple 23-6, dont la figure 23-5 illustre les résultats.

Exemple 23-6. Ajout de plusieurs points d'arrêt de couleur

```
gradient = context.createLinearGradient(0, 0, 640, 0)
gradient.addColorStop(0.00, 'red')
gradient.addColorStop(0.14, 'orange')
gradient.addColorStop(0.28, 'yellow')
gradient.addColorStop(0.42, 'green')
gradient.addColorStop(0.56, 'blue')
gradient.addColorStop(0.70, 'indigo')
gradient.addColorStop(0.84, 'violet')
```

Figure 23-5. Quelques exemples de dégradés linéaires différents

L'exemple 23-6 espace toutes les couleurs d'arrêt à environ la même distance (chaque couleur se place à des écarts de 14 % du dégradé et la dernière à 16 %), mais vous n'êtes pas obligé de vous référer absolument à ce principe : vous pouvez rassembler un peu plus certaines couleurs pour en espacer d'autres. Cela demeure à votre discrétion, ainsi que le nombre de couleurs que vous souhaitez et les emplacements de début et de fin des couleurs dans le dégradé.

Méthode createRadialGradient

HTML ne restreint pas les dégradés aux linéaires, car vous pouvez aussi créer des dégradés radiaux dans un canevas. Ils sont un peu plus complexes à mettre en place que les premiers mais à peine plus.

Le dégradé radial nécessite que vous précisiez l'emplacement central du dégradé sous la forme d'une paire de coordonnées *x* et *y*, ainsi qu'un rayon en pixels. Ils servent respectivement à définir le début du dégradé et la circonférence extérieure. Ensuite, fournissez aussi un autre ensemble de coordonnées et un rayon pour préciser la fin du dégradé.

Par exemple, pour créer un dégradé qui débute au centre d'un cercle et s'étend vers l'extérieur, écrivez une commande du genre de celle de l'exemple 23-7, dont la figure 23-6 montre le résultat.

Exemple 23-7. Création d'un dégradé radial

```
gradient = context.createRadialGradient (320, 120, 0, 320, 120, 320)
```

Figure 23-6. Un dégradé radial centré

Mais vous pouvez aussi faire preuve de fantaisie et déplacer l'emplacement du début et de la fin du dégradé radial, comme dans l'exemple 23-8 (figure 23-7), dont le centre débute à 0,120 avec un rayon de 0 pixel, pour reporter son centre de fin en 480,120, avec un rayon de 480 pixels.

Exemple 23-8. Étirement d'un dégradé radial

```
gradient = context.createRadialGradient(0, 120, 0, 480, 120, 480)
```

Figure 23-7. Un dégradé radial étiré

La manipulation des chiffres fournis à cette méthode permet de produire des effets étranges et merveilleux. N'hésitez pas à vous y essayer au départ des exemples proposés.

Utiliser des motifs de remplissage

Tout comme pour les remplissages par dégradé, vous pouvez aussi appliquer une image en guise de motif (*pattern*) de remplissage. Le motif peut provenir d'une image présente dans le document courant ou d'une image créée dans un canevas et copiée à l'aide de la méthode `toDataURL` (voir plus haut dans ce chapitre).

L'exemple 23-9 charge une image de 100×100 pixels (le symbole du yin et du yang) dans un nouvel objet image. À l'évènement `onload` de l'objet est associée une fonction qui crée un motif répétitif pour la propriété `fillStyle` du contexte. Celle-ci sert ensuite à remplir une zone de 600×200 pixels du canevas, comme illustré à la figure 23-8.

Exemple 23-9. Utilisation d'une image comme motif de remplissage

```
image     = new Image()
image.src = 'image.png'

image.onload = function()
{
  pattern          = context.createPattern(image, 'repeat')
  context.fillStyle = pattern
  context.fillRect(20, 20, 600, 200)
}
```


Figure 23-8. Tapissage avec une image comme motif de remplissage

Nous utilisons la méthode `createPattern` pour créer le motif, qui prend aussi en charge des motifs non répétitifs, ou simplement une répétition selon l'axe des *x* ou *y*. Ce réglage se situe dans le passage de l'une des valeurs suivantes en deuxième argument, après l'objet image à utiliser :

`repeat`
: Répète l'image horizontalement et verticalement.

`repeat-x`
: Répète l'image horizontalement.

`repeat-y`
: Répète l'image verticalement.

`no-repeat`
: Ne répète pas l'image.

Le motif de remplissage s'inscrit dans la totalité de la zone du canevas. Par conséquent, la commande de remplissage n'intéresse qu'une portion plus réduite dans le canevas, ce qui explique que les images apparaissent rognées sur la gauche et le haut du canevas.

Si nous n'avions pas utilisé l'évènement `onload` dans cet exemple et si le code avait été exécuté aussitôt rencontré, l'image n'aurait peut-être pas été chargée à temps et elle ne se serait pas affichée. L'association à cet évènement garantit que l'image soit disponible pour son utilisation dans le canevas, car l'évènement se déclenche dès la réussite du chargement d'une image. Par conséquent, si l'image référencée n'existe pas à l'emplacement attendu, l'évènement ne se déclenche jamais.

Écrire du texte dans le canevas

Comme on peut s'y attendre de la part d'un ensemble de fonctionnalités graphiques, le canevas prend totalement en charge l'écriture de texte, avec toutes sortes de polices, différents alignements et méthodes de remplissage. On peut toutefois se demander quel est l'intérêt d'écrire du texte dans le canevas alors que la prise en charge des polices en CSS est déjà tout à fait correcte.

Eh bien, supposons que vous souhaitez afficher un graphique ou un tableau avec des éléments graphiques. Vous aurez forcément envie d'en étiqueter des portions. De plus, les commandes disponibles permettent de produire bien plus de choses qu'une simple police en couleur. Commençons en imaginant que vous avez été chargé de créer un entête pour un site web sur la vannerie nommé OsierpédiA (en fait, il en existe déjà au moins deux mais qu'importe, poursuivons).

Pour commencer, vous sélectionnez une police et une taille adéquates, du genre de celle de l'exemple 23-10, avec un style de police gras, une taille de 140 pixels et une police Times. Vous réglez aussi la propriété `textBaseline` (ligne de base du texte) sur `top`, pour que la méthode `strokeText` puisse placer les coordonnées 0,0 de l'origine en haut à gauche du texte dans le coin supérieur gauche du canevas. La figure 23-9 montre le résultat.

Exemple 23-10. Écriture de texte dans le canevas

```
context.font         = 'bold 140px Times'
context.textBaseline = 'top'
context.strokeText('OsierpédiA', 0, 0)
```

Figure 23-9. Du texte écrit dans le canevas

Méthode strokeText

Pour écrire du texte dans le canevas, envoyez la chaine de texte et une paire de coordonnées à la méthode `strokeText`, comme suit :

```
context.strokeText('OsierpédiA', 0, 0)
```

Les coordonnées *x* et *y* servent de référence relative aux propriétés `textBaseline` et `textAlign`.

Cette méthode, qui utilise un tracé de traits, constitue le seul moyen de dessiner du texte dans le canevas. Donc, en plus de toutes les propriétés suivantes qui affectent le texte, les propriétés de tracé de traits telles que `lineWidth` (détaillée plus loin dans ce chapitre) influencent aussi l'affichage du texte.

Propriété textBaseline

La propriété `textBaseline` accepte les principales valeurs suivantes :

`top`
: S'aligne sur le sommet du texte.

`middle`
: S'aligne sur le milieu du texte.

`alphabetic`
: S'aligne sur la ligne de base alphabétique (en bas de la lettre *b* par exemple) du texte.

`bottom`
: S'aligne sur le bas de la police (en bas de la lettre *p* par exemple).

Propriété font

Le style de `font` (police) accepte une des valeurs `bold`, `italic` ou `normal` (par défaut), ou la combinaison `italic bold`, et l'expression des valeurs des tailles emprunte les mesures en `em`, `ex`, `px`, `%`, `in`, `cm`, `mm`, `pt`, ou `pc`, exactement comme en CSS. La police doit être accessible au navigateur, donc c'est généralement une des suivantes : `Helvetica`, `Impact`, `Courier`, `Times` ou `Arial`. Mais vous pouvez aussi choisir la police `Serif` ou `Sans-serif` par défaut du système de l'utilisateur. Généralement, si vous êtes certain que le navigateur a accès à une police, vous pouvez l'utiliser.

Si vous voulez utiliser une police telle que le `Times New Roman` qui comporte des espaces dans son nom, vous devez modifier la ligne correspondante comme suit, avec des apostrophes pour entourer le contenu de la propriété, différentes des guillemets qui entourent le nom de police :

```
context.font = 'bold 140px «Times New Roman»'
```

Propriété textAlign

Outre la sélection de l'alignement vertical du texte, vous pouvez préciser l'alignement horizontal à l'aide d'une des valeurs suivantes affectées à la propriété `textAlign` :

`start`
: Aligne le texte à gauche si l'écriture du document est de gauche à droite, ou à droite dans le cas contraire. Il s'agit du réglage par défaut.

`end`
: Aligne le texte à droite si l'écriture du document est de gauche à droite, ou à gauche dans le cas contraire.

`left`
: Aligne le texte à gauche.

`right`
: Aligne le texte à droite.

`center`
: Centre le texte.

La propriété s'utilise comme suit :

```
context.textAlign = 'center'
```

Dans le cas de l'exemple courant, il est nécessaire d'aligner le texte à gauche pour qu'il vienne buter près du bord du canevas, donc la propriété `textAlign` n'est pas utilisée et c'est sa valeur par défaut qui s'applique, l'alignement à gauche.

Méthode fillText

Vous disposez de toutes sortes de possibilités avec la méthode `fill` pour remplir le texte du canevas, que ce soit une couleur pleine, un dégradé linéaire ou radial, ou encore un motif de remplissage. Voyons ce que donne le remplissage du texte d'entête par une texture de panier en osier, comme dans l'exemple 23-11, dont la figure 23-10 illustre le résultat.

Exemple 23-11. Remplissage du texte avec un motif

```
image     = new Image()
image.src = 'osier.jpg'

image.onload = function()
{
  pattern         = context.createPattern(image, 'repeat')
  context.fillStyle = pattern
  context.fillText(  'OsierpédiA', 0, 0)
  context.strokeText('OsierpédiA', 0, 0)
}
```

Figure 23-10. Le texte écrit cette fois avec un motif de remplissage

Par mesure de précaution, nous avons conservé l'appel à `strokeText` dans cet exemple pour tracer les contours des lettres en noir. Sans cette précaution, la définition des contours s'avère insuffisante.

Une grande variété de types et de motifs de remplissage peut également servir et la simplicité du canevas permet de les expérimenter facilement. Qui plus est, si vous le souhaitez, dès que l'entête vous satisfait, il vous suffit d'en enregistrer une copie à l'aide d'un appel à `toDataURL`, comme décrit plus haut dans ce chapitre. Vous pourrez ensuite exploiter cette image en guise de logo et la placer sur d'autres pages et d'autres sites, par exemple.

Méthode measureText

Lorsque vous travaillez sur du texte dans un canevas, il est parfois nécessaire de connaitre la quantité d'espace qu'il occupe pour peaufiner sa disposition. Pour connaitre cette information et d'autres du même genre, utilisez la méthode `measureText`, comme suit (en supposant que les propriétés du texte sont déjà définies à ce stade) :

```
metrics = context.measureText('OsierpédiA')
width   = metrics.width
```

Remarquez toutefois que, comme la hauteur en pixels de la police du texte est égale à la taille de police en points au moment de la définition de la police, l'objet `metrics` ne fournit aucune mesure de hauteur.

Tracer des traits

Le canevas fournit une pléthore de fonctions de tracé de traits pour satisfaire à peu près tous les besoins, y compris des types de lignes, d'extrémités et de jointures de traits, ainsi que des chemins et des courbes de toutes sortes. Commençons par examiner la propriété évoquée dans la section précédente sur l'écriture de texte dans le canevas.

Propriété lineWidth

Toutes les méthodes de canevas qui dessinent à l'aide de traits font usage de `lineWidth` et d'un certain nombre d'autres propriétés des traits. Son utilisation est très simple, puisqu'il suffit de préciser la largeur du trait en pixels, comme suit, où cette largeur est réglée à 3 pixels :

```
context.lineWidth = 3
```

Propriétés lineCap et lineJoin

Lorsqu'une ligne s'interrompt et qu'elle a une largeur supérieure à un pixel, vous pouvez choisir la manière dont s'affiche l'extrémité (*line cap*, littéralement « capuchon de ligne ») à l'aide de la propriété `linecap`, qui accepte les valeurs `butt` (terminaison plate à la fin exacte du trait), `round` (ajout d'un demi-cercle de terminaison) ou `square` (ajout d'un carré de terminaison). Par exemple :

```
context.lineCap = 'round'
```

En outre, lorsque vous joignez deux lignes de largeur supérieure à un pixel, il importe de préciser exactement comment elles doivent se rencontrer. La propriété `lineJoin` permet de choisir le mode d'affichage de cette jonction avec une des valeurs `round` (pointe arrondie), `bevel` (pointe tronquée) ou `miter` (pointe prolongée, littéralement « assemblage à onglet »), comme suit :

```
context.lineJoin = 'bevel'
```

L'exemple 23-12 (exposé au complet, cette fois, du fait de sa complexité relative) applique les trois valeurs à chacune de ces deux propriétés utilisées en conjonction pour créer le résultat récapitulatif de la figure 23-11. Les explications relatives aux méthodes `beginPath`, `closePath`, `moveTo` et `lineTo` apparaissent dans les sections suivantes.

Exemple 23-12. Affichage de combinaisons des terminaisons et jonctions de lignes

```
<!DOCTYPE html>
<html>
  <head>
    <title>Tracé de lignes</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <canvas id='moncanvas' width='535' height='360'></canvas>

    <script>
      canvas              = O('moncanvas')
      context             = canvas.getContext('2d')
      S(canvas).background = 'lightblue'
      context.fillStyle   = 'red'
      context.font        = 'bold 12pt Courier'
      context.strokeStyle = 'blue'
      context.textBaseline = 'top'
      context.lineWidth   = 20
      caps                = ['butt',  'round', 'square']  // extrémités
      joins               = ['round', 'bevel', 'miter']   // jonctions

      for (j = 0 ; j < 3 ; ++j)
      {
        for (k = 0 ; k < 3 ; ++k)
        {
          context.lineCap  = caps[j]
          context.lineJoin = joins[k]

          context.fillText(' cap:' + caps[j],  33 + j * 180, 45 + k * 120)
          context.fillText('join:' + joins[k], 33 + j * 180, 65 + k * 120)

          context.beginPath()
          context.moveTo( 20 + j * 180, 100 + k * 120)
          context.lineTo( 20 + j * 180,  20 + k * 120)
          context.lineTo(155 + j * 180,  20 + k * 120)
          context.lineTo(155 + j * 180, 100 + k * 120)
          context.stroke()
          context.closePath()
        }
      }
    </script>
  </body>
</html>
```

Le code règle quelques propriétés puis imbrique deux boucles, l'une pour les extrémités de traits (`caps`), l'autre pour les jonctions (`join`). La boucle centrale règle d'abord les valeurs de `lineCap` et `lineJoin`, puis les affiche dans le canevas à l'aide de la méthode `fillText`.

À l'aide de ces réglages, le code trace neuf formes avec un trait de 20 pixels de large, chacune avec une combinaison différente de terminaison et de jonction de traits, comme l'illustre la figure 23-11.

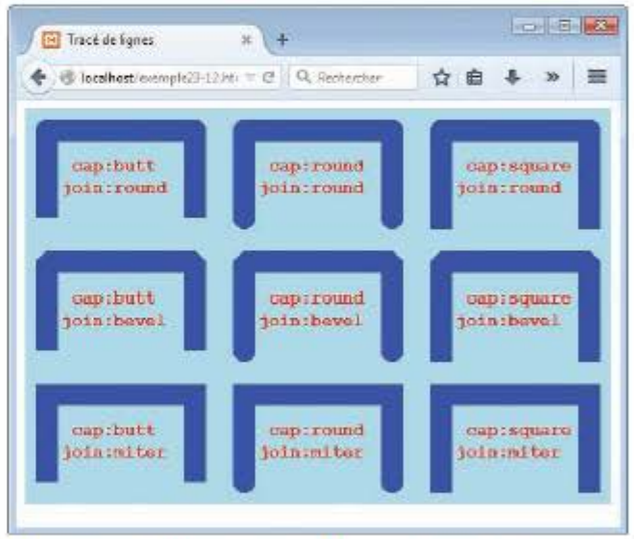


Figure 23-11. Toutes les combinaisons possibles d'extrémités et de jonctions de traits

Si vous êtes attentif, les extrémités `butt` sont courtes, les `squares` sont plus longues et les `round` sont entre les deux. De plus, les extrémités des jonctions `round` sont incurvées, les `bevel` ont le coin tronqué et les `miter` ont des coins pointus. Les jonctions de lignes s'appliquent aussi aux jointures d'angles différents de 90 degrés.

Propriété miterLimit

Lorsque les jonctions paraissent trop courtes pour afficher une belle finition, vous pouvez les prolonger à l'aide de la propriété `mitterLimit`, comme suit :

```
context.miterLimit = 15
```

Comme la valeur prédéfinie est de `10`, vous pouvez aussi réduire la limite de jonction. Si `miterLimit` est définie à une valeur insuffisante pour une jonction, les jointures trop découpées subissent une troncature (comme pour `lineJoin = bevel`). Donc si vous éprouvez des soucis avec vos pointes de jonction, augmentez simplement la valeur de `miterLimit` pour que les choses rentrent dans l'ordre.

Tracer des lignes brisées (paths)

L'exemple précédent tire parti de deux méthodes pour définir des chemins pour les méthodes de tracé de lignes. La méthode `beginPath` définit le début du chemin et `endPath` en scelle la fin. Au sein de chaque chemin, vous utilisez différentes méthodes pour déplacer l'emplacement du dessin et créer des lignes, des courbes et d'autres formes. Reprenons l'examen de la section correspondante de l'exemple 23-12, simplifiée cette fois pour n'afficher qu'une seule copie du motif :

```
context.beginPath()
context.moveTo(20, 100)
context.lineTo(20, 20)
context.lineTo(155, 20)
context.lineTo(155,100)
context.stroke()
context.closePath()
```

Cet extrait de code initialise un chemin de tracé à la première ligne, déplace la position du tracé à l'emplacement défini à 20 pixels à droite et 100 pixels du haut, à partir du coin supérieur gauche du canevas, à l'aide de la méthode `moveTo`.

Suivent trois appels à `lineTo` qui dessinent réellement des traits, le premier vers le haut, à l'emplacement 20,20, le deuxième vers la droite, à l'emplacement 155,20, puis vers le bas, à 155,100. Dès que ce chemin est ainsi défini, l'appel à la méthode `stroke` impose réellement le tracé, puis la méthode `endPath` clôture le chemin car il n'est plus nécessaire.

Clôturez toujours vos chemins dès que vous en avez fini avec eux ; sinon, vous risquez d'être confronté à des résultats inattendus lors de l'utilisation de plusieurs chemins de tracé.

Méthodes moveTo et LineTo

Les méthodes `moveTo` et `lineTo` prennent toutes deux des coordonnées en x et en y en guise d'arguments. La différence entre ces méthodes se situe dans le fait que `moveTo` relève un stylo imaginaire à l'emplacement courant et le déplace ensuite à l'endroit

précisé, tandis que `lineTo` abaisse le stylo imaginaire sur le canevas à l'emplacement courant et dessine un trait jusqu'à l'endroit indiqué. Et la ligne n'apparaît réellement qu'à l'appel de la méthode `stroke`, pas avant. Autrement dit, `lineTo` crée un trait potentiel, qui peut tout aussi bien constituer une ligne de contour d'une zone de remplissage, par exemple.

Méthode stroke

La méthode `stroke` endosse la tâche de dessiner réellement toutes les lignes créées jusqu'au moment de son appel dans un chemin du canevas. Lorsqu'appelée au sein d'un chemin non clos, elle a pour effet de dessiner immédiatement tout, jusqu'à l'emplacement le plus récent du stylo imaginaire.

En revanche, si vous clôturez le chemin, puis appelez `stroke`, celle-ci relie aussi le chemin depuis l'emplacement courant jusqu'à l'emplacement de début du chemin, ce qui, dans notre exemple, crée une forme rectangulaire. Or, ce n'était pas l'effet voulu car nous voulions voir les extrémités et les jonctions des traits.

Comme nous le verrons un peu plus loin, cet effet de fermeture du chemin pour obtenir une figure fermée devient indispensable avant d'appliquer à la forme un remplissage avec une méthode `fill`, sinon le remplissage déborde des limites du chemin pour envahir le canevas.

Méthode rect

S'il avait été nécessaire de dessiner des rectangles au lieu des figures à trois côtés précédentes, sans fermer le chemin tout de suite, un dernier appel à `lineTo` aurait pu assurer la jonction avec le début du chemin, comme suit (avec le dernier `lineTo` en gras) :

```
context.beginPath()
context.moveTo(20, 100)
context.lineTo(20, 20)
context.lineTo(155, 20)
context.lineTo(155, 100)
context.lineTo(20, 100)
context.closePath()
```

Mais il existe un moyen plus simple pour dessiner de tels rectangles détourés, à l'aide de la méthode `rect`, comme suit :

```
rect(20, 20, 155, 100)
```

En un seul appel, cette méthode prend deux paires de coordonnées x et y et trace un rectangle avec son coin supérieur gauche situé à 20,20 et son coin inférieur droit à 155,100.

Remplir des zones

Les chemins autorisent la création de zones fermées complexes, qu'il est ensuite possible de remplir à l'aide de couleurs pleines, de dégradés ou d'un motif. L'exemple 23-13 emprunte un peu de trigonométrie élémentaire pour créer une forme en étoile complexe. Nous ne détaillons pas les mathématiques nécessaires pour aboutir à ce résultat car elles ne sont pas essentielles à l'exemple, mais si vous souhaitez jouer avec le code, essayez de modifier les valeurs affectées aux variables `pointes`, `echelle1` et `echelle2` pour voir les effets obtenus.

Ce qui doit attirer surtout votre attention, ce sont les instructions en gras. La première débute un chemin ; deux appels en boucle à `lineTo` définissent la forme ; le chemin est clôturé ; puis la méthode `stroke` dessine le contour de l'étoile en orange et la méthode `fill` la remplit de jaune (figure 23-12).

Exemple 23-13. Remplissage d'un chemin complexe

```
<!DOCTYPE html>
<html>
  <head>
    <title>Remplissage d'un chemin</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <canvas id='moncanvas' width='320' height='320'></canvas>

    <script>
      canvas              = O('moncanvas')
      context             = canvas.getContext('2d')
      S(canvas).background = 'lightblue'
      context.strokeStyle = 'orange'
      context.fillStyle   = 'yellow'

      orig     = 160
      pointes  = 21
      dist     = Math.PI / pointes * 2
      echelle1 = 150
      echelle2 = 80

      context.beginPath()

      for (j = 0 ; j < pointes ; ++j)
      {
        x = Math.sin(j * dist)
        y = Math.cos(j * dist)
        context.lineTo(orig + x * echelle1, orig + y * echelle1)
        context.lineTo(orig + x * echelle2, orig + y * echelle2)
      }

      context.closePath()
      context.stroke()
      context.fill()
```



```
    </script>
  </body>
</html>
```

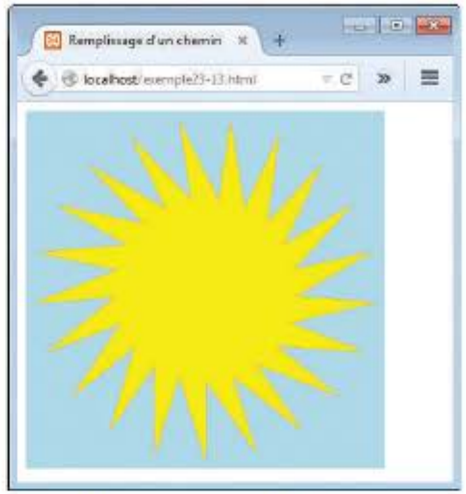


Figure 23-12. Tracé et remplissage d'un chemin complexe

Les chemins permettent de créer des objets d'une grande complexité à l'aide de formules, de boucles (comme dans cet exemple) ou de longues successions de `moveTo`, de `lineTo` ou autres.

Méthode clip

Lors de la construction d'un chemin, il est parfois nécessaire d'ignorer des sections du canevas, par exemple pour dessiner « derrière » un autre objet et n'afficher que les portions visibles. La méthode `clip` (rogner) permet d'obtenir cet effet et crée une frontière en dehors de laquelle les méthodes `stroke`, `fill` et autres sont ignorées et n'ont aucun effet.

Pour illustrer cette technique, l'exemple 23-14 crée l'illusion d'un store à lames devant une fenêtre, par le déplacement du pointeur du stylo imaginaire au bord gauche du canevas, puis le tracé d'un `lineTo` jusqu'au bord droit, un tracé vers de bas de 30 pixels, puis un autre de retour au bord gauche et ainsi de suite. Ceci crée un serpentin qui délimite des barres horizontales successives tous les 30 pixels, comme à la figure 23-13.

Exemple 23-14. Création d'une zone de clip

```
context.beginPath()

for (j = 0 ; j < 10 ; ++j)
{
  context.moveTo(20,  j * 48)
  context.lineTo(620, j * 48)
  context.lineTo(620, j * 48 + 30)
  context.lineTo(20,  j * 48 + 30)
}

context.stroke()
context.closePath()
```

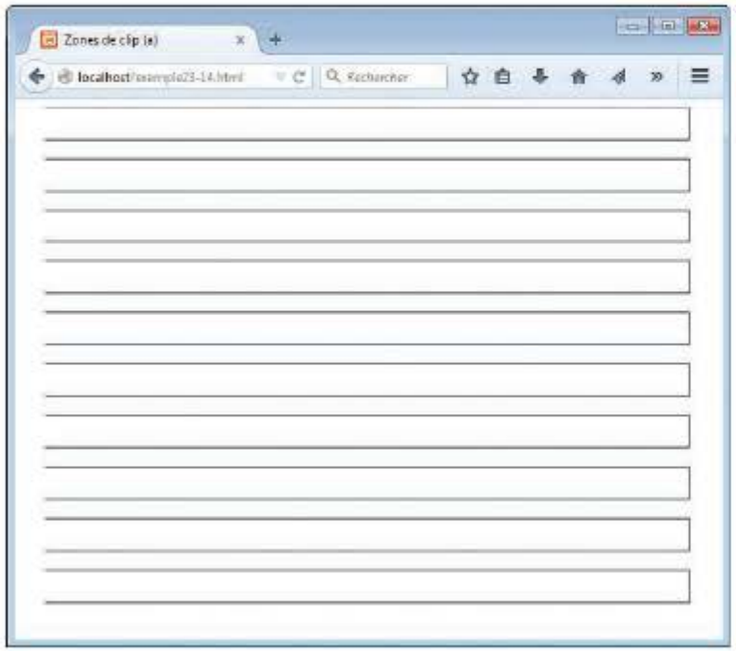


Figure 23-13. Un chemin de barres horizontales

Pour convertir cet exemple en une zone de rognage du canevas, remplacez simplement l'appel à `stroke` (en gras) par un autre à la méthode `clip`, comme suit :

```
context.clip()
```


À partir de là, les contours des barres ne sont plus visibles mais la zone de rognage est en place. Pour le démontrer, l'exemple 23-15 applique cette substitution de méthodes puis ajoute à l'exemple précédent le dessin d'une image simple au canevas, avec de l'herbe verte sous un ciel bleu et un soleil éclatant. Les modifications à partir de l'exemple 23-12 sont en gras et le résultat est celui de la figure 23-14.

Exemple 23-15. Dessins entre les limites de la zone de rognage

```
context.fillStyle    = 'white'
context.strokeRect(20, 20, 600, 440) // Bordure noire
context.fillRect(  20, 20, 600, 440)  // Fond blanc

context.beginPath()

for (j = 0 ; j < 10 ; ++j)
{
  context.moveTo(20,  j * 48)
  context.lineTo(620, j * 48)
  context.lineTo(620, j * 48 + 30)
  context.lineTo(20,  j * 48 + 30)
}

context.clip()                      // Rognage
context.closePath()

context.fillStyle    = 'blue'       // Ciel bleu
context.fillRect(20, 20,  600, 320)
context.fillStyle    = 'green'      // Herbe verte
context.fillRect(20, 320, 600, 140)
context.strokeStyle = 'orange'
context.fillStyle    = 'yellow'

orig     = 170
pointes  = 21
dist     = Math.PI / pointes * 2
echelle1 = 130
echelle2 = 80

context.beginPath()

for (j = 0 ; j < pointes ; ++j)
{
  x = Math.sin(j * dist)
  y = Math.cos(j * dist)
  context.lineTo(orig + x * echelle1, orig + y * echelle1)
  context.lineTo(orig + x * echelle2, orig + y * echelle2)
}

context.closePath()
context.stroke()                    // Soleil au contour orange
context.fill()                      // et rempli de jaune
```

Figure 23-14. Le dessin apparait seulement dans la zone autorisée par le rognage

Bon, il est évident que nous ne remporterons aucun concours d'art avec ceci mais, au moins, vous constatez la puissance du rognage lorsqu'il est exploité avec efficacité.

Méthode isPointInPath

Il est parfois nécessaire de déterminer si un point précis s'inscrit à l'intérieur d'un chemin que vous avez construit. C'est le genre de fonction exploitable uniquement si vous êtes très adroit en JavaScript et que vous rédigez un programme plutôt complexe. Elle intervient plus généralement dans une instruction conditionnelle `if`, comme suit :

```
if (context.isPointInPath(23, 87))
{
  // Alors faire quelque chose ici
}
```

Si l'emplacement précisé se situe sur un des points du chemin, la méthode renvoie la valeur `true` et le corps de l'instruction `if` s'exécute. Sinon, elle renvoie `false` et le corps du `if` est éludé.

La méthode `isPointInPath` trouve une utilisation idéale dans la création de jeux basés sur le canevas, où il faut déterminer, par exemple, si un missile touche une cible, une balle frappe un mur ou une raquette, ou toute autre condition d'atteinte de limite comparable.

Dessiner des courbes

Outre les chemins linéaires, vous pouvez créer une infinité de variétés de chemins courbés à l'aide d'un choix de méthodes différentes, allant des simples arcs et cercles aux courbes quadratiques et de Bézier.

En pratique, il n'est pas nécessaire d'utiliser les chemins pour créer nombre de lignes, de rectangles et de courbes, car vous pouvez les dessiner directement par un appel à leurs méthodes. Cependant, comme l'utilisation des chemins procure un contrôle plus précis, j'ai tendance à toujours dessiner sur le canevas parmi des chemins définis, comme dans les exemples qui suivent.

Méthode arc

La méthode `arc` exige avant tout les coordonnées en *x* et *y* du centre de l'arc et un rayon en pixels. En plus de ces valeurs, vous devez préciser les décalages d'angles en radians de début et de fin, et, facultativement, un sens de tracé, comme suit :

```
context.arc(55, 85, 45, 0, Math.PI / 2, false)
```

Le sens par défaut est celui des aiguilles d'une montre et correspond à la valeur `false`, que vous pouvez omettre. La valeur `true` désigne le sens contraire des aiguilles d'une montre.

L'exemple 23-16 crée trois ensembles de quatre arcs, dont les deux premiers sont dessinés dans le sens des aiguilles d'une montre et les deux derniers dans le sens contraire. Dans le premier ensemble de quatre arcs, le chemin est clôturé avant l'appel de `stroke`, ce qui provoque une jonction entre les points de départ et de fin. Dans les deux autres ensembles, les arcs sont dessinés avant la fermeture du chemin, donc aucun trait ne rejoint les points de départ et d'arrivée.

Exemple 23-16. Tracé de différents arcs

```
context.strokeStyle  = 'blue'
arcs =
[
  Math.PI,
  Math.PI * 2,
  Math.PI / 2,
  Math.PI / 180 * 59
]

for (j = 0 ; j < 4 ; ++j)
{
  context.beginPath()
```

```
  context.arc(80 + j * 160, 80, 70, 0, arcs[j])
  context.closePath()
  context.stroke()
}

context.strokeStyle = 'red'

for (j = 0 ; j < 4 ; ++j)
{
  context.beginPath()
  context.arc(80 + j * 160, 240, 70, 0, arcs[j])
  context.stroke()
  context.closePath()
}

context.strokeStyle = 'green'

for (j = 0 ; j < 4 ; ++j)
{
  context.beginPath()
  context.arc(80 + j * 160, 400, 70, 0, arcs[j], true)
  context.stroke()
  context.closePath()
}
```

Pour conserver un code plus court, nous dessinons les arcs à l'aide de boucles, de sorte que la longueur de chacun est mémorisée dans le tableau `arcs`. Ces valeurs sont en radians, or, comme un radian équivaut en degrés à 180 ÷ π (où π est le rapport entre la circonférence du cercle et son diamètre, soit approximativement 3,1415927), elles sont équivalentes à :

`Math.PI`
: équivaut à 180 degrés

`Math.PI * 2`
: équivaut à 360 degrés

`Math.PI / 2`
: équivaut à 90 degrés

`Math.PI / 180 * 59`
: équivaut à 59 degrés

La figure 23-15 montre les trois rangées d'arcs et illustre tant l'utilisation de l'argument `true` dans le dernier ensemble, que l'importance de choisir soigneusement où vous placez la clôture du chemin, selon que vous souhaitez ou non un trait de fermeture de l'arc entre ses points de début et de fin.

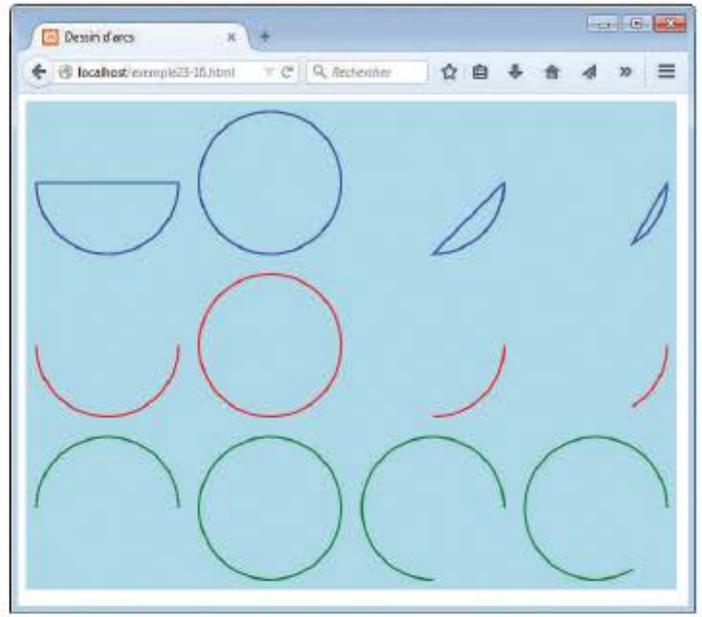


Figure 23-15. Tracé de différents types d'arcs

Si vous vous sentez plus à l'aise avec les angles en degrés au lieu des radians, vous pouvez vous créer une nouvelle fonction de bibliothèque Math, comme celle-ci :

```
Math.degresEnRadians = function(degres)
{
  return degres * Math.PI / 180
}
```

Ensuite, remplacez le code de création du tableau des arcs qui commence à la deuxième ligne du code de l'exemple 23-16, par ce qui suit :

```
arcs =
[
  Math.degresEnRadians(180),
  Math.degresEnRadians(360),
  Math.degresEnRadians(90),
  Math.degresEnRadians(59)
]
```

Méthode arcTo

Au lieu de créer un arc isolé, il est possible de tracer un arc depuis l'emplacement courant du chemin vers un autre, comme avec l'appel suivant à `arcTo`, qui nécessite deux paires de coordonnées *x* et *y*, ainsi qu'un rayon d'arc :

```
context.arcTo(100, 100, 200, 200, 100)
```

Les emplacements passés à la méthode représentent les points des tangentes imaginaires qui touchent la circonférence de l'arc à ses points de début et de fin.

Pour illustrer le fonctionnement, l'exemple 23-17 dessine huit arcs différents avec des rayons compris entre 0 et 280 pixels. Chaque itération de la boucle crée un nouveau chemin au point de départ 20,20. Puis, elle trace un arc à l'aide de tangentes imaginaires de cet emplacement jusqu'au point 240,240, puis de là, vers l'emplacement 460,20. Dans ce contexte, cela définit deux tangentes imaginaires qui se croisent à 90 degrés en 240,240, pour créer un grand V.

Exemple 23-17. Tracé de huit arcs de rayons différents

```
for (j = 0 ; j <= 280 ; j += 40)
{
  context.beginPath()
  context.moveTo( 20, 20)
  context.arcTo(240, 240, 460, 20, j)
  context.lineTo(460, 20)
  context.stroke()
  context.closePath()
}
```

En réalité, la méthode `arcTo` s'arrête de dessiner au point où l'arc touche la deuxième tangente imaginaire. C'est pour cette raison qu'après chaque appel à `arcTo`, la méthode `lineTo` intervient pour tracer le reste de la ligne de ce point vers l'emplacement 460,20. Ensuite, l'appel à `stroke` dessine le résultat dans le canevas et `closePath` clôture le chemin.

La figure 23-16 montre que l'appel de la méthode `arcTo` avec un rayon de `0` provoque une jonction directe. Dans ce cas-ci, il s'agit d'un angle de 90 degrés, mais si les deux tangentes imaginaires sont disposées selon un angle différent, alors la jonction respecte cet angle. Ensuite, au fur et à mesure des itérations de la boucle, comme le rayon augmente, les arcs deviennent de plus en plus grands.

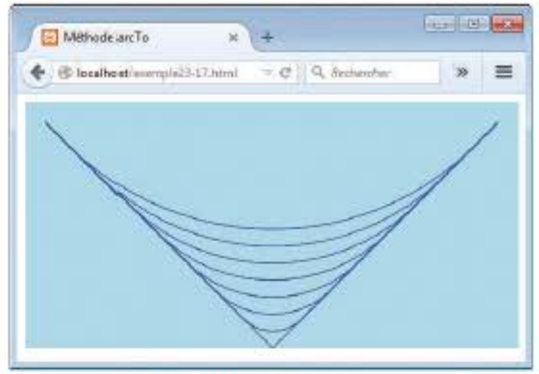


Figure 23-16. Tracé d'arcs de différents rayons

Fondamentalement, la méthode `arcTo` s'avère la plus utile lorsqu'il s'agit de relier une section de dessin à une autre à l'aide d'un arc en fonction des emplacements précédent et suivant, comme s'ils étaient tangents à l'arc à créer. Si cela semble un peu compliqué, ne vous inquiétez pas : vous maîtriserez bientôt le concept et comprendrez mieux les aspects pratiques et logiques de ce tracé d'arcs.

Méthode quadraticCurveTo

Les types d'arcs que nous venons de voir sont bien utiles mais comportent un défaut : leur rayon est figé. Et, dans certains cas complexes, ils ne suffisent plus. C'est pour cette raison que d'autres types d'arcs existent, comme ceux que permet la méthode `quadraticCurveTo`. Cette méthode permet en quelque sorte de placer un aimant (appelé point d'attraction) près, ou loin, d'une courbe pour l'attirer dans cette direction, un peu comme un objet dans l'espace est attiré dans sa course par la gravité des planètes et des étoiles qu'il côtoie. À l'inverse de la gravité, plus le point d'attraction est éloigné, plus il déforme la courbe.

L'exemple 23-18 propose six appels à cette méthode pour créer le chemin d'un nuage moelleux, rempli ensuite de blanc. La figure 23-17 illustre par des traits en pointillé les angles qui représentent les points d'attraction appliqués à chaque courbe.

Exemple 23-18. Dessin d'un nuage à l'aide de courbes quadratiques

```
context.beginPath()
context.moveTo(180, 60)
context.quadraticCurveTo(240,   0, 300,  60)
context.quadraticCurveTo(460,  30, 420, 100)
context.quadraticCurveTo(480, 210, 340, 170)
context.quadraticCurveTo(240, 240, 200, 170)
context.quadraticCurveTo(100, 200, 140, 130)
context.quadraticCurveTo( 40,  40, 180,  60)
context.fillStyle = 'white'
```

```
context.fill()
context.closePath()
```

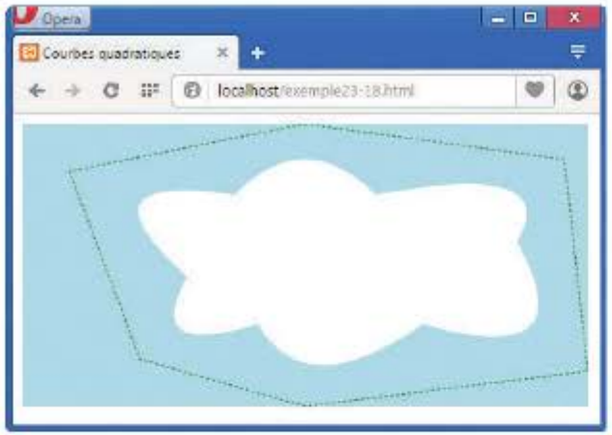


Figure 23-17. Dessin à l'aide de courbes quadratiques

Remarquez que la réalisation des traits en pointillé autour du nuage fait appel à la méthode `stroke` conjointement à la méthode `setLineDash`, qui nécessite une liste représentant les longueurs des tirets et des espaces. L'exemple choisi utilise `setLineDash([2, 3])`, mais vous pouvez créer des traits interrompus aussi complexes que vous le souhaitez, comme `setLineDash([1, 2, 1, 3, 5, 1, 2, 4])`. Toutefois, ceci n'est pas documenté dans ce livre parce que la méthode n'est prise en charge actuellement que par Internet Explorer, Chrome et Opera (figure 23-17). Espérons qu'elle sera aussi implémentée sous peu par les autres navigateurs, car elle représente une excellente amélioration pour la création de contours et de bordures dans le cadre de cartes géographiques, par exemple.

Méthode bezierCurveTo

Si les courbes quadratiques ne suffisent pas à satisfaire vos besoins, il reste alors les courbes de Bézier et leurs deux points d'attraction. Avec cette méthode, vous pouvez réaliser exactement ce qu'illustre la figure 23-18, issue de l'exemple 23-19 qui crée une courbe entre les points 240,20 et 240,220, mais avec des points d'attraction invisibles en dehors du canevas (dans ce cas-ci), aux emplacements 720,480 et -240,-240.

Exemple 23-19. Création d'une courbe de Bézier avec deux points d'attraction

```
context.beginPath()
context.moveTo(240, 20)
context.bezierCurveTo(720, 480, -240, -240, 240, 220)
```



```
context.stroke()
context.closePath()
```

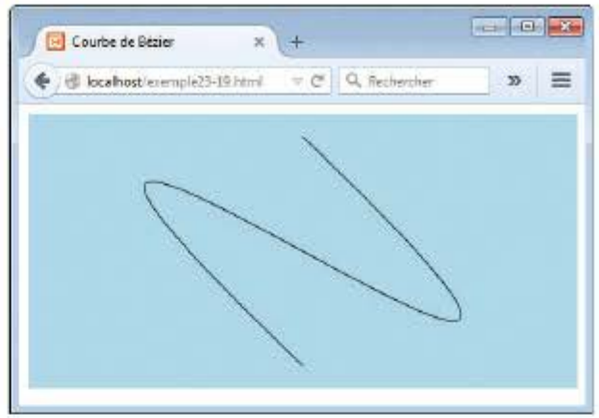


Figure 23-18. Une courbe de Bézier avec deux points d'attraction

Il n'est pas obligatoire de placer les points d'attraction de côtés opposés du canevas car vous pouvez les disposer où vous le souhaitez. S'ils sont placés l'un près de l'autre, ils exercent une attraction conjointe, à l'inverse de ce qu'illustre l'exemple précédent. L'utilisation de tous ces types d'arcs et de courbes permet de dessiner n'importe quelle courbe.

Manipuler des images

Les méthodes graphiques permettent de dessiner et d'écrire dans le canevas, mais vous avez aussi la possibilité d'y placer des images et d'en extraire. Les commandes ne se limitent pas à du copier-coller, car vous pouvez aussi étirer et déformer des images à leur lecture ou leur écriture, et vous disposez du plein contrôle sur les effets de composition et d'ombre.

Méthode drawImage

Avec la méthode `drawImage`, vous prenez un objet image chargé à partir d'un site web, téléversé sur un serveur ou même extrait d'un canevas, et vous le redessinez dans le canevas. La méthode prend en charge toutes sortes d'arguments dont nombre sont facultatifs. La forme la plus simple d'appel de `drawImage` est la suivante, qui ne prend que l'image et des coordonnées en *x* et *y* :

```
context.drawImage(monImage, 20, 20)
```

Cette commande dessine l'image contenue dans l'objet `monImage` sur le canevas par l'entremise du contexte `context`, avec son coin supérieur gauche placé à l'emplacement 20,20.

Pour garantir qu'une image a bel et bien été chargée avant de l'utiliser, une bonne pratique de programmation consiste à enfermer le code de traitement de l'image dans une fonction, déclenchée seulement à l'achèvement du chargement de l'image, comme suit :

```
monimage = new Image()
monimage.src = 'image.gif'

monimage.onload = function()
{
  context.drawImage(monimage, 20, 20)
}
```

Redimensionner une image

Pour redimensionner une image lorsqu'elle est placée dans le canevas, ajoutez deux autres arguments (en gras) qui représentent la largeur et la hauteur finale exigée, comme suit :

```
context.drawImage(monimage, 140, 20, 220, 220)
context.drawImage(monimage, 380, 20,  80, 220)
```

Cet exemple dépose deux copies de l'image : la première à l'emplacement 140,20 avec un agrandissement (si elle mesure initialement 100×100 pixels, sa taille est portée à 220×220 pixels), tandis que la seconde aboutit à l'emplacement 380,20 avec un rétrécissement horizontal et un agrandissement de hauteur, pour atteindre 80×220 pixels.

Sélectionner une zone d'une image

Vous n'êtes pas obligé d'utiliser la totalité d'une image, car vous pouvez choisir seulement une zone de celle-ci lors de l'appel à `drawImage`. Ceci s'avère pratique si vous voulez placer toutes les images que vous comptez utiliser dans un seul fichier image, puis n'extraire que les sections nécessaires. C'est d'ailleurs une astuce souvent employée par les développeurs pour accélérer le chargement des pages et réduire les requêtes au serveur.

Cette possibilité exige un peu plus de subtilité parce qu'au lieu d'ajouter des arguments à la suite des autres lors de l'appel, lorsque vous extrayez une portion d'une image, vous devez placer ces arguments en premier dans la liste.

Par exemple, pour positionner une image aux coordonnées 20,140, vous écrivez :

```
context.drawImage(monimage, 20, 140)
```

Pour lui donner une largeur et une hauteur de 100×100 pixels, vous ajoutez ces dimensions à la suite des précédentes, comme suit :

```
context.drawImage(monimage, 20, 140, 100, 100)
```

En revanche, pour extraire une section de 40×40 pixels (par exemple), dont le coin supérieur gauche se situe à 30,30 par rapport à l'image, alors vous devez appeler la méthode comme suit (avec les nouveaux arguments en gras) :


```
context.drawImage(monimage, 30, 30, 40, 40, 20, 140)
```

Pour redimensionner cette portion dans un carré de 100 pixels de côtés, vous devez écrire :

```
context.drawImage(monimage, 30, 30, 40, 40, 20, 140, 100, 100)
```

Cette façon de procéder est mélangeante et il n'existe aucune raison logique pour expliquer le fonctionnement de cette méthode de la sorte. Mais comme c'est ainsi, vous n'avez pas d'autre choix que de vous souvenir des arguments qui viennent en premier et dans quelles conditions.

L'exemple 23-20 exploite une série d'appels différents à la méthode `drawImage` pour obtenir le résultat illustré à la figure 23-19. Le code espace les arguments pour que les valeurs de chaque colonne contiennent les mêmes informations.

Exemple 23-20. Diverses façons de dessiner une image dans le canevas

```
monimage     = new Image()
monimage.src = 'image.png'

monimage.onload = function()
{
  context.drawImage(monimage,                  20,  20         )
  context.drawImage(monimage,                 140,  20, 220, 220)
  context.drawImage(monimage,                 380,  20,  80, 220)
  context.drawImage(monimage, 30, 30, 40, 40, 20, 140, 100, 100)
}
```

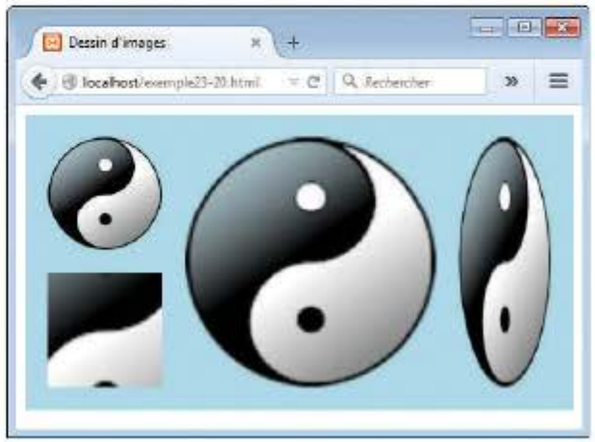


Figure 23-19. Dessin, redimensionnement et extraction d'images dans un canevas

Copier à partir d'un canevas

Un canevas peut aussi servir d'image source pour dessiner dans le même canevas (ou un autre). Fournissez le nom d'objet du canevas à la place d'un objet image et utilisez tous les arguments suivants de la même manière que pour une image.

À ce stade, j'aurais été ravi de vous montrer l'utilisation d'un élément `<video>` (voir chapitre suivant) en tant qu'image source pour dessiner dans un canevas. Malheureusement, Internet Explorer ne prend pas encore en charge cette fonctionnalité, donc il est encore trop tôt pour nous étendre sur les possibilités stupéfiantes que vous pouvez obtenir de cette manière, comme la manipulation de vidéo en direct, la colorisation, le gaufrage et bien d'autres encore. Toutefois, si cela éveille votre curiosité, vous obtiendrez plus d'informations (en anglais) sur la combinaison de vidéo avec le canevas sur *http://html5doctor.com/video-canvasmagic*.

Ajouter des ombres

Que vous dessiniez une (portion d') image ou n'importe quoi d'autre dans le canevas, vous pouvez également définir une ombre portée sous l'objet correspondant à l'aide d'une ou plusieurs propriétés suivantes :

`shadowOffsetX`
: Le décalage horizontal en pixels de l'ombre vers la droite (ou la gauche si la valeur est négative).

`shadowOffsetY`
: Le décalage vertical en pixels de l'ombre vers le bas (ou le haut si la valeur est négative).

`shadowBlur`
: Le nombre de pixels du flou (*blur*) des contours de l'ombre.

`shadowColor`
: La couleur de base de l'ombre. Si du flou est en action, cette couleur se mélange à celle du fond dans la zone de flou.

Ces propriétés s'appliquent aussi aux textes, aux traits et aux images pleines, comme dans l'exemple 23-21 qui dessine du texte, une image et un objet créé à partir d'un chemin, avec des ombres portées. La figure 23-20 montre les ombres qui se répartissent intelligemment autour des portions visibles d'images et non seulement autour de leurs limites de contour.

Exemple 23-21. Application d'ombres portées dans les dessins d'un canevas

```
monimage     = new Image()
monimage.src = 'apple.png'

orig     = 95
pointes  = 21
dist     = Math.PI / pointes * 2
echelle1 = 75
echelle2 = 50

monimage.onload = function()
{
  context.beginPath()

  for (j = 0 ; j < pointes ; ++j)
  {
    x = Math.sin(j * dist)
    y = Math.cos(j * dist)
    context.lineTo(orig + x * echelle1, orig + y * echelle1)
    context.lineTo(orig + x * echelle2, orig + y * echelle2)
  }

  context.closePath()

  context.shadowOffsetX = 5
  context.shadowOffsetY = 5
  context.shadowBlur    = 6
  context.shadowColor   = '#444'
  context.fillStyle     = 'red'
  context.stroke()
  context.fill()

  context.shadowOffsetX = 2
  context.shadowOffsetY = 2
  context.shadowBlur    = 3
  context.shadowColor   = 'yellow'
  context.font          = 'bold 36pt Times'
  context.textBaseline  = 'top'
  context.fillStyle     = 'green'
  context.fillText('En vente!', 200, 5)

  context.shadowOffsetX = 3
  context.shadowOffsetY = 3
  context.shadowBlur    = 5
  context.shadowColor   = 'black'
  context.drawImage(monimage, 245, 45)
}
```

Figure 23-20. Ombres portées sous différents types d'objets dessinés

Modifier au niveau des pixels

Le canevas HTML5 offre une pléthore de méthodes de dessin et de tracé mais il favorise aussi le travail directement au niveau des pixels, avec un trio de puissantes méthodes dédiées à cet effet.

Méthode getImageData

La méthode `getImageData` permet d'extraire une portion (ou la totalité) d'un canevas pour modifier les données récupérées et, ceci, de la manière que vous souhaitez, puis de l'enregistrer au même endroit ou ailleurs dans le canevas.

Pour illustrer le fonctionnement de ceci, l'exemple 23-22 commence par charger une image prête à l'emploi et la dessine dans le canevas. Il en lit ensuite les données et les dépose dans un objet nommé `idata`, où il calcule la moyenne de toutes les couleurs et change chaque pixel en une échelle de gris, puis il décale légèrement chaque couleur pour obtenir de la sépia, comme illustré à la figure 23-21.

Exemple 23-22. Manipulation de données d'image

```
monimage     = new Image()
monimage.src = 'photo.jpg'

monimage.onload = function()
{
  context.drawImage(monimage, 0, 0)
  idata = context.getImageData(0, 0, monimage.width, monimage.height)

  for (y = 0 ; y < monimage.height ; ++y)
  {
    for (x = 0 ; x < monimage.width ; ++x)
    {
      pos     = y * monimage.width * 4 + x * 4
      moyenne =
      (
        idata.data[pos]     +
        idata.data[pos + 1] +
        idata.data[pos + 2]
      ) / 3

      idata.data[pos]     = moyenne + 50
      idata.data[pos + 1] = moyenne
      idata.data[pos + 2] = moyenne - 50
    }
  }
  context.putImageData(idata, 320, 0)
}
```

Figure 23-21. Conversion d'une image en sépia (l'affichage en échelle de gris montre peu de différences)

Tableau data

Cette manipulation d'image fonctionne grâce au tableau `data`, une propriété de l'objet `idata` retourné par l'appel à `getImageData`. Cette méthode renvoie un tableau qui contient toutes les données des pixels en composantes de rouge, vert, bleu et transparence alpha. Par conséquent, il faut quatre données (octets) pour stocker chaque pixel en couleur.

Toutes les données sont stockées en séquence dans le tableau `data`, de telle façon que la valeur de rouge est suivie de celle de vert, suivie de celle de bleu, puis du canal alpha ; ensuite vient la valeur de rouge du pixel suivant et ainsi de suite. Cela donne, pour le pixel à l'emplacement 0,0 :

```
idata.data[0] // Niveau de rouge
idata.data[1] // Niveau de vert
```

```
idata.data[2] // Niveau de bleu
idata.data[3] // Niveau d'alpha
```

Le pixel de l'emplacement 1,0 suit :

```
idata.data[4] // Niveau de rouge
idata.data[5] // Niveau de vert
idata.data[6] // Niveau de bleu
idata.data[7] // Niveau d'alpha
```

Le même principe se poursuit jusqu'à atteindre le pixel le plus à droite, le 320e, à l'emplacement 319 de la 1re rangée, de rang 0, donc son emplacement est 319,0. À ce stade, nous multiplions la valeur 319 par 4 (le nombre de données de chaque pixel) pour parvenir aux éléments suivants du tableau qui contiennent les données de ce pixel :

```
idata.data[1276] // Niveau de rouge
idata.data[1277] // Niveau de vert
idata.data[1278] // Niveau de bleu
idata.data[1279] // Niveau d'alpha
```

Ensuite, le pointeur passe de nouveau au premier pixel mais, cette fois, de la rangée suivante, de rang 1, donc à l'emplacement 0,1. Comme chaque rangée comporte 320 pixels en largeur, l'emplacement 0,1 correspond au décalage de (0 × 4) + (1 × 320 × 4), soit 1 280 :

```
idata.data[1280] // Niveau de rouge
idata.data[1281] // Niveau de vert
idata.data[1282] // Niveau de bleu
idata.data[1283] // Niveau d'alpha
```

Donc, si les données de l'image sont stockées dans `idata`, la largeur totale de l'image en pixels est dans `w` et l'emplacement du pixel à atteindre est défini par `x` et `y`, alors les formules à utiliser pour accéder directement aux données dans le tableau sont :

```
red   = idata.data[x * 4 + y * w * 4    ]
green = idata.data[x * 4 + y * w * 4 + 1]
blue  = idata.data[x * 4 + y * w * 4 + 2]
alpha = idata.data[x * 4 + y * w * 4 + 3]
```

Sachant cela, pour créer l'effet sépia de la figure 23-12, nous ne prenons que les composantes rouge, vert et bleu de chaque pixel et nous en calculons la moyenne, comme suit (où `pos` est la variable du pointeur de position du pixel courant dans le tableau) :

```
moyenne =
(
  idata.data[pos] +
  idata.data[pos + 1] +
  idata.data[pos + 2]
) / 3
```

Comme nous avons dans `moyenne` la valeur de couleur moyenne (obtenue par l'addition des trois valeurs de couleur et la division du résultat par 3), nous écrivons cette valeur dans chacune des trois composantes de couleur du pixel, mais nous amplifions la valeur de rouge en lui ajoutant 50, et nous réduisons la portion de bleu en lui retirant la même quantité :


```
idata.data[pos]     = moyenne + 50
idata.data[pos + 1] = moyenne
idata.data[pos + 2] = moyenne - 50
```

Le résultat de cette augmentation du niveau de rouge et de la réduction de niveau de bleu de chaque pixel donne l'aspect sépia, tandis que si nous n'écrivions que la valeur moyenne dans chaque composante de couleur, l'image obtenue serait en noir et blanc (ou plutôt, en niveaux de gris).

Si vous vous intéressez à d'autres manipulations plus évoluées d'images, reportez-vous à *Halfpap* ou à *HTML5 Rocks*, qui exposent tous deux (en anglais) l'utilisation des convolutions dans un canevas HTML5.

Méthode putImageData

Dès que vous avez modifié les données de l'image dans le tableau selon vos exigences, il ne reste plus qu'à les récrire dans le canevas, à l'aide de la méthode `putImageData`. Donnez-lui l'objet `idata` et les coordonnées du canevas où doit aboutir le coin supérieur gauche de l'image, comme dans l'exemple précédent, soit comme suit, où l'image vient se placer juste à droite de l'originale :

```
context.putImageData(idata, 320, 0)
```

Si vous souhaitez modifier seulement une partie d'un canevas, il n'est pas nécessaire de lire la totalité de celui-ci. Prenez juste la section qui contient la zone qui vous intéresse. Ensuite, vous n'êtes pas obligé de récrire les données traitées à l'emplacement où vous les avez obtenues car vous pouvez écrire les données dans n'importe quelle zone du canevas, ou même d'un autre canevas.

Méthode createImageData

Vous n'avez aucune obligation de créer un objet de données d'image à partir d'un canevas. Vous pouvez en effet créer un nouvel objet de données d'image avec des données vides, à l'aide de la méthode `createImageData`. L'exemple suivant crée un objet vide d'une largeur de 320 pixels sur une hauteur de 240 pixels :

```
idata = createImageData(320, 240)
```

Mais vous pouvez aussi créer un nouvel objet à partir d'un objet existant, comme suit :

```
nouvelobjetimagedata = createImageData(imagedata)
```

Ce que vous faites de ces objets ne dépend que de vous : ajoutez des données de pixels, modifiez-les, collez-les dans des canevas, créez d'autres objets à partir d'eux et ainsi de suite.

Effets graphiques avancés

Parmi ses fonctionnalités avancées, le canevas HTML5 permet d'apporter des effets divers de composition et de transparence, ainsi que des transformations puissantes, telles que le changement d'échelle, l'étirement et la rotation.

Propriété globalCompositeOperation

Douze méthodes différentes sont disponibles pour affiner la manière dont vous placez un objet dans un canevas, en tenant compte des objets existants et à venir. Ces méthodes portent le nom général d'options de *composition* et elles s'appliquent comme suit :

```
context.globalCompositeOperationProperty = 'source-over'
```

Les types de composition sont les suivants (voir figure 23-22) :

`source-over`
: C'est l'option par défaut. L'image source est copiée sur l'image de destination.

`source-in`
: Seules les portions de l'image source qui doivent apparaitre dans la destination sont montrées et l'image de destination est supprimée. Toute transparence alpha dans l'image source provoque la disparition de la destination sous elle.

`source-out`
: Seules les portions de l'image source qui ne doivent pas apparaitre dans la destination sont montrées et l'image de destination est supprimée. Toute transparence alpha dans l'image source provoque la disparition de la destination sous elle.

`source-atop`
: L'image source est affichée lorsqu'elle se superpose à la destination. L'image de destination s'affiche où l'image de destination est opaque et où l'image source est transparente. Les autres régions sont transparentes.

`destination-over`
: L'image source est dessinée sous l'image de destination.

`destination-in`
: L'image de destination s'affiche là où l'image source et l'image de destination se chevauchent, mais pas dans les zones de transparence de l'image source. L'image source ne s'affiche pas.

`destination-out`
: Seules les portions de la destination en dehors des sections non transparentes de l'image source apparaissent. L'image source ne s'affiche pas.

`destination-atop`
: L'image source s'affiche là où la destination n'est pas affichée. Dans les zones où la destination et la source se chevauchent, l'image de destination s'affiche. Toute zone de transparence dans l'image source empêche l'affichage de l'image de destination dans cette zone.

`lighter`
: La somme de la source et destination s'applique de telle sorte que dans les zones où elles ne se chevauchent pas, elles s'affichent normalement ; tandis que dans les zones où elles se superposent, la somme des deux images apparait mais légèrement plus éclairée (lighter).

`darker`
: La somme de la source et destination s'applique de telle sorte que dans les zones où elles ne se chevauchent pas, elles s'affichent normalement ; tandis que dans les zones où elles se superposent, la somme des deux images apparait mais légèrement plus assombrie (darker).

`copy`
: L'image source est copiée sur la destination. Toute zone de transparence dans l'image source empêche l'affichage de l'image de destination dans cette zone.

`xor`
: Dans les zones où les images source et de destination ne se chevauchent pas, elles s'affichent normalement. Dans les zones où elles se superposent, leurs couleurs subissent une opération `ou exclusif`.

L'exemple 23-23 illustre l'effet de chacun de ces types de composition et crée 12 canevas différents, avec chacun deux objets (un cercle plein et l'image du yin et du yang) décalés légèrement entre eux et avec un chevauchement.

Exemple 23-23. Démonstration des 12 effets de composition

```
image     = new Image()
image.src = 'image.png'

image.onload = function()
{
  types =
  [
    'source-over',      'source-in',        'source-out',
    'source-atop',      'destination-over', 'destination-in',
    'destination-out',  'destination-atop', 'lighter',
    'darker',           'copy',             'xor'
  ]

  for (j = 0 ; j < 12 ; ++j)
  {
    canvas               = O('c' + (j + 1))
    context              = canvas.getContext('2d')
    S(canvas).background = 'lightblue'
    context.fillStyle    = 'red'
```

Figure 23-23. Modifications d'échelle pour agrandir, puis réduire la taille de l'image

Si vous examinez attentivement la figure, vous constatez que la copie de l'image sous l'originale est un peu plus floue, à cause de l'agrandissement et de la réduction successive.

Si vous appliquez une valeur d'échelle négative à un des paramètres, vous pouvez inverser un élément dans une ou deux directions en même temps que (ou à la place de) la modification d'échelle. Ainsi, l'instruction suivante crée une image en miroir horizontal :

```
context.scale(-1, 1)
```

Méthodes save et restore

Lorsque vous devez appliquer plusieurs opérations de modification d'échelle sur des éléments de dessin différents, non seulement vous introduisez du flou dans les résultats, mais cela demande beaucoup de temps pour calculer une augmentation d'échelle horizontale de trois puis de réduire celle-ci à `0.33` (et l'augmentation verticale de deux puis la réduction de `0.5` pour revenir à l'original).

Par conséquent, il est préférable dans ce cas-ci d'enregistrer le contexte courant à l'aide de la méthode `save`, puis de le restituer plus tard à l'aide de la méthode `restore`. Examinez ce qui suit, qui peut remplacer le code de l'exemple 23-24 :

```
context.drawImage(monimage, 0, 0)
context.save()
context.scale(3, 2)
context.drawImage(monimage, 40, 0)
context.restore()
context.drawImage(monimage, 0, 100)
```


Les méthodes `save` et `restore` s'avèrent très puissantes parce qu'elles ne s'appliquent pas qu'aux échelles d'image, mais également à toutes les propriétés suivantes et, de ce fait, elles permettent d'enregistrer toutes les propriétés courantes pour les restaurer ultérieurement : `fillStyle`, `font`, `globalAlpha`, `globalCompositeOperation`, `lineCap`, `lineJoin`, `lineWidth`, `miterLimit`, `shadowBlur`, `shadowColor`, `shadowOffsetX`, `shadowOffsetY`, `strokeStyle`, `textAlign` et `textBaseline`. Les propriétés des méthodes suivantes sont également prises en charge par `save` et `restore` : `scale`, `rotate`, `translate` et `transform`.

Méthode rotate

La méthode `rotate` permet d'imposer un angle de rotation exprimé en radians à appliquer pour placer un objet (ou de n'importe quelle méthode de dessin) dans le canevas. Un radian équivaut à 180 / π, soit environ 57 degrés d'arc.

La rotation se déroule autour du point d'origine du canevas, soit par défaut son coin supérieur gauche. Mais vous verrez sous peu comment changer cette référence. L'exemple 23-25 affiche quatre fois l'image du yin et du yang avec des rotations successives de chaque copie incrémentées de `Math.PI / 25` radians.

Exemple 23-25. Rotation d'une image

```
for (j = 0 ; j < 4 ; ++j)
{
  context.drawImage(monimage, 20 + j * 120 , 20)
  context.rotate(Math.PI / 25)
}
```

Comme l'indique la figure 23-24, le résultat n'est pas tout à fait celui escompté, car l'image n'a pas tourné que sur elle-même. Les rotations se produisent autour de l'origine du canevas, à l'emplacement 0,0. De plus, chaque rotation vient s'ajouter à la précédente. Cependant, pour corriger ces soucis, vous pouvez toujours faire appel à la méthode `translate` (pour déplacer l'objet), accompagnée des méthodes `save` et `restore`.

Le radian est une unité de mesure d'angles très appropriée car le cercle complet compte exactement π × 2 radians. Ainsi, le demi-cercle comporte π radians, π ÷ 2 radians correspondent à un quart de cercle, π ÷ 2 × 3 (ou π × 1,5) radians correspondent à trois-quarts de cercle et ainsi de suite. Pour vous éviter de retenir la valeur de π, n'hésitez pas à utiliser la valeur de `Math.PI`.

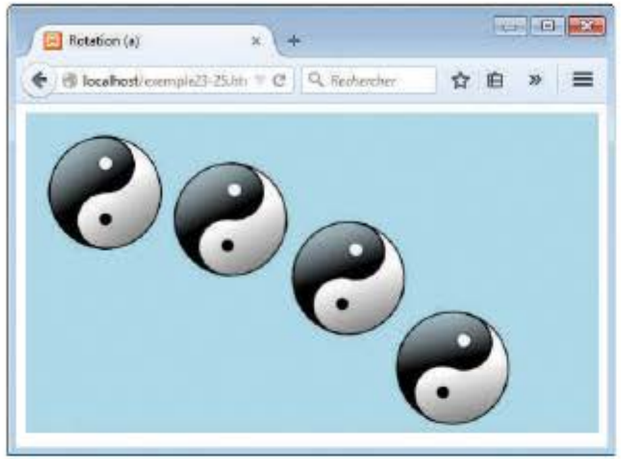


Figure 23-24. Quatre rotations différentes d'une image

Méthode translate

Pour modifier l'origine d'une rotation, appelez par exemple la méthode `translate` pour décaler l'objet ailleurs, soit au sein (ou en dehors) du canevas, ou plus généralement, quelque part dans l'emplacement de destination de l'objet (souvent son centre).

L'exemple 23-26 effectue ce déplacement avant chaque appel à `rotate` pour produire l'effet probablement attendu. De plus, les appels aux méthodes `save` avant et `restore` après chaque opération garantissent l'application indépendante de chaque rotation et non une rotation composée avec la précédente.

Exemple 23-26. Rotation d'objets en place

```
w = monImage.width          // Largeur
h = monImage.height         // Hauteur

for (j = 0 ; j < 4 ; ++j)
{
  context.save()
  context.translate(20 + w / 2 + j * (w + 20), 20 + h / 2)

  context.rotate(Math.PI / 5 * j)
  context.drawImage(monImage, -(w / 2), -(h / 2))
  context.restore()
}
```

Dans cet exemple, avant chaque rotation, nous enregistrons (`save`) le contexte et nous déplaçons (`translate`) l'origine à un emplacement correspondant exactement au centre

de la zone de dessin de chaque image. Nous appliquons ensuite la rotation (`rotate`) et nous dessinons l'image à gauche et en haut du nouvel emplacement de la nouvelle origine à l'aide de valeurs négatives, pour que le centre de l'image coïncide avec le point de l'origine. La figure 23-25 montre les résultats obtenus.

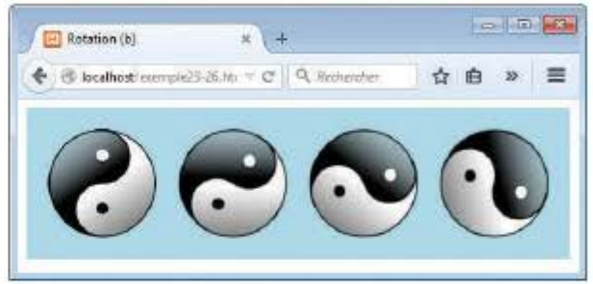


Figure 23-25. Quatre rotations différentes d'une image

Pour récapituler, lorsque vous voulez appliquer une rotation ou une transformation (voir ci-après) à un objet sur place, respectez les étapes suivantes :

1. Enregistrez le contexte.
2. Déplacez l'origine du canevas au centre de la zone où l'objet doit venir se placer.
3. Appliquez l'instruction de rotation ou de transformation.
4. Dessinez l'objet avec n'importe quelle méthode de dessin prise en charge, avec des valeurs négatives pour les coordonnées de l'emplacement, égales à la moitié de la largeur de l'objet vers la gauche et la moitié de la hauteur de l'objet vers le haut.
5. Restaurez le contexte pour revenir à la situation initiale.

Méthode transform

Lorsque vous avez épuisé toutes les autres fonctionnalités du canevas et n'avez toujours pas trouvé de quoi manipuler les objets de la manière voulue, il vous reste à vous tourner vers la méthode `transform`. Celle-ci permet d'appliquer une matrice de transformation aux objets que vous dessinez dans le canevas, avec une multitude de possibilités et de puissantes fonctionnalités qui combinent des modifications d'échelle et des rotations en une seule instruction.

La matrice de transformation utilisée par cette méthode comporte 3×3, soit neuf valeurs, Mais seulement six de celles-ci sont fournies en externe à la méthode `transform`. Donc, plutôt que d'expliquer comment la multiplication de cette matrice fonctionne, il nous suffit d'examiner les effets de ces six arguments, qui sont les suivants, dans l'ordre :

1. Échelle horizontale
2. Inclinaison horizontale

3. Inclinaison verticale
4. Échelle verticale
5. Déplacement horizontal
6. Déplacement vertical

Ces valeurs s'appliquent de nombreuses manières. Pour émuler la méthode `scale` de l'exemple 23-24, par exemple, remplacez l'appel :

```
context.scale(3, 2)
```

par le suivant :

```
context.transform(3, 0, 0, 2, 0, 0)
```

De la même manière, cet appel de l'exemple 23-26 :

```
context.translate(20 + w / 2 + j * (w + 20), 20 + h / 2)
```

équivaut au suivant :

```
context.transform(1, 0, 0, 1, 20 + w / 2 + j * (w + 20), 20 + h / 2)
```

Remarquez que les arguments d'échelle horizontale et verticale reçoivent les valeurs `1` pour garantir le résultat au 1:1, tandis que les valeurs d'inclinaison sont à `0` pour éviter toute inclinaison du résultat.

Vous pouvez même combiner les lignes de code précédentes pour appliquer un déplacement et un changement d'échelle en même temps :

```
context.transform(3, 0, 0, 2, 20 + w / 2 + j * (w + 20), 20 + h / 2)
```

En ce qui concerne les arguments d'inclinaison, vous vous attendez avec raison à obtenir un élément incliné dans la direction précisée, par exemple pour obtenir un rhombe (losange sans angle droit) à partir d'un carré.

Ainsi, l'exemple 23-27 dessine l'image du yin et du yang sur le canevas, suivie d'une copie inclinée, créée à l'aide de la méthode `transform`. La valeur d'inclinaison peut être négative, égale à zéro ou positive, mais nous choisissons `1` pour la valeur horizontale, ce qui provoque l'inclinaison du bas de l'image d'une largeur d'image vers la droite et tire tout le reste proportionnellement (figure 23-26).

Exemple 23-27. Création d'une image originale et d'une autre inclinée

```
context.drawImage(monimage,  20, 20)
context.transform(1, 0, 1, 1, 0, 0)
context.drawImage(monimage, 140, 20)
```


Figure 23-26. Inclinaison horizontale d'une image vers la droite

La méthode `transform` permet même de faire tourner un objet, comme avec `rotate`, si vous lui fournissez une valeur négative d'inclinaison et une valeur positive opposée, mais soyez prudent car cela modifie la taille de l'élément et vous devrez donc ajuster simultanément les arguments d'échelle. De plus, vous devez vous souvenir de déplacer l'origine. C'est pourquoi il est préférable de vous en tenir à la méthode `rotate` pour ce genre d'opération, tant que vous ne maitrisez pas parfaitement l'utilisation de `transform`.

Méthode setTransform

Comme alternative à l'utilisation des méthodes `save` et `restore`, vous pouvez définir une transformation absolue, qui a pour effet de réinitialiser la matrice de transformation et d'appliquer ensuite les valeurs fournies. Utilisez `setTransform` de la même manière que `transform`, comme dans l'exemple suivant, qui applique une inclinaison horizontale positive de valeur `1` :

```
context.setTransform(1, 0, 1, 1, 0, 0)
```

Pour de plus amples informations sur les matrices de transformation, lisez l'article complet (en anglais) de Wikipedia (la version française étant réduite). *http://bit.ly/transform-mx*

En résumé

Au moment de la rédaction de ces lignes, la norme HTML5 n'était pas encore prise en charge à 100 % parmi tous les navigateurs principaux mais, heureusement, l'essentiel des fonctionnalités du canevas l'est. Même s'il faut s'attendre à en voir plus, comme les contextes en trois dimensions, le canevas HTML5 représente déjà un nouvel acquis immense des développeurs web pour continuer à réaliser des sites web plus grands,

meilleurs, plus professionnels et convaincants. Au chapitre suivant, nous examinons deux autres améliorations du HTML5 : l'audio et la vidéo au sein du navigateur et sans utiliser de module d'extension.

Questions

1. Comment crée-t-on un élément canevas en HTML ?
2. Comment donner à JavaScript accès à l'élément canevas ?
3. Comment débutez et terminez-vous la création d'un chemin de canevas ?
4. Quelle méthode permet d'extraire des données d'un canevas vers une image ?
5. Comment faire pour créer des remplissages en dégradé de plus de deux couleurs ?
6. Comment pouvez-vous ajuster la largeur des traits lorsque vous dessinez ?
7. Quelle méthode utilisez-vous pour préciser une section d'un canevas pour que les dessins à venir n'apparaissent qu'au sein de cette zone ?
8. Comment faire pour dessiner une courbe complexe avec deux points d'attraction ?
9. Combien de données la méthode `getImageData` renvoie-t-elle pour chaque pixel ?
10. Quels sont les deux paramètres de la méthode `transform` qui s'appliquent aux opérations de modification d'échelle ?

Retrouvez les réponses du chapitre 23 dans l'annexe A.

CHAPITRE 24

Audio et vidéo en HTML5

L'une des principales forces de l'internet trouve sa source dans la demande insatiable de la part d'utilisateurs, de plus en plus de multimédia sous forme d'audio et de vidéo. Au début, la bande passante était si précieuse qu'il n'était pas envisageable d'y trouver de la diffusion en flux continu (ou *streaming*) et il fallait des heures, voire des jours pour télécharger une piste audio, sans parler de la vidéo.

Le cout élevé de la bande passante et la faible disponibilité des modems rapides ont conduit au développement d'algorithmes plus rapides et efficaces, comme l'audio en MP3 et la vidéo MPEG, mais même là, le seul moyen de télécharger des fichiers dans un délai raisonnable résidait dans la réduction drastique de leur qualité.

Un de mes premiers projets sur l'internet, en 1997, fut pour la première radio en ligne du Royaume-Uni. En fait, il s'agissait plutôt d'un balado (*podcast*), avant même que le terme soit créé, parce que nous produisons une émission quotidienne d'une demi-heure, pour la compresser en 8 bits à 11 kHz et en mono à l'aide d'un algorithme développé au départ pour la téléphonie, donc cela donnait une qualité tout juste digne du téléphone ou pire. Néanmoins, nous avons obtenu des milliers d'auditeurs qui devaient télécharger l'émission, puis l'écouter à l'aide d'une fenêtre contextuelle de navigateur contenant un module d'extension, tout en surfant sur les sites évoqués dans l'émission.

Heureusement pour nous et pour quiconque diffusait du multimédia, il est vite devenu possible d'offrir une bien meilleure qualité audio et vidéo, mais cela demandait encore et toujours de télécharger et d'installer un module d'extension de lecture de média. Flash a acquis le plus grand succès parmi ces lecteurs, après avoir relégué ses rivaux, comme RealAudio, aux oubliettes. Mais il a acquis aussi une mauvaise réputation, car il fut à la source de nombreux plantages des navigateurs et nécessitait des mises à niveau constantes.

Ainsi, un consensus général s'est établi pour considérer que l'avenir était à une normalisation web pour la prise en charge du multimédia directement dans le navigateur. Évidemment, les développeurs de navigateurs tels que Microsoft et Google ont émis des visions différentes de ce à quoi ces standards devaient ressembler. Toutefois, lorsque toute la poussière est retombée, ils se sont mis d'accord sur un ensemble restreint de types de fichiers que tous les navigateurs doivent pouvoir lire en natif, et ce sous-ensemble a été introduit dans la normalisation du HTML5.

Il devient donc enfin possible, tant que vous encodez votre audio et votre vidéo dans quelques formats différents, de téléverser du multimédia sur un serveur web et de placer quelques balises HTML dans une page web pour que tous les principaux navigateurs puissent lire le multimédia sur un ordinateur de bureau, une tablette ou un téléphone intelligent, sans devoir télécharger un module d'extension ou se livrer à des modifications techniques.

Le marché comprend encore de nombreux navigateurs anciens. Par conséquent, Flash conserve une certaine importance pour les prendre en charge. Ce chapitre vous explique comment faire appel à Flash en guise de solution de repli, par rapport à l'audio et la vidéo intégrées de HTML5, pour cibler le plus de combinaisons de matériel et de logiciels possible.

À propos des codecs

Le terme *codec* signifie *co*deur/*déc*odeur. Il décrit la fonctionnalité offerte par des logiciels qui encodent et décodent du multimédia comme de l'audio et de la vidéo. HTML5 dispose d'un certain nombre d'ensembles différents de codecs, qui dépendent du navigateur utilisé.

La liste suivante énumère et décrit les codecs pris en charge par la balise HTML5 `<audio>` (ainsi que par la vidéo HTML5 lorsque celle-ci comporte de l'audio) :

AAC
: Ce codec audio, qui signifie *Advanced Audio Encoding*, est utilisé par le magasin iTunes d'Apple, est une technologie sous licence propriétaire d'Apple et est pris en charge par Apple, Google et Microsoft. Il utilise généralement l'extension de fichier *.aac*. Son type MIME est `audio/aac`.

MP3
: Ce codec audio, qui signifie *MPEG Audio Layer 3*, est disponible depuis de nombreuses années. Bien que le terme soit souvent utilisé (de façon incorrecte) pour désigner n'importe quel type d'audio numérique, il s'agit au sens strict d'une technologie sous licence propriétaire, prise en charge par Apple, Google, Mozilla Firefox et Microsoft. Il utilise l'extension de fichier .mp3 et son type MIME est `audio/mpeg`.

PCM
: Ce codec audio, qui signifie *Pulse Coded Modulation*, stocke les données complètes encodées par un convertisseur analogique-numérique. C'est le format utilisé pour le stockage de données sur les CD audio. Comme il n'effectue aucune compression, il est considéré comme un codec *sans perte* (*lossless*) et, de ce fait, ses fichiers sont généralement beaucoup plus volumineux que ceux des formats AAC et MP3. Il est pris en charge par Apple, Mozilla Firefox et Opera. Ce type de fichier porte généralement l'extension *.wav*. Son type MIME est `audio/wav` mais vous pouvez aussi le trouver sous la forme `audio/wave`.

Vorbis
: Désigné parfois sous le nom Ogg Vorbis, parce qu'il utilise généralement l'extension *.ogg*, ce codec audio n'est encombré d'aucun brevet restrictif ni de redevance de droits d'utilisation. Il est pris en charge par Google Chrome, Mozilla Firefox et Opera. Son type MIME est `audio/ogg` ou parfois `audio/oga`.

La liste suivante énumère les principaux systèmes d'exploitation et navigateurs, avec les types audio que leurs dernières versions prennent en charge :

- **Apple iOS** : AAC, MP3, PCM
- **Apple Safari** : AAC, MP3, PCM
- **Google Android** : 2.3+ AAC, MP3, Vorbis
- **Google Chrome** : AAC, MP3, Vorbis
- **Microsoft Internet Explorer** : AAC, MP3
- **Mozilla Firefox** : MP3, PCM, Vorbis
- **Opera** : PCM, Vorbis

L'inconvénient de ces différentes prises en charge des codecs par les navigateurs se situe dans la nécessité de toujours proposer au moins deux versions de chaque fichier audio pour garantir qu'il puisse être lu sur toutes les plateformes. Le sous-ensemble minimal comprend Vorbis pour Opera et aussi AAC ou MP3, au choix, pour les autres.

Élément <audio>

Pour satisfaire toutes les plateformes, vous devez enregistrer ou convertir votre contenu audio sous plusieurs codecs, puis les énumérer entre les balises `<audio>` et `</audio>`, comme dans l'exemple 24-1. Les balises imbriquées `<source>` contiennent ensuite les différents fichiers audio que vous proposez au navigateur. L'attribut `controls` impose l'affichage des commandes de lecture, pause et ainsi de suite, comme illustré à la figure 24-1.

Exemple 24-1. Insertion de trois types de fichiers audio

```
<audio controls>
  <source src='audio.m4a' type='audio/aac'>
  <source src='audio.mp3' type='audio/mp3'>
  <source src='audio.ogg' type='audio/ogg'>
</audio>
```

Figure 24-1. Lecture d'un fichier audio

Cet exemple inclut trois types d'audio différents, car c'est parfaitement admis et c'est d'ailleurs une bonne idée si vous voulez que chaque navigateur repère son format préféré au lieu de lui en imposer qu'il est capable de gérer, sans plus. L'exemple fonctionne sur toutes les plateformes, même si vous éludez l'un ou l'autre (mais pas les deux) des fichiers MP3 ou AAC.

L'élément `<audio>` et sa balise partenaire `<source>` prennent en charge les attributs suivants :

`autoplay`
: Provoque la lecture audio dès que le fichier est prêt.

`controls`
: Impose l'affichage du volet de commandes.

`loop`
: Impose la lecture audio en boucle.

`preload`
: Impose le chargement de l'audio avant même que l'utilisateur ne sélectionne le bouton de lecture.

`src`
: Indique l'emplacement source du fichier audio.

`type`
: Précise le codec à utiliser pour le fichier audio.

Si vous éludez simultanément l'attribut `controls` dans la balise `<audio>` et l'attribut `autoplay`, alors l'audio n'est pas lu et aucune commande de lecture n'apparaît, donc l'utilisateur ne peut cliquer nulle part pour démarrer la lecture audio. Cette situation ne vous laisse aucune autre option que de fournir cette fonctionnalité en JavaScript, comme dans l'exemple 24-2 (où le code nécessaire figure en gras), qui fournit la possibilité de lire et de mettre l'audio en pause, comme illustré à la figure 24-2.

Exemple 24-2. Lecture d'audio à l'aide de JavaScript

```
<!DOCTYPE html>
<html>
  <head>
    <title>Lecture audio en JavaScript</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <audio id='monaudio'>
      <source src='audio.m4a' type='audio/aac'>
      <source src='audio.mp3' type='audio/mp3'>
      <source src='audio.ogg' type='audio/ogg'>
    </audio>

    <button onclick='playaudio()'>Lecture</button>
    <button onclick='pauseaudio()'>Pause</button>

    <script>
      function playaudio()
      {
        O('monaudio').play()
      }
      function pauseaudio()
      {
        O('monaudio').pause()
      }
    </script>
  </body>
</html>
```

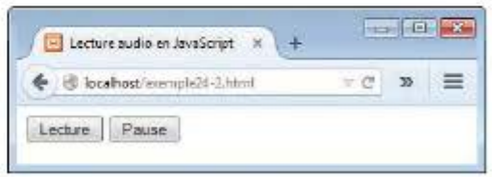


Figure 24-2. L'audio HTML5 contrôlé par JavaScript

Le code de cet exemple appelle la méthode `play` ou `pause` de l'élément `monaudio` lors d'un clic sur ces boutons.

Tenir compte des navigateurs non-HTML5

Il est probable qu'il faille encore prendre en charge des navigateurs plus anciens dans un avenir plus ou moins prévisible et donc fournir une solution de remplacement avec le lecteur Flash. L'exemple 24-3 montre comment procéder pour exploiter un module d'extension Flash (fourni aux côtés des exemples, téléchargeables sur le site d'accompagnement du livre) nommé *audioplayer.swf*. Le code à ajouter figure en gras.

Exemple 24-3. Fourniture d'une solution de repli en Flash pour les navigateurs non-HTML5

```
<audio controls>
  <object type="application/x-shockwave-flash"
    data="audioplayer.swf" width="300" height="30">
    <param name="FlashVars"
      value="mp3=audio.mp3&showstop=1&showvolume=1">
  </object>

  <source src='audio.m4a' type='audio/aac'>
  <source src='audio.mp3' type='audio/mp3'>
  <source src='audio.ogg' type='audio/ogg'>
</audio>
```

Nous tirons ici parti du fait que les navigateurs qui ne reconnaissent pas HTML5 prennent en compte tout ce qui se trouve dans la balise `<audio>` (à l'exception des éléments `<source>` qu'ils ignorent purement et simplement). Par conséquent, insérer un élément `<object>` à l'intérieur qui appelle un lecteur Flash garantit que tout navigateur non compatible HTML5 a une chance de lire l'audio, à condition du moins que Flash y soit installé, ce qu'illustre la figure 24-3.

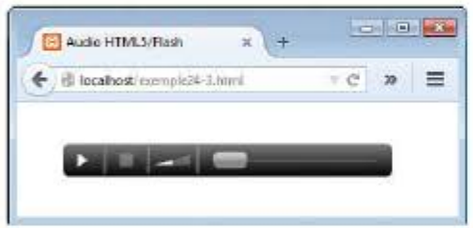


Figure 24-3. Le lecteur audio Flash est chargé

Le lecteur audio utilisé dans cet exemple, *audioplayer.swf*, prend les arguments et valeurs suivants pour l'attribut `FlashVars` de l'élément `<param>` :

`mp3`
: L'URL d'un fichier audio MP3.

`showstop`
: Si égal à 1, affiche le bouton Arrêt ; sinon, il est masqué.

`showvolume`
: Si égal à 1, affiche la barre de volume ; sinon, elle est masquée.

Comme pour nombre d'éléments, vous pouvez redimensionner l'objet, par exemple à 300×30 pixels, en précisant ces valeurs pour ses attributs `width` et `height`.

Élément <video>

La lecture de vidéo en HTML5 s'apparente fort à la lecture audio. Indiquez la balise `<video>` et fournissez les éléments `<source>` des contenus à lire. L'exemple 24-4 montre comment procéder avec trois types de codecs différents, pour obtenir le résultat illustré à la figure 24-4.

Exemple 24-4. Lecture de vidéo en HTML5

```
<video width='560' height='320' controls>
  <source src='video.mp4'  type='video/mp4'>
  <source src='video.webm' type='video/webm'>
  <source src='video.ogv'  type='video/ogg'>
</video>
```

Figure 24-4. Lecture de vidéo en HTML5

Codecs vidéo

Tout comme en audio, plusieurs codecs vidéo sont disponibles, avec des différences de prise en charge par les navigateurs. Ces codecs sont fournis dans les conteneurs suivants :

MP4
: Soumis à une licence, ce format de conteneur multimédia fait partie de la norme MPEG-4, prise en charge par Apple, Microsoft et, dans une moindre mesure par Google, puisque celui-ci possède son propre format de conteneur WebM. Son type MIME et `video/mp4`.

OGG
: Un format libre de conteneur maintenu par la *Xiph.Org Foundation*. Les créateurs du format OGG affirment qu'il n'est pas restreint par des brevets sur des logiciels et qu'il est conçu pour produire une diffusion en flux continu efficace et une manipulation de multimédia de haute qualité. Son type MIME est `video/ogg` ou parfois `video/ogv`.

WebM
: Un format audio-vidéo conçu pour assurer un format de compression vidéo libre de redevance de droits d'utilisation pour l'utilisation avec la vidéo HTML5. Le développement de ce projet est parrainé par Google. Il en existe deux versions : VP8 et la VP9 plus récente. Son type mime est `video/webm`.

Ceux-ci peuvent contenir un des codecs vidéo suivants :

H.264
: Un codec vidéo breveté propriétaire dont la lecture est gratuite pour l'utilisateur final, mais peut exiger des droits d'utilisation pour toutes les étapes du codage et de la transmission. Au moment de la rédaction de ces lignes, tous parmi Apple, Google, Mozilla Firefox et Microsoft Internet Explorer prennent ce codec en charge, contrairement à Opera.

Theora
: Ce codec vidéo est libre de brevet et de droits d'utilisation à tous les niveaux du codage, de la transmission et de la lecture. Il est pris en charge par Google Chrome, Mozilla Firefox et Opera.

VP8
: Ce codec vidéo est comparable à Theora mais appartient à Google, qui l'a publié sous forme de source libre, ce qui le rend libre de tout droit d'utilisation. Il est pris en charge par Google Chrome, Mozilla Firefox et Opera.

VP9
: Comme VP8 mais plus puissant et il ne nécessite que la moitié de débit binaire.

La liste suivante énumère les principaux systèmes d'exploitation et navigateurs, avec les types vidéo que leurs dernières versions prennent en charge :

- **Apple iOS** : MP4/H.264
- **Apple Safari** : MP4/H.264
- **Google Android** : MP4, OGG, WebM/H.264, Theora, VP8
- **Google Chrome** : MP4, OGG, WebM/H.264, Theora, VP8, VP9

- **Internet Explorer** : MP4/H.264
- **Mozilla Firefox** : MP4, OGG, WebM/H.264, Theora, VP8, VP9
- **Opera** : OGG, WebM/Theora, VP8

La lecture de cette dernière liste montre clairement la prise en charge unanime de MP4/H.264, sauf par le navigateur Opera. Par conséquent, si vous décidez de négliger le petit pourcent ou quelques de ses utilisateurs (en espérant qu'Opera adopte rapidement ce format également), vous ne devez fournir qu'un seul type de fichier : MP4/H.264. Cependant, pour garantir une visibilité maximale, envisagez tout de même de produire une version codée en OGG/Theora ou OGG/VP8 (mais pas VP9 car celui-ci n'est pas encore adopté par Opera).

En conséquence, le fichier *video.webm* repris par l'exemple 24-4 n'est pas strictement nécessaire, mais il montre au moins comment ajouter tous les types de fichiers différents pour offrir aux navigateurs la possibilité de lire les formats qu'ils préfèrent.

L'élément `<video>` et sa balise partenaire `<source>` prennent en charge les attributs suivants :

`autoplay`
: Provoque la lecture automatique de la vidéo dès qu'elle est prête.

`controls`
: Impose l'affichage du volet de commandes.

`height`
: Précise la hauteur d'affichage de la vidéo.

`loop`
: Impose la lecture en boucle de la vidéo.

`muted`
: Applique la sourdine à la partie audio.

`poster`
: Permet de sélectionner une image à afficher dans la zone de lecture de la vidéo.

`preload`
: Impose le chargement de la vidéo avant même que l'utilisateur sélectionne le bouton de lecture.

`src`
: Précise l'emplacement du fichier source de la vidéo.

`type`
: Précise le codec utilisé lors de la création de la vidéo.

`width`
: Précise la largeur d'affichage de la vidéo.

Si vous préférez piloter la lecture de vidéo à partir de JavaScript, utilisez du code comme celui de l'exemple 24-5, où le code supplémentaire est indiqué en gras, dont la figure 24-5 illustre les résultats.

Exemple 24-5. Contrôle de la lecture vidéo en JavaScript

```
<!DOCTYPE html>
<html>
  <head>
    <title>Lecture de vidéo en JavaScript</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <video id='mavideo' width='560' height='320'>
      <source src='video.mp4'  type='video/mp4'>
      <source src='video.webm' type='video/webm'>
      <source src='video.ogv'  type='video/ogg'>
    </video><br>

    <button onclick='playvideo()'>Lecture</button>
    <button onclick='pausevideo()'>Pause</button>

    <script>
      function playvideo()
      {
        O('mavideo').play()
      }
      function pausevideo()
      {
        O('mavideo').pause()
      }
    </script>
  </body>
</html>
```


Figure 24-5. JavaScript mis en œuvre pour piloter la vidéo

Ce code est quasi identique à celui nécessaire pour contrôler l'audio en JavaScript. Appelez les méthodes `play` et `pause` de l'objet `mavideo` pour lire et mettre la vidéo en pause.

Prendre en charge les anciens navigateurs

Comme dans le cas de l'audio, les anciens navigateurs verront encore leur usage généralisé pour un moment, donc il faut envisager de proposer une solution de repli en vidéo Flash aux utilisateurs qui ne disposent pas d'un navigateur compatible HTML5. L'exemple 24-6 montre comment procéder (en gras) à l'aide du fichier *flowplayer.swf* (disponible avec les sources sur le site d'accompagnement du livre) et la figure 24-6 montre l'affichage obtenu dans un tel navigateur.

Exemple 24-6. Flash comme solution de repli pour les anciens navigateurs

```
<!DOCTYPE html>
<html>
  <head>
    <title>Vidéo HTML5/Flash</title>
  </head>
  <body>
    <video width='560' height='320' controls>
      <object width='560' height='320'
        type='application/x-shockwave-flash'
        data='flowplayer.swf'>
        <param name='video' value='flowplayer.swf'>
        <param name='flashvars'
```

```
          value='config={"clip": {
            "url": "http://localhost/video.mp4",
            "autoPlay":false, "autoBuffering":true}}'>
          </object>
        <source src='video.mp4'  type='video/mp4'>
        <source src='video.webm' type='video/webm'>
        <source src='video.ogv'  type='video/ogg'>
      </video>
    </body>
</html>
```

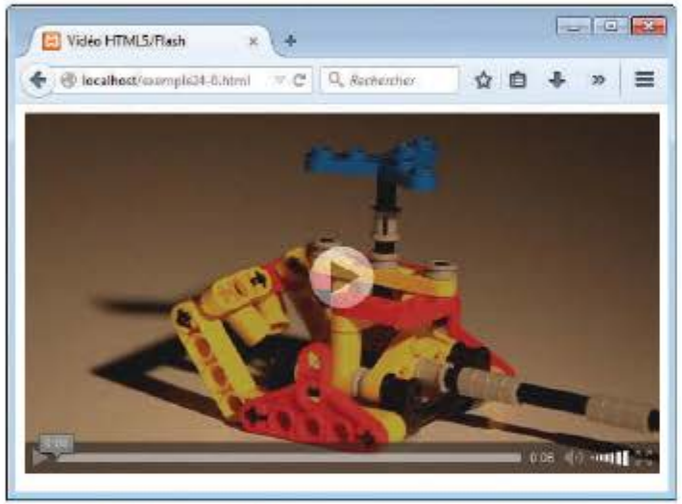


Figure 24-6. Flash offre une solution de repli pratique pour les navigateurs non-HTML5

Comme le lecteur vidéo Flash est assez susceptible en termes de sécurité, il ne peut pas lire de vidéo d'un système de fichiers local mais seulement à partir d'un serveur web. Par conséquent, l'URL indiquée doit obligatoirement citer un serveur web, soit *http://localhost/video.mp4* ici. Mais cela aurait pu être une URL de l'internet.

Les arguments à fournir dans l'attribut `flashvars` de l'élément `<param>` sont :

`url`
: L'URL d'un serveur web qui héberge le fichier *.mp4* à lire.

`autoPlay`
: Si vrai, la lecture débute automatiquement ; sinon, elle attend un clic sur le bouton de lecture.

`autoBuffering`
: La mise en cache automatique. Si vrai, pour réduire la mise en cache ultérieure sur des connexions lentes, la vidéo est chargée en suffisance avant sa lecture, en fonction du débit disponible.

Pour de plus amples informations sur le programme Flash *flowplayer* (et sa version HTML5), consultez *http://flowplayer.org* (en anglais).

En résumé

Les informations proposées dans ce chapitre permettent d'intégrer de l'audio et de la vidéo dans presque tous les navigateurs et plateformes, sans vous inquiéter de la possibilité des utilisateurs de les lire ou non.

Le chapitre suivant examine un certain nombre de fonctionnalités supplémentaires offertes par HTML5, notamment la géolocalisation et le stockage local.

Questions

1. Quelles sont les deux balises HTML nécessaires pour insérer de l'audio et de la vidéo dans un document en HTML5 ?
2. Quels sont les deux codecs audio nécessaires pour garantir une diffusion optimale sur toutes les plateformes ?
3. Quelles sont les méthodes disponibles pour lire et mettre en pause la lecture d'un contenu multimédia ?
4. Comment faire pour prendre en charge la lecture de multimédia dans des navigateurs non compatibles avec HTML5 ?
5. Quels sont les deux codecs vidéo nécessaires pour garantir une diffusion optimale sur toutes les plateformes ?

Retrouvez les réponses du chapitre 24 dans l'annexe A.

CHAPITRE 25

Autres fonctionnalités de HTML5

Ce dernier chapitre sur HTML5 explique comment exploiter la géolocalisation, le stockage local et les traitements web ; montre comment faire s'exécuter des applications web hors ligne ; et démontre l'utilisation du glisser-déposer dans un navigateur.

À proprement parler, la plupart de ces fonctionnalités (comme l'essentiel de HTML5) ne forment pas d'extensions du HTML, car vous y accédez à l'aide de JavaScript et non par des balises HTML. Ce sont simplement des technologies reprises par les développeurs de navigateurs et elles ont été rassemblées sous l'égide du vocable HTML5.

Ceci signifie cependant que vous devez avoir assimilé complètement le tutoriel sur JavaScript de ce livre pour les exploiter correctement. Ceci étant dit, dès que vous les maitriserez, vous vous demanderez comment vous avez pu vous passer de toutes ces puissantes possibilités.

Géolocalisation et le service GPS

Le service GPS (*Global Positioning Satellite*) est constitué d'une multitude de satellites géostationnaires en orbite autour de la terre, dont les positions sont connues avec une grande précision. Lorsqu'un appareil GPS se synchronise sur quelques-uns parmi eux, le temps nécessaire pour que leurs signaux parviennent de ces satellites à l'appareil récepteur permet à celui-ci de connaitre avec une assez grande précision où il se trouve, parce que, comme la vitesse de la lumière (et des ondes radios) est supposée constante, le temps nécessaire pour que le signal d'un satellite parvienne à l'appareil GPS indique la distance entre eux.

Le relevé des différents délais nécessaires aux signaux pour arriver depuis différents satellites, dont les positions orbitales sont connues avec précision, permet par une simple triangulation de calculer la position relative de l'appareil par rapport aux satellites, avec une précision de quelques mètres ou moins.

Nombre d'appareils mobiles, tels que des téléphones et des tablettes, contiennent une puce GPS et fournissent ces informations. Certains n'en possèdent pas, tandis que d'autres l'ont mais elle est désactivée, ou encore l'ont activée mais comme ils sont à l'intérieur d'un bâtiment imperméable aux ondes, ne peuvent accéder aux signaux satellites GPS et ne peuvent dont recevoir aucun signal. Dans ces cas-là, d'autres techniques permettent de déterminer plus ou moins leur emplacement.

Autres méthodes de localisation

Lorsqu'un appareil possède des fonctions de téléphonie mais sans GPS, il demeure possible d'estimer sa position géographique à partir du délai de réception des signaux émis par les antennes relais de télécommunications avec lesquelles il communique, alors que les emplacements des antennes sont parfaitement connus. Si ces antennes sont peu nombreuses, la précision peut s'avérer du même ordre que celle du GPS. En revanche, s'il n'y a qu'une seule antenne relais à proximité, alors la puissance du signal permet d'estimer un rayon approximatif autour de l'antenne et le cercle de couverture qu'elle crée représente la zone dans laquelle vous devez vous trouver. Cela vous situe n'importe où dans un rayon de deux à quatre kilomètres de votre emplacement réel, dans une bande géographique circulaire de quelques dizaines de mètres.

Si cela ne suffit pas, il se peut que vous accédiez à des points d'accès Wifi (*hotspots*) à portée de votre appareil, dont les emplacements sont connus. Or, comme tous ces points d'accès sans fil possèdent une adresse MAC (*Media access control*) unique, il est possible de déduire une assez bonne estimation de la position où vous vous trouvez, disons dans une rue ou deux. Ce sont là des informations typiques de ce que les véhicules de Google (les fameux *Google cars*) relèvent.

Et si cela ne suffit toujours pas, l'adresse IP (*Internet protocol*) utilisée par votre appareil pour accéder à l'internet peut remonter et servir d'indicateur approximatif de votre emplacement géographique. Cependant, cette information n'indique que l'emplacement d'un commutateur principal, propriété de votre fournisseur d'accès internet, qui peut se trouver à des dizaines, voire des centaines de kilomètres. Mais au moins, votre adresse IP permet généralement de déterminer votre pays, votre région, voire la ville où vous vous trouvez.

Les adresses IP sont généralement exploitées par les fournisseurs de contenu (multimédia) pour restreindre l'accès à leur contenu au cadre strict du pays d'émission (c'est le cas notamment de certaines télévisions nationales qui interdisent l'accès à certaines de leurs diffusions numériques en dehors de leurs frontières, pour respecter les accords internationaux en termes de diffusion de contenu multimédia). Il est cependant possible de paramétrer un serveur mandataire (proxy server) qui utilise une adresse IP relais dans le territoire-même du diffuseur pour récupérer et retransmettre le contenu outre la zone de restriction vers un navigateur étranger. Les serveurs mandataires servent aussi souvent à déguiser l'adresse IP réelle d'un utilisateur ou à contourner les restrictions de censure, et sont parfois partagés parmi de nombreux utilisateurs d'un point d'accès Wifi public (par exemple). Donc, lorsque vous essayez de localiser quelqu'un à partir de son adresse IP, vous n'avez aucune garantie absolue que l'emplacement identifié est authentique, ni que le pays d'origine est exact. Par conséquent, considérez cette information comme une estimation, sans plus.

Géolocalisation et HTML5

Le chapitre 22 présentait brièvement la géolocalisation HTML5. Examinons plus en détail le concept, en commençant par l'exemple proposé alors, repris dans l'exemple 25-1.

Exemple 25-1. Affichage d'une carte de votre emplacement courant

```
<!DOCTYPE html>
<html>
  <head>
    <title>Exemple de géolocalisation</title>
    <script src='OSC.js'></script>
    <script src="https://maps.googleapis.com/maps/api/js?sensor=false"></script>
  </head>
  <body>
    <div id='status'></div>
    <div id='map'></div>

    <script>
      if (typeof navigator.geolocation == 'undefined')
         alert("Géolocalisation non prise en charge.")
      else
        navigator.geolocation.getCurrentPosition(autorise, refuse)

      function autorise(position)
      {
        O('status').innerHTML = 'Permission accordée'
        S('map').border       = '1px solid black'
        S('map').width        = '640px'
        S('map').height       = '320px'

        var lat   = position.coords.latitude
        var long  = position.coords.longitude
        var gmap  = O('map')
        var gopts =
        {
          center: new google.maps.LatLng(lat, long),
          zoom: 9, mapTypeId: google.maps.MapTypeId.ROADMAP
        }
        var map = new google.maps.Map(gmap, gopts)
      }

      function refuse(error)
      {
        var message
        switch(error.code)
        {
          case 1: message = 'Permission refusée' ; break;
          case 2: message = 'Position non disponible' ; break;
          case 3: message = 'Dépassement de délai' ; break;
          case 4: message = 'Erreur inconnue' ; break;
        }
```

```
        O('status').innerHTML = message
      }
    </script>
  </body>
</html>
```

Examinons ce code et voyons comment il fonctionne. La section `<head>` affiche le titre, charge le fichier *OSC.js* contenant les fonctions `O`, `S` et `C` déjà bien connues, qui facilitent l'accès aux éléments HTML à partir de JavaScript, puis extrait le code JavaScript du service Google Maps, que nous retrouverons plus loin dans le programme.

Ensuite, nous créons dans le corps deux sections `<div>` : l'une pour afficher le statut de connexion et l'autre pour la carte :

```
<div id='status'></div>
<div id='map'></div>
```

La suite du document contient du JavaScript, d'abord pour interroger la propriété `navigator.geolocalisation`. Si la valeur renvoyée est `undefined`, alors le navigateur ne prend pas en charge la géolocalisation et nous affichons une fenêtre d'alerte.

Sinon, nous appelons la méthode `getCurrentPosition` de la propriété, et nous lui passons les noms de deux fonctions : `autorise` et `refuse` (comprenez : autorisé et refusé). Rappelez-vous qu'en passant des noms de fonctions, nous transmettons en réalité le code complet de ces fonctions et non le résultat de leur appel, ce qui serait le cas si nous avions placé des parenthèses, comme `autorise()` :

```
navigator.geolocation.getCurrentPosition(autorise, refuse)
```

Ces fonctions figurent plus loin dans le script et gèrent les deux possibilités d'autorisation à accéder aux données de géolocalisation : `autorise` ou `refuse`. La fonction `autorise` apparait en premier et n'est exécutée que si l'accès aux données est possible.

Dans cette fonction, nous réglons la propriété `innerHTML` de l'élément `div` d'identifiant `status` avec la chaine `Permission accordée` pour signaler la réussite durant le délai nécessaire pour obtenir la carte. Nous en profitons pour appliquer des styles à la section `div` de la carte `map` afin de lui donner une bordure et des dimensions déterminées :

```
O('status').innerHTML = 'Permission accordée'
S('map').border       = '1px solid black'
S('map').width        = '640px'
S('map').height       = '320px'
```

Ensuite, nous récupérons dans les variables `lat` et `long` les valeurs renvoyées par les routines (sous-programmes) de géolocalisation du navigateur, puis nous créons l'objet `gmap` pour accéder à l'élément `div map` :

```
var lat  = position.coords.latitude
var long = position.coords.longitude
var gmap = O('map')
```

Ceci fait, nous remplissons l'objet `gopts` avec les valeurs de `lat` et `long`, nous précisons le niveau de zoom (à 9 dans ce cas-ci) et nous sélectionnons le type de carte `ROADMAP` (carte routière) :


```
var gopts =
{
  center: new google.maps.LatLng(lat, long),
  zoom: 9, mapTypeId: google.maps.MapTypeId.ROADMAP
}
```

Enfin, dans cette fonction, nous créons un nouvel objet `map` et passons à la méthode `Map` de l'objet `google.maps` (le code, rappelez-vous, qui figure plus haut, juste après l'insertion du fichier OSC.js) les références à gmap et gopts.

```
var map = new google.maps.Map(gmap, gopts)
```

Si l'accès est autorisé à la géolocalisation de l'utilisateur, le résultat est illustré à la figure 25-1.

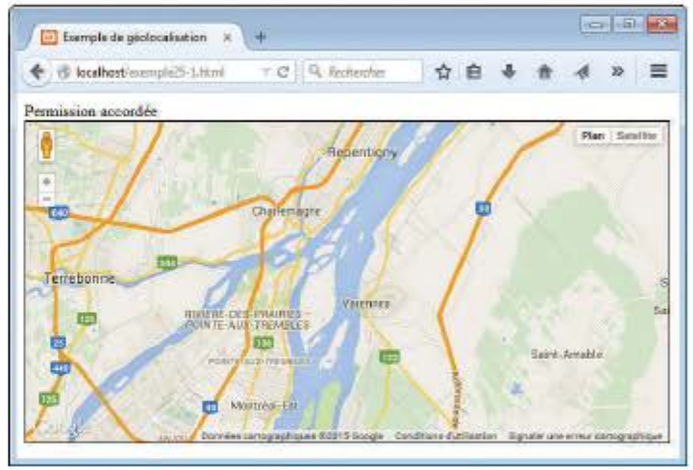


Figure 25-1. Affichage d'une carte de géolocalisation de l'utilisateur

Si la permission est refusée ou s'il y a un problème, la seule chose affichée est un message d'erreur, comme l'indique la propriété `innerHTML` de la `<div> status`, réglée par la fonction `refuse`, selon le problème détecté :

```
switch(error.code)
{
  case 1: message = 'Permission refusée' ; break;
  case 2: message = 'Position non disponible' ; break;
  case 3: message = 'Dépassement de délai' ; break;
  case 4: message = 'Erreur inconnue' ; break;
}

O('status').innerHTML = message
```

La carte de Google demeure totalement interactive et l'utilisateur peut y zoomer à volonté, et peut même la faire basculer en mode d'imagerie satellite.

Vous avez la possibilité de régler le niveau de zoom ou de type d'imagerie en fournissant des valeurs différentes à l'objet `gopts` (options Google). Par exemple, la valeur `1` pour `zoom` s'éloigne au maximum, tandis que la valeur `20` se rapproche au maximum. La valeur `SATELLITE` de la propriété `google.maps.MapTypeId` bascule en imagerie satellite et `HYBRID` combine les deux types de données.

Le réglage `sensor=false` de la fin de l'URL à partir de laquelle nous chargeons le script de Google (proche du début du document) dépend de la disponibilité d'un capteur GPS sur l'appareil de l'utilisateur : réglez-le à `true` si c'est le cas, sinon à `false`. Pour afficher une carte Google pour une géolocalisation déterminée, indépendante de l'emplacement de l'utilisateur, reprenez le code de la fonction `autorise` et remplacez les valeurs de `lat` et `long` (ainsi que les autres) par celles de votre choix. Si, par ailleurs, vous préférez exploiter les données des cartes Bing au lieu de celles de Google, sachez qu'elles sont devenues payantes, mais vous pouvez consulter les URL suivantes pour de plus amples informations : *http://www.microsoft.com/maps/choose-your-bing-maps-API.aspx* et *https://msdn.microsoft.com/fr-fr/library/ff428642.aspx*.

Stockage local

Les cookies constituent un élément essentiel de l'internet moderne, car ils permettent aux sites web d'enregistrer de petits extraits d'information sur la machine de l'utilisateur à des fins de suivi. Le concept n'est pas aussi inquiétant qu'il puisse paraître, car l'essentiel du suivi consiste à permettre aux visiteurs d'enregistrer des noms d'utilisateur et des mots de passe, de conserver leur connexion à des réseaux sociaux tels que Twitter ou Facebook et autres.

Les cookies enregistrent également localement vos préférences d'accès et de navigation sur un site web (au lieu d'enregistrer ces informations sur le serveur du site), ainsi que le contenu d'un panier d'achats à mesure que vous préparez une commande dans un site de commerce électronique.

Il est aussi vrai qu'ils peuvent servir à des fins plus agressives, comme pister les sites web que vous fréquentez pour dresser un portrait de vos intérêts et tenter de cibler plus efficacement les publicités qui vous sont adressées. C'est la raison qui explique que l'Union Européenne impose à tous les sites web au sein de ses frontières, d'alerter les visiteurs sur ce fait et de leur permettre de désactiver les cookies, s'ils le souhaitent.

Pourtant, en tant que développeur web, pensez à l'utilité de conserver des données sur les appareils des utilisateurs, en particulier si vous ne disposez que d'un petit budget pour vos serveurs et leur espace disque. Par exemple, vous pourriez créer des applications web intégrées au navigateur et des services pour l'édition de documents de traitement de texte, de feuilles de calcul et d'images, tout en enregistrant les données en dehors du serveur, sur les ordinateurs des utilisateurs, pour alléger autant que possible le budget de location du serveur.

Selon le point de vue de l'utilisateur, comprenez que la vitesse de chargement d'un document en local est bien plus grande que par l'entremise du web, surtout sur des connexions lentes. De plus, l'absence de copies de documents personnels sur le serveur est un gage de plus grande sécurité. Bien entendu, vous ne pouvez jamais garantir une sécurité absolue pour un site ou une application web, pas plus que vous ne pouvez traiter des documents extrêmement sensibles à l'aide de logiciels (ni de matériels) en ligne. Par contre, pour des documents au caractère privé relatif comme des photos de famille, vous gagnez en confort à utiliser une application web qui enregistre localement les documents plutôt que sur un serveur externe.

Tirer parti du stockage local

Le plus gros souci que présentent les cookies pour du stockage local réside dans le maximum de 4 Ko de données par cookie. Les cookies doivent également transiter en lecture et en écriture à chaque chargement de page. Enfin, à moins que vous n'utilisiez le chiffrement *SSL (Secure sockets layer)*, un cookie transite à chaque échange.

En HTML5, vous disposez d'un espace de stockage local beaucoup plus vaste (généralement entre 5 et 10 Mo par domaine, selon le navigateur) qui demeure en place d'un chargement d'une page au suivant et entre les visites du site web (et même après arrêt et redémarrage de l'ordinateur de l'utilisateur). De plus, les données de stockage local ne sont pas transmises au serveur lors du chargement de chaque page.

La gestion des données de stockage local opère sur des paires clé-valeur. La clé est le nom attribué pour faire référence aux données et la valeur peur contenir n'importe quel type de donnée, même si elle est stockée réellement sous forme de chaine. Toutes les données sont uniques pour le domaine courant et, pour des raisons de sécurité, l'espace de stockage créé par un site web est conservé distinctement de ceux des autres domaines, sans possibilité d'accéder à l'espace courant à partir d'un autre domaine que celui qui a stocké ces données.

Objet localStorage

Vous accédez à l'espace de stockage local à l'aide de l'objet `localStorage`. Pour tester la disponibilité de cet objet, interrogez son type pour vérifier qu'il a déjà été défini, comme suit :

```
if (typeof localStorage == 'undefined')
{
  // Stockage local non disponible, l'indiquer à l'utilisateur et quitter
  // ou proposer d'enregistrer les données sur le serveur à la place?
}
```

La manière dont vous gérez l'absence de disponibilité du stockage local dépend de la façon dont vous comptez l'utiliser, donc le code que vous placez dans l'instruction `if` dépend de vous.

Dès que vous avez la certitude que le stockage local est accessible, vous pouvez en tirer profit à l'aide des méthodes `setItem` et `getItem` de l'objet `localStorage`, comme suit :

```
localStorage.setItem('nomutil', 'ceastwood')
localStorage.setItem('motpasse', 'cestmonjour')
```

Pour retrouver ces données par la suite, passez les clés à la méthode `getItem`, comme ceci :

```
nomutil  = localStorage.getItem('nomutil')
motpasse = localStorage.getItem('motpasse')
```

Au contraire des cookies, dont l'enregistrement et la lecture ont lieu avant l'envoi des entêtes de document par le serveur, vous pouvez appeler ces méthodes à n'importe quel moment. Les valeurs enregistrées demeurent dans le stockage local tant qu'elles ne sont pas supprimées, comme ceci :

```
localStorage.removeItem('nomutil')
localStorage.removeItem('motpasse')
```

En guise d'alternative, vous pouvez vider la totalité de l'espace de stockage local correspondant au domaine courant, à l'aide de la méthode `clear`, comme cela :

```
localStorage.clear()
```

L'exemple 25-2 rassemble les exemples précédents en un seul document qui affiche les valeurs courantes des deux clés dans une boite d'alerte, avec les valeurs initiales réglées à `null`. Il enregistre ensuite les clés et les valeurs dans le stockage local, les relit et les affiche mais cette fois avec des valeurs attribuées. Enfin, il supprime les clés et tente de relire ces valeurs, alors que celles-ci sont de nouveau nulles. La figure 25-2 montre la deuxième boite d'alerte parmi les trois successives.

Exemple 25-2. Lecture, écriture et suppression de données dans le stockage local

```
if (typeof localStorage == 'undefined')
{
  alert("Stockage local inaccessible")
}
else
{
  nomutil = localStorage.getIten('nomutil')
  motpasse = localStorage.getIten('motpasse')
  alert("Les valeurs courantes de 'nomutil' et 'motpasse' sont\n\n" +
    nomutil + " / " + motpasse +
    "\n\nCliquez sur OK pour attribuer des valeurs")

  localStorage.setItem('nomutil', 'ceastwood')
  localStorage.setItem('motpasse', 'cestmonjour')
  nomutil = localStorage.getIten('nomutil')
  motpasse = localStorage.getIten('motpasse')
  alert("Les valeurs courantes de 'nomutil' et 'motpasse' sont\n\n" +
    nomutil + " / " + motpasse +
    "\n\nCliquez sur OK pour effacer les valeurs")

  localStorage.removeItem('nomutil')
  localStorage.removeItem('motpasse')
  nomutil = localStorage.getIten('nomutil')
  motpasse = localStorage.getIten('motpasse')
  alert("Les valeurs courantes de 'nomutil' et 'motpasse' sont\n\n" +
    nomutil + " / " + motpasse)
}
```

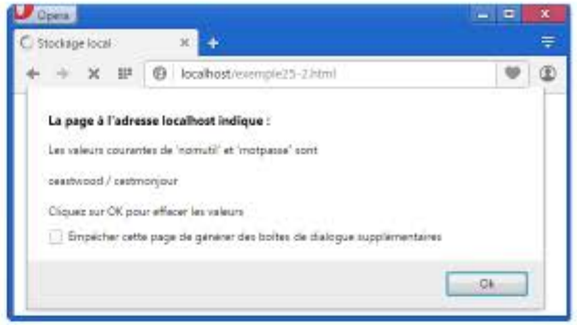



Figure 25-2. Lecture de deux clés et de leurs valeurs dans le stockage local

Rien ne vous oblige à vous restreindre à des noms d'utilisateurs et des mots de passe car vous pouvez gérer pratiquement n'importe quel type de données et autant de paires clé-valeur que vous le souhaitez, dans la limite de l'espace de stockage disponible pour votre domaine.

Traitement web : tâches de fond et multitâche

Selon le principe, les *traitements web* (*web workers*) sont des sections de code JavaScript qui s'exécutent à l'arrière-plan, sans qu'il soit nécessaire de paramétrer ni de surveiller des interruptions. Au lieu de cela, chaque fois qu'un traitement web doit signaler quelque chose, le processus en arrière-plan communique avec le script principal par l'entremise d'évènements.

Ceci implique que l'interpréteur JavaScript doit décider comment allouer des tranches de temps de la manière la plus efficace et que votre code doit s'enquérir des communications avec la tâche en coulisses lorsqu'il y a lieu d'échanger des informations.

L'exemple 25-3 montre comment définir un traitement web pour calculer une tâche répétitive à l'arrière-plan, soit dans ce cas-ci, calculer des nombres premiers.

Exemple 25-3. Définition d'un traitement web et des communications avec lui

```
<!DOCTYPE html>
<html> <!-- webworkers.html -->
  <head>
    <title>Traitement web</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    Plus grand nonbre premier courant :
```

```
    <span id='resultat'>0</span>

    <script>
      if (!!window.Worker)
      {
        var worker = new Worker('worker.js')

        worker.onmessage = function (event)
        {
          O('resultat').innerHTML = event.data;
        }
      }
      else
      {
        alert("Traitements web non pris en charge")
      }
    </script>
  </body>
</html>
```

Cet exemple crée un élément `<span>` d'identifiant `resultat` qui recevra les résultats produits par le traitement web. Ensuite, la section `<script>` teste `window.Worker` à l'aide d'une paire d'opérateurs `!` (`not`), qui a pour effet de renvoyer la valeur booléenne `true` si la méthode `Worker` existe ou `false` dans le cas contraire. Si le résultat vaut `false`, la section `else` affiche un message indiquant que les traitements web ne sont pas pris en charge.

Si le test réussit, nous créons un objet nommé `worker` et de classe `Worker`, à laquelle nous passons le nom de fichier *worker.js* (voir plus loin). Nous associons ensuite l'événement `onmessage` de l'objet `worker` à une fonction anonyme qui dépose tout message transmis par *worker.js* à l'événement dans la propriété `innerHTML` de l'élément `<span>` défini plus haut.

Nous enregistrons le traitement web proprement dit dans le fichier *worker.js* de l'exemple 25-4.

Exemple 25-4. Le traitement web worker.js

```
var n = 1

recherche: while (true)
{
  n += 1

  for (var i = 2; i <= Math.sqrt(n); i += 1)
  {
    if (n % i == 0) continue recherche
  }

  postMessage(n)
}
```


Ce fichier affecte la valeur `1` à la variable `n`. Il boucle ensuite de façon permanente et incrémente `n`, pour rechercher ensuite de manière systématique toutes les valeurs de `1` à la racine carrée de `n` et vérifier si elles produisent une division entière, sans reste. Si un tel facteur est présent, la commande `continue` arrête immédiatement la recherche par la manière systématique, puisque le nombre n'est pas premier. La commande `continue` redémarre une nouvelle `recherche` avec la valeur de `n` suivante.

Lorsque tous les facteurs ont été testés et qu'aucun résultat entier sans reste décimal n'a été trouvé, alors `n` est un nombre premier. Dans ce cas, sa valeur est passée à `postMessage`, qui renvoie un message à l'évènement `onmessage` de l'objet qui a mis ce traitement web en place.

Le résultat prend l'aspect suivant :

```
Plus grand nombre premier courant : 30477191
```

Pour interrompre l'exécution du traitement web, effectuez un appel à la méthode `terminate` de l'objet `worker`, comme suit :

```
worker.terminate()
```

Pour interrompre l'exécution de cet exemple précis, entrez l'URL suivante dans la barre d'adresse du navigateur :

```
javascript:worker.terminate()
```

Remarquez qu'à cause de la façon dont Chrome gère la sécurité, vous ne pouvez utiliser des traitements web qui touchent au système de fichiers, mais seulement à partir d'un serveur web (ou l'exécution de fichiers sur `localhost` dans le cadre d'un serveur de développement tel que le serveur XAMPP évoqué au chapitre 2).

Applications web hors ligne

Si vous fournissez à un navigateur les informations adéquates, vous pouvez lui demander de télécharger tous les composants d'une page web, puis la charger et l'exécuter hors ligne. Le principal fichier dont vous avez besoin est un *fichier de manifeste* (*manifest*) d'extension *.appcache*. Pour illustrer le principe avec une application web simple, l'exemple 25-5 crée le fichier *horloge.appcache* d'une horloge.

Exemple 25-5. Le fichier de manifeste horloge.appcache

```
CACHE MANIFEST
horloge.html
OSC.js
horloge.css
horloge.js
```

La première ligne du fichier le déclare en tant que fichier de manifeste. Les lignes suivantes énumèrent les fichiers que le navigateur doit télécharger et stocker, à

commencer par l'exemple 25-6, le fichier *horloge.html*, suivi du fichier *OSC.js*, identique à celui déjà utilisé depuis de nombreux chapitres précédents.

Exemple 25-6. Le fichier horloge.html

```
<!DOCTYPE html>
<html manifest='horloge.appcache'>
  <head>
    <title>Application web hors ligne</title>
    <script src='OSC.js'></script>
    <script src='horloge.js'></script>
    <link rel='stylesheet' href='horloge.css'>
  </head>
  <body>
    Nous sommes : <output id='horloge'></output>
  </body>
</html>
```

Ce code déclare qu'il possède un fichier de manifeste au sein-même de la balise `<html>` :

```
<html manifest='horloge.appcache'>
```

Pour être certain de pouvoir prendre en charge les applications web, vous devez peut-être ajouter le type MIME `text/cache-manifest` pour l'extension *.appcache* dans votre serveur, afin que celui-ci renvoie le fichier de manifeste avec le bon type. Il existe un petit raccourci pour éviter les complications, qui consiste à créer un fichier dénommé *.htaccess* dans le même dossier que celui des autres fichiers à rendre accessibles hors ligne, avec le contenu suivant :

```
AddType text/cache-manifest .appcache
```

Ensuite, la section de JavaScript importe les fichiers *OSC.js*, *horloge.js* et *horloge.css* pour les exploiter dans le document. L'exemple 25-7 reprend le code JavaScript du fichier *horloge.js*.

Exemple 25-7. Le fichier horloge.js

```
setInterval(function()
{
  O('horloge').innerHTML = new Date()
}, 1000)
```

Il s'agit là d'une fonction anonyme simple associée à un intervalle qui se répète toutes les secondes pour écrire la date et l'heure courantes dans la propriété `innerHTML` de l'élément `<output>` d'identifiant horloge.

Le dernier fichier s'appelle *horloge.css* (exemple 25-8) et applique du style gras à l'élément `<output>`.

Exemple 28-8. Le fichier horloge.css

```
output { font-weight:bold ; }
```

À condition que le fichier *horloge.appcache* les énumère tous, ces quatre fichiers (*horloge.html*, *OSC.js*, *horloge.js* et *horloge.css*) forment ensemble une application web capable de fonctionner hors ligne. Ils seront tous téléchargés et mis localement à la disposition de tout navigateur web en mesure de comprendre et de gérer des applications hors ligne. Lors de l'exécution, le résultat donne ce qui suit :

```
Nous sommes : Mon May 04 2015 11:34:26 GMT+0200
```

Pour de plus amples informations à propos de la norme des applications web hors ligne, consultez (en anglais) le site officiel, sur https://html.spec.whatwg.org/#offline.

Glisser-déposer

La prise en charge du glisser (*drag*) et déposer (*drop*) d'objets dans une page web se résume à définir les gestionnaires des évènements `ondragstart` (au début du glisser), `ondragover` (au passage par-dessus) et `ondrop` (au dépôt), comme dans l'exemple 25-9.

Exemple 25-9. Glisser-déposer d'objets

```
<!DOCTYPE HTML>
<html> <!-- draganddrop.html -->
  <head>
    <title>Glisser et déposer</title>
    <script src='OSC.js'></script>
    <style>
      #dest {
        background:lightblue ;
        border     :1px solid #444 ;
        width      :320px ;
        height     :100px ;
        padding    :10px ;
      }
    </style>
  </head>
  <body>
    <div id='dest' ondrop='deposer(event)' ondragover='autoriser(event)'></div><br>
    Glissez les images du dessous dans l'élément du dessus<br><br>

    <img id='source1' src='image1.png' draggable='true' ondragstart='glisser(event)'>
    <img id='source2' src='image2.png' draggable='true' ondragstart='glisser(event)'>
    <img id='source3' src='image3.png' draggable='true' ondragstart='glisser(event)'>
```

```
    <script>
      function autoriser(event)
      {
        event.preventDefault()
      }

      function glisser(event)
      {
        event.dataTransfer.setData('image/png', event.target.id)
      }

      function deposer(event)
      {
        event.preventDefault()
        var data=event.dataTransfer.getData('image/png')
        event.target.appendChild(O(data))
      }
    </script>
  </body>
</html>
```

Le code de cet exemple commence par déclarer le HTML, définir le titre et charger le fichier *OSC.js*. Il donne ensuite un style à l'élément `div` d'identifiant `dest`, constitué d'une couleur de fond, d'une bordure, de dimensions et de largeur de marge intérieure.

La section `<body>` apparait ensuite, qui crée l'élément `<div>` avec ses évènements `ondrop` et `ondragover`, pour leur associer les fonctions respectives `deposer` et `autoriser`, puis crée un peu de texte explicatif et trois images. Chaque image voit sa propriété `draggable` réglée à `true`, tandis que la fonction `glisser` vient s'associer à son évènement `ondragstart`.

Dans la section `<script>`, la fonction `autoriser` de gestionnaire d'évènement neutralise l'action prédéfinie (qui empêche le glisser), puis la fonction `glisser` de gestion d'évènement appelle la méthode `setData` de l'objet `dataTransfer` de l'évènement, lui passe le type MIME `image/png` et l'identifiant de cible `target.id` de l'évènement (qui correspond à l'objet sujet du glisser). L'objet `dataTransfer` détient les données relatives à l'objet en cours pendant l'opération de glisser-déposer.

Enfin, la fonction `deposer` de gestion d'évènement intercepte aussi son action prédéfinie (qui autorise le dépôt par défaut), lit dans `data` le contenu en cours du glisser à partir de l'objet `dataTransfer` et impose le type MIME de l'objet lu. Les données déposées sont ajoutées à la cible, soit la `div dest`, à l'aide de la méthode `appendChild`.

Lorsque vous testez cet exemple, vous pouvez glisser et déposer les images dans l'élément `div`, où ils restent, comme illustré à la figure 25-3.

Figure 25-3. Deux images glissées et déposées

Vous disposez encore d'autres évènements, comme `ondragenter`, lorsqu'une opération de glisser entre dans un élément, `ondragleave`, quand elle quitte un élément, et `ondragend`, lorsque s'achève l'opération de glisser-déposer. Vous pouvez exploiter ces évènements pour, par exemple, modifier la forme du pointeur durant ces opérations.

Messagerie interdocuments

La section sur les traitements web donnait déjà un aperçu de la messagerie sans entrer dans les détails, car ce n'était pas le sujet de la section. En outre, le message n'était publié qu'au même document. Pour des raisons de sécurité évidentes, la messagerie doit faire l'objet de précautions et mérite toute votre attention si vous la mettez en œuvre. Il vaut donc mieux comprendre parfaitement son fonctionnement si vous comptez l'exploiter.

Avant HTML5, les développeurs de navigateurs interdisaient la mise en place de scripts intersites. Mais, tout en bloquant les sites d'attaque potentielle, cela interdisait toute communication entre des pages légitimes. Il fallait donc assurer l'interaction entre des documents, de quelque ordre que ce soit, généralement par de l'Ajax et un serveur web tiers, ce qui représentait une maintenance lourde et compliquée.

Or, la messagerie web permet désormais à des scripts d'interagir au-delà de ces limites, sous condition de quelques restrictions de sécurité évidentes et raisonnables pour empêcher toute tentative de détournement malveillant. Cela fonctionne à l'aide de la méthode `postMessage`, qui autorise l'envoi de messages en texte brut d'un domaine à un autre.

Ceci exige que JavaScript puisse d'abord récupérer l'objet `Window` du document récepteur, pour être en mesure de publier (ou « poster ») des messages à toutes sortes d'autres fenêtres, cadres et *iframes* directement liés au document de l'expéditeur. L'évènement du message reçu possède les attributs suivants :

`data`
: Le message entrant.

`origin`
: L'origine du document expéditeur, avec son schéma, son nom d'hôte et son port.

`source`
: La fenêtre source du document expéditeur.

Le code nécessaire pour envoyer des messages se résume à une seule instruction, dans laquelle vous passez le message à envoyer et le domaine auquel il s'applique, comme dans l'exemple 25-10.

Exemple 25-10. Envoi de messages web vers un iframe

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Messagerie web (a)</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <iframe id='cadre' src='exemple25-11.html' width='360' height='75'></iframe>

    <script>
      compteur = 1

      setInterval(function()
      {
        O('cadre').contentWindow.postMessage('Message ' + compteur++, '*')
      }, 1000)
    </script>
  </body>
</html>
```

À nouveau, l'appel du fichier *OSC.js* permet d'accéder à la fonction `O`, puis nous définissions un `iframe` d'identifiant `cadre`, qui charge l'exemple 25-11. La section `<script>` qui suit initialise la variable compteur à `1` et définit un intervalle de minuterie qui se répète toutes les secondes et publie la chaine `'Message '` (à l'aide de la méthode `postMessage`), ainsi que la valeur courante de `compteur`, incrémentée ensuite. L'appel à `postMessage` est associé à la propriété `contentWindow` de l'objet `iframe`, et non à l'objet `iframe` lui-même. Ceci est important parce que la messagerie web exige que les publications soient déposées dans une fenêtre et non dans un objet faisant partie d'une fenêtre.

Exemple 25-11. Réception de messages d'un autre document

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Messagerie web (b)</title>
    <style>
      #output {
        font-family:"Courier New";
        white-space:pre;
      }
    </style>
    <script src='OSC.js'></script>
  </head>
  <body>
    <div id='output'>Les messages reçus s'affichent ici</div>

    <script>
      window.onmessage = function(event)
      {
        O('output').innerHTML =
          '<b>Origine :</b> ' + event.origin + '<br>' +
          '<b>Source :</b>  ' + event.source + '<br>' +
          '<b>Data :</b>    ' + event.data
      }
    </script>
  </body>
</html>
```

L'exemple impose un peu de style pour faciliter la lecture des résultats, puis crée un élément `div` d'identifiant `output`, qui recevra les messages reçus. La section `<script>` contient une seule fonction anonyme associée à l'évènement `onmessage` de la fenêtre. Cette fonction affiche ensuite les valeurs des propriétés `event.origin`, `event.source` et `event.data`, comme illustré à la figure 25-4.

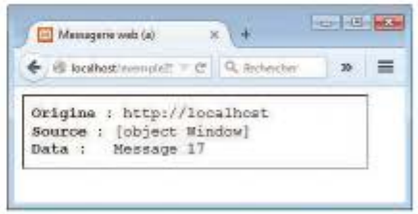


Figure 25-4. Le cadre iframe a reçu 17 messages jusqu'ici

La messagerie web fonctionne seulement entre des domaines donc il n'est pas possible de tester l'exemple à partir du système de fichiers local : vous devez passer par un serveur web. La figure 25-4 indique que l'origine se situe dans *http://localhost* parce que ces exemples s'exécutent sur le serveur de développement local. La source est dans l'objet `Window` et la valeur courante du message est `Message 17`.

À ce stade, l'exemple 25-10 n'est absolument pas sécuritaire car la valeur de domaine passée à `postMessage` est le caractère générique `*` :

```
O('cadre').contentWindow.postMessage('Message ' + compteur++, '*')
```

Pour rediriger les messages uniquement à des documents provenant d'un domaine déterminé, modifiez ce paramètre. Dans ce cas présent, la valeur `http://localhost` garantirait que seuls les documents chargés par le serveur local peuvent envoyer ces messages :

```
O('cadre').contentWindow.postMessage('Message ' + compteur++, 'http://localhost')
```

De la même manière, dans la situation actuelle, le programme écouteur affiche tout message qu'il reçoit. Cela ne conduit pas à une sécurité optimale parce que des documents malveillants également présents dans le navigateur risquent de tenter d'envoyer des messages que le code d'un écouteur imprudent dans un autre document risque à son tour d'ouvrir. Par conséquent, si vous êtes certain du domaine de l'émetteur que vous attendez, vous pouvez restreindre les messages qu'interceptent vos écouteurs à l'aide d'une instruction `if`, comme suit :

```
window.onmessage = function(event)
{
  if (event.origin) == 'http://localhost')
  {
    O('output').innerHTML =
      '<b>Origine :</b>  ' + event.origin + '<br>' +
      '<b>Source :</b>   ' + event.source + '<br>' +
      '<b>Data :</b>     ' + event.data
  }
}
```

Si vous prenez la précaution de toujours utiliser le domaine adéquat pour le site avec lequel vous travaillez, vous augmenterez la sécurité de vos communications par messagerie web. Cependant demeurez conscient que, comme les messages sont transmis en clair, des risques de sécurité persistent dans certains navigateurs ou dans les modules d'extension du navigateur qui rendent ce genre de communication non sécuritaire. Une manière d'améliorer la sécurité consiste donc à créer votre propre système de dissimulation ou de chiffrement pour tous vos messages web et d'envisager également vos propres protocoles de communication à deux voies pour vérifier l'authenticité de chaque message.

Normalement, vous n'informez pas l'utilisateur des valeurs de l'`origin` et de la `source`, qui ne servent en principe qu'à des vérifications de sécurité. Ces exemples les exposent néanmoins pour vous permettre de vous livrer à des expériences avec la messagerie web et de voir ce qu'elles produisent. Tout comme les iframes, les documents dans des fenêtres contextuelles et autres onglets peuvent communiquer entre eux à l'aide de cette méthode.

Microdonnées

Les *microdonnées* (*microdata*) constituent un sous-ensemble de HTML conçu pour fournir des métadonnées à un document pour que celui-ci ait une signification vis-à-vis des logiciels, autant qu'il a une signification pour le lecteur du document. Les microdonnées rendent accessibles les nouveaux attributs de balise suivants : `itemscope`, `itemtype`, `itemid`, `itemref` et `itemprop`. Grâce à ceux-ci, vous pouvez définir clairement les propriétés d'un article, comme un livre, en fournissant des informations qu'un ordinateur peut comprendre, par exemple son auteur, son éditeur, son contenu et ainsi de suite.

Les microdonnées servent plus fréquemment de nos jours à renseigner les moteurs de recherche et les sites de réseaux sociaux. L'exemple 25-12 crée une courte biographie de George Washington, comme s'il était sur un site de réseau social, avec des microdonnées ajoutées aux différents éléments (en gras). La figure 25-5 montre le résultat, qui apparait de façon identique avec ou sans microdonnées, car elles ne sont pas visibles par l'utilisateur.

Exemple 25-12. Ajout de microdonnées en HTML

```
<!DOCTYPE html>
<html> <!-- microdonnees.html -->
  <head>
    <title>Microdonnées</title>
  </head>
  <body>
    <section itemscope itemtype='http://schema.org/Person'>
      <img itemprop='image' src='gw.jpg' alt='George Washington'
        style='margin-right:10px; float:left;'>
      <h2 itemprop='name'>George Washington</h2>
      <p>Je suis le premier <span itemprop='jobTitle'>président américain</span>.
      Mon site web est : <a itemprop='url'
        href='http://georgewashington.si.edu'>georgewashington.si.edu</a>.
      Voici mon adresse :</p>
      <address itemscope itemtype='http://schema.org/PostalAddress'
        itemprop='address'>
        <span itemprop='streetAddress'>1600 Pennsylvania Avenue</span>,<br>
        <span itemprop='addressLocality'>Washington</span>,<br>
        <span itemprop='addressRegion'>DC</span>,
        <span itemprop='postalCode'>20500</span>,<br>
        <span itemprop='addressCountry'>États-Unis d'Amérique</span>.
      </address>
    </section>
  </body>
</html>
```

Figure 25-5. Ce document contient des microdonnées, qui demeurent invisibles

Les navigateurs ne font rien de particulier des microdonnées, mais cela vaut néanmoins la peine de les connaitre. L'utilisation précise des microdonnées appropriées fournit de nombreuses informations aux moteurs de recherche, tels que Google et Bing, et aide à promouvoir les pages clairement annotées dans les classements, par rapport aux sites qui ne les mettent pas en œ uvre.

Dans une certaine mesure, les navigateurs trouvent cependant une utilité à ces informations et vous pouvez déterminer s'ils tiennent compte ou non des microdonnées en vérifiant si la méthode `getItems` existe, comme suit :

```
if (!!document.getItems)
{
  // Microdonnées prises en charge
}
else
{
  // Microdonnées non prises en charge
}
```

La paire d'opérateurs `not` et `!!` constitue un raccourci pour forcer le renvoi d'une valeur booléenne qui représente l'existence (ou l'absence) de la méthode `getItems`. Si elle existe, alors l'expression renvoie `true` et les microdonnées sont prises en charge. Dans le cas contraire, elle renvoie `false`.

Actuellement, seuls les navigateurs Mozilla Firefox et Opera prennent en charge l'accès aux microdonnées, mais les autres navigateurs devraient suivre rapidement le mouvement. Lorsque ce sera le cas, vous serez en mesure d'extraire les données de la manière suivante, où (après le chargement de la page) nous récupérons l'objet `data` à l'aide d'un appel à `getItems`, puis la valeur de la clé `'jobTitle'` (en guise d'exemple) à partir de l'objet `properties` de l'objet `data`, et de la propriété `textContent` du premier objet :

```
window.onload = function()
{
  if (!!document.getItems)
  {
    data = document.getItems('http://schema.org/Person')[0]
    alert(data.properties['jobTitle'][0].textContent)
  }
}
```

Les navigateurs qui prennent en charge cette fonctionnalité affichent l'équivalent de la figure 25-6, tandis que les autres ne déclenchent pas l'affichage de la fenêtre contextuelle.

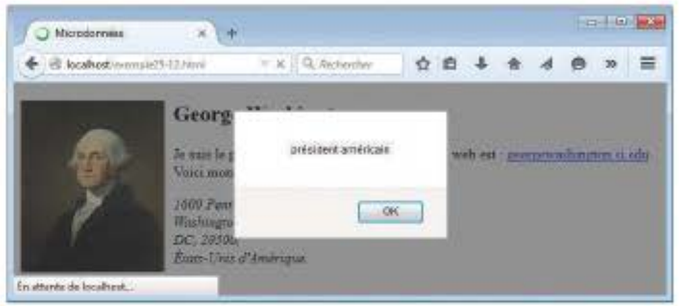


Figure 25-6. Affichage de la valeur de la clé de microdonnée du 'jobTitle'

Google a déclaré faire désormais usage des microdonnées lorsqu'il en trouve et que les microdonnées constituent également le format d'extrait préférentiel pour Google+, donc cela vaut la peine de commencer à en introduire dans votre HTML lorsque c'est possible. Pour une information complète sur la myriade de propriétés disponibles pour les microdonnées, consultez *http://schema.org*, qui constitue aussi la référence des schémas de microdonnées tels qu'on les déclare dans les propriétés `itemType`.

Autres balises en HTML5

Plusieurs autres nouvelles balises de HTML5 n'ont pas encore été implémentées dans de nombreux navigateurs, donc nous ne les détaillons pas ici, d'autant plus que leurs spécifications techniques risquent encore de changer. Par simple souci d'exhaustivité, ces balises sont : `<article>`, `<aside>`, `<details>`, `<figcaption>`, `<figure>`, `<footer>`, `<header>`, `<hgroup>`, `<keygen>`, `<mark>`, `<menuitem>`, `<meter>`, `<nav>`, `<output>`, `<progress>`, `<rp>`, `<rt>`, `<ruby>`, `<section>`, `<summary>`, `<time>` et `<wbr>`. Pour de plus amples informations sur celles-ci et d'autres balises HTML5, consultez *http://tinyurl.com/h5markup* (consultez les éléments qui portent l'icône *NEW*).

En résumé

Ceci conclut l'introduction à HTML5. Vous disposez à présent de nouvelles fonctionnalités puissantes qui permettent de développer des sites web encore plus dynamiques et percutants. Le chapitre final vous montre comment rassembler toutes ces différentes technologies exposées dans le livre pour créer un mini-site de réseau social.

Questions

1. Quelle méthode appelez-vous pour obtenir des données de géolocalisation d'un navigateur web ?
2. Comment déterminez-vous si un navigateur prend en charge le stockage local ?
3. Quelle méthode appelez-vous pour effacer la totalité des données du stockage local pour le domaine courant ?
4. Quel est le meilleur moyen pour un traitement web de communiquer avec un programme principal ?
5. Comment faire pour qu'un code informe un navigateur web de ce que le document peut s'exécuter hors ligne comme une application web locale ?
6. Pour prendre en charge les opérations de glisser-déposer, comment empêchez-vous l'action par défaut de bloquer le glisser-déposer pour ces évènements ?
7. Comment faites-vous pour améliorer la sécurité de la messagerie interdocuments ?
8. Quel est le but des microdonnées ?

Retrouvez les réponses du chapitre 25 dans l'annexe A.

CHAPITRE 26

Assembler le tout

Maintenant que vous avez atteint la fin de votre voyage dans l'apprentissage des tenants et aboutissants de la programmation web dynamique, avec tous les pourquoi, comment et où, ce chapitre vous propose de vous faire les dents sur un exemple réel. En pratique, il s'agit plutôt d'une suite d'exemples qui, rassemblés, forment un projet de plateforme de réseau social reprenant les principales fonctionnalités que l'on peut attendre d'un tel site.

Parmi les différents fichiers, vous trouverez des exemples de création de table MySQL, d'accès à une base de données, de feuilles de styles CSS, d'inclusion de fichiers, de contrôle de session, d'accès au DOM, d'appels Ajax, de gestion d'évènements et d'erreurs, de versement de fichier à distance, de manipulation d'image, de canevas HTML5 et de bien d'autres choses.

Chaque fichier exemple est complet et autonome, tout en collaborant avec les autres pour constituer un site de réseau social opérationnel, qui inclut même une feuille de styles dont vous pouvez modifier les détails pour changer complètement l'aspect et le comportement du projet. Petit et léger, le produit final est particulièrement adapté aux plateformes mobiles, telles que des téléphones intelligents (*smartphones*) et des tablettes, mais également aux ordinateurs de bureau classiques.

N'hésitez pas à prendre des extraits de code qui peuvent vous être utiles pour les étendre, les améliorer et les adapter. Peut-être souhaiterez-vous bâtir sur ces fichiers votre propre site de réseau social. Allez-y et faites-vous plaisir !

Concevoir un site de réseau social

Avant d'écrire la moindre ligne de code, je me suis assis et jeté sur papier un certain nombre d'idées que j'ai décidé d'inclure, parce que je les considère comme essentielles pour un tel site. Ce sont notamment les suivantes :

- un processus d'inscription ;
- un formulaire d'identification (connexion) ;

- un utilitaire de déconnexion ;
- un contrôle de session ;
- des profils d'utilisateurs avec des vignettes d'images ;
- un répertoire de membres ;
- l'ajout de membres en tant qu'amis ;
- une messagerie publique et privée parmi les membres ;
- une possibilité de donner du style au projet.

Dans la foulée, j'ai décidé d'appeler ce projet le *Nid de Robin*, mais vous avez tout le loisir de le renommer et il vous suffit pour cela de modifier une seule ligne de code dans *functions.php*.

Comme il s'agit ici d'un exemple réel, résolument ouvert vers la modification de code par toutes sortes de contributeurs, je respecte ici la convention déjà martelée à plusieurs reprises au cours des chapitres précédents : le code est en anglais mais l'interface, c'est-à-dire tout ce qui apparait à l'utilisateur, est en français. Ainsi, si des programmeurs de tous horizons doivent apporter leur contribution à vos codes sources, ils n'auront aucune difficulté à les relire. Quelques commentaires en français viennent toutefois compléter le code si nécessaire.

Pour garantir l'affichage des caractères accentués, il est préférable d'enregistrer tous les fichiers d'exemples au format *utf8*.

Sur le site web

Vous trouverez tous les exemples de ce chapitre sur le site d'accompagnement du livre. Cliquez sur le lien de téléchargement pour obtenir les fichiers. Le fichier compressé ZIP que vous obtenez contient un dossier nommé *nidderobin*, dans lequel tous les exemples suivants sont repris avec les noms de fichiers corrects, nécessaires pour cette application. Copiez simplement ce dossier et son contenu dans le dossier racine de votre serveur web de développement pour tester les exemples.

Fonctions annexes : functions.php

À partir d'ici, nous plongeons directement dans le projet, à commencer par l'exemple 26-1, nommé *functions.php*, qui regroupe les fonctions principales et que nous inclurons selon les nécessités. Ce fichier contient un peu plus que des fonctions car j'y ai ajouté les détails de connexion à la base de données au lieu de les placer dans un fichier distinct. Donc les quelques premières lignes définissent l'hôte, le nom de base de données, le nom d'utilisateur et le mot de passe qui permettent d'accéder à la base de données du projet.

Peu importe le nom que vous donnez à la base de données, pourvu qu'elle existe déjà (reportez-vous au chapitre 8 pour la façon de créer une base de données). Assurez-vous

également d'affecter un utilisateur et un mot de passe MySQL aux variables `$dbuser` et `$dbpass`. La dernière instruction d'initialisation définit le nom du site de réseau social, sous la forme de la chaine `Nid de Robin` affectée à la variable `$appname`. Si vous voulez modifier le nom, changez le contenu de cette chaine. Avec des valeurs correctes, les deux instructions suivantes ouvrent une connexion à MySQL et sélectionnent la base de données adéquate. Le bloc de code suivant exécute une requête MySQL avec l'instruction `SET NAMES utf8`, pour qu'un nom d'utilisateur et des messages avec des caractères accentués s'affichent dans les pages du site.

Les fonctions

Le projet utilise cinq fonctions PHP principales :

`createTable`
: Vérifie si une table existe et, sinon, la crée.

`queryMysql`
: Envoie une requête (*query*) à MySQL et produit un message d'erreur en cas d'échec.

`destroySession`
: Détruit une session PHP et efface ses données pour déconnecter un utilisateur.

`sanitizeString`
: Supprime tout code malintentionné éventuel et toute balise interdite d'une entrée de l'utilisateur.

`showProfile`
: Affiche une image de l'utilisateur et le message « À propos de moi » s'il en possède un.

Toutes ces fonctions devraient vous être familières à ce stade-ci, sauf peut-être `showProfile`, qui recherche une image nommée *utilisateur.jpg* (où utilisateur est remplacé par le nom de l'utilisateur courant) et, si elle la trouve, l'affiche. Elle affiche aussi le texte « À propos de moi » si l'utilisateur en a enregistré un.

Je me suis assuré de mettre en place une gestion d'erreur pour toutes les fonctions qui la nécessitent, pour qu'elles soient en mesure de capturer toute erreur typographique ou autre que l'utilisateur risque d'induire, et de générer des messages d'erreur adéquats. Néanmoins, si vous portez ce site sur un serveur de production, n'hésitez pas à fournir vos propres routines de gestion d'erreur pour apporter une convivialité maximale au code.

Entrez le code de l'exemple 26-1 et enregistrez le fichier sous le nom *functions.php*, ou téléchargez ce fichier de l'archive ZIP à partir du site d'accompagnement, et vous serez prêt à passer à la section suivante.

Exemple 26-1. Fonctions annexes : functions.php

```
<?php
  $dbhost  = 'localhost';    // Hôte probable
  $dbname  = 'nidderobin';   // Modifiez ces...
  $dbuser  = 'nidderobin';   // ...variables en fonction
```

```
$dbpass  = 'ndrpassword';  // ...de votre installation
$appname = "Nid de Robin"; // ...et de vos préférences

$connection = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
if ($connection->connect_error) die($connection->connect_error);

function createTable($name, $query)
{ // Créer une table
  // Imposer le jeu de caractères utf8
  queryMysql("CREATE TABLE IF NOT EXISTS $name($query) CHARSET utf8");
  echo "Table '$name' créée ou existe déjà.<br>";
}

function queryMysql($query)
{ // Lancer une requête SQL
  global $connection;
  $result = $connection->query($query);
  if (!$result) die($connection->error);
  return $result;
}

function destroySession()
{ // Supprimer la session
  $_SESSION=array();

  if (session_id() != "" || isset($_COOKIE[session_name()]))
    setcookie(session_name(), '', time()-2592000, '/');

  session_destroy();
}

function sanitizeString($var)
{ // Aseptiser une chaine
  global $connection;
  $var = strip_tags($var);
  $var = htmlentities($var);
  $var = stripslashes($var);
  return $connection->real_escape_string($var);
}

function showProfile($user)
{ // Afficher le profil
  if (file_exists("$user.jpg"))
    echo "<img src='$user.jpg' style='float:left;'>";

  $result = queryMysql("SELECT * FROM profiles WHERE user='$user'");

  if ($result->num_rows)
  {
    $row = $result->fetch_array(MYSQLI_ASSOC);
    echo stripslashes($row['text']) . "<br style='clear:left;'><br>";
  }
}
?>
```


Pour faire référence à la base de données MySQL avec la nouvelle extension `mysqli`, vous devez appliquer le mot-clé `global` aux fonctions `queryMysql` et `sanitizeString` pour leur permettre d'accéder à la valeur contenue dans `$connection`.

En-tête : header.php

Pour des raisons d'uniformité, chaque page du projet doit accéder au même ensemble de fonctionnalités. L'exemple 26-2, qui correspond au fichier *header.php*, contient ces détails communs. Il est inclus par les autres fichiers et reprend *functions.php*. Ceci implique qu'un seul `require_once` permet de tout regrouper dans chaque fichier.

Le fichier *header.php* commence par appeler la fonction *session_start*. Souvenez-vous de ce qu'indiquait le chapitre 12 : cette fonction définit une session qui permet de retenir certaines valeurs à mémoriser d'un fichier PHP à un autre.

La session démarrée, le programme vérifie ensuite que la variable de session `user` possède une valeur courante. Si c'est le cas, un utilisateur s'est connecté et la variable `$loggedin` (connecté) vaut `TRUE`.

Le code principal de paramétrage charge une feuille de styles, crée un élément canevas pour le logo, puis une section `div`. Il charge ensuite le fichier *javascript.js* (voir l'exemple 26-14 plus loin) pour obtenir les fonctions `O`, `S` et `C`, désormais bien connues. Ces fonctions font habituellement partie de notre fichier *OSC.js* mais pour réduire le nombre global de fichiers, nous les ajoutons au JavaScript utilisé pour créer le logo.

À partir de la valeur de `$loggedin`, un bloc `if` affiche un parmi deux ensembles de menus possibles. L'ensemble qui correspond à l'absence de connexion propose les options d'accueil, d'inscription et d'identification. Tandis que la version correspondant à l'identification déjà effectuée offre l'accès complet aux fonctionnalités du projet. En outre, dès qu'un utilisateur est connecté, son nom d'utilisateur est ajouté entre parenthèses au titre de la page et placé après l'entête de page. Nous pouvons librement faire référence à `$user` chaque fois que nous devons disposer du nom car, si l'utilisateur n'est pas connecté, cette variable est vide et n'a aucune influence sur la sortie.

Le style appliqué à la page se situe dans le fichier *styles.css* (exemple 26-13, détaillé à la fin du chapitre) qui implique aussi la création d'un large titre sur fond coloré et la mise en forme des liens en des boutons aux coins arrondis.

Exemple 26-2. En-tête : header.php

```
<?php
  session_start();

  echo "<!DOCTYPE html>\n<html><head>";

  require_once 'functions.php';
```

```
$userstr = ' (Invité)';

if (isset($_SESSION['user']))
{
  $user     = $_SESSION['user'];
  $loggedin = TRUE;
  $userstr  = " ($user)";
}
else $loggedin = FALSE;  // Utilisateur non identifié

echo "<title>$appname$userstr</title><link rel='stylesheet' "  .
     "href='styles.css' type='text/css'>"                      .
     "</head><body><center><canvas id='logo' width='638' "     .
     "height='96'>$appname</canvas></center>"                  .
     "<div class='appname'>$appname$userstr</div>"             .
     "<script src='javascript.js'></script>";

if ($loggedin)
{ // Si identifié
  echo "<br ><ul class='menu'>" .
       "<li><a href='members.php?view=$user'>Accueil</a></li>" .
       "<li><a href='members.php'>Membres</a></li>"            .
       "<li><a href='friends.php'>Amis</a></li>"               .
       "<li><a href='messages.php'>Messages</a></li>"          .
       "<li><a href='profile.php'>Modifier profil</a></li>"    .
       "<li><a href='logout.php'>Déconnecter</a></li></ul><br>";
}
else
{ // Sinon, non identifié
  echo ("<br><ul class='menu'>" .
        "<li><a href='index.php'>Accueil</a></li>"                    .
        "<li><a href='signup.php'>S'inscrire</a></li>"                .
        "<li><a href='login.php'>Se connecter</a></li></ul><br>"      .
        "<span class='info'>&#8658; Vous devez être connecté pour " .
        "voir cette page.</span><br><br>");
}
?>
```

L'utilisation de la balise `<br>` comme dans cet exemple constitue un moyen rapide et « sauvage » pour créer de l'espacement dans une disposition de page. Dans ce cas-ci, il fonctionne bien mais vous préférerez généralement utiliser des marges CSS pour affiner l'espacement autour des éléments.

Paramétrage : setup.php

À ce stade, les deux fichiers à inclure sont rédigés et prêts à l'emploi. Nous pouvons préparer les tables MySQL qu'ils utiliseront. C'est le rôle de l'exemple 26-3, nommé *setup.php*, que vous devez entrer et charger en premier lieu dans le navigateur avant d'appeler les autres fichiers ; sinon, vous obtiendrez de nombreuses erreurs issues de MySQL.

Nous créons de petites tables simples, nommées comme suit et avec les champs suivants :

- `members` : le nom d'utilisateur `username` (indexé) et le mot de passe `pass`
- `messages` : l'identifiant `id` (indexé), l'auteur `auth` (indexé), le destinataire `recip`, le type de message `pm`, le message `message`
- `friends` : le nom d'utilisateur `user` (indexé), le nom d'utilisateur de l'ami `friend`
- `profiles` : le nom d'utilisateur `user` (indexé), le texte « À propos de moi » `text`

Du fait que la fonction `createTable` vérifie d'abord si une table existe avant de la créer, vous pouvez appeler plusieurs fois le fichier sans générer d'erreur ni écraser les tables existantes.

Vous devrez très probablement ajouter plusieurs autres colonnes à ces tables si vous souhaitez étendre le projet. Dans ce cas, envoyez une instruction MySQL `DROP TABLE` avant de recréer une table.

Exemple 26-3. Paramétrage : setup.php

```
<!DOCTYPE html>
<html>
  <head>
    <title>Paramétrage base de données</title>
  </head>
  <body>

    <h3>Préparation...</h3>

<?php // Exemple 26-3: setup.php
  require_once 'functions.php';

  // Table des membres
  createTable('members',
              'user VARCHAR(16),
              pass VARCHAR(16),
              INDEX(user(6))');

  // Table des messages
  createTable('messages',
              'id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
              auth VARCHAR(16),
              recip VARCHAR(16),
              pm CHAR(1),
              time INT UNSIGNED,
              message VARCHAR(4096),
              INDEX(auth(6)),
              INDEX(recip(6))');

  // Table des amis
  createTable('friends',
```

```
            'user VARCHAR(16),
            friend VARCHAR(16),
            INDEX(user(6)),
            INDEX(friend(6))');

  // Table des profils de membres
  createTable('profiles',
            'user VARCHAR(16),
            text VARCHAR(4096),
            INDEX(user(6))');
?>

    <br>...terminée.
  </body>
</html>
```

Pour que cet exemple fonctionne, vous devez vous assurer d'avoir déjà créé la base de données précisée dans la variable `$dbname` de l'exemple 26-1, et d'avoir créé l'utilisateur de nom indiqué dans la variable `$dbuser` et de mot de passe dans `$dbpass`.

Page principale : index.php

Ce fichier est plutôt trivial mais nécessaire pour donner une page d'accueil au projet. La page affiche simplement un message de bienvenue. Dans une application mieux finie, c'est dans cette page que vous vanteriez les vertus de votre site pour encourager les inscriptions.

De manière incidente, comme vous avez créé toutes les tables MySQL et enregistré les fichiers à inclure, vous pouvez à présent charger dans le navigateur l'exemple 26-4, nommé *index.php*, pour obtenir votre première visite de la nouvelle application. La figure 26-1 illustre le résultat obtenu.

Exemple 26-4. Page principale : index.php

```
<?php
  require_once 'header.php';

  echo "<br><span class='main'>Bienvenue dans le $appname";

  if ($loggedin) echo ", $user, vous êtes connecté.";
  else           echo '.<br>Inscrivez-vous ou connectez-vous pour nous rejoindre.';
?>

    </span><br><br>
  </body>
</html>
```


Figure 26-1. La page principale du site

Remarquez que vous n'obtiendrez pas de suite ce résultat si vous entrez un à un les fichiers d'exemples dans l'ordre. Le dossier *nidderobin* et son contenu doivent avoir été copiés au complet du site d'accompagnement du livre dans le dossier racine du serveur pour obtenir ce résultat.

Inscription : signup.php

Nous avons ensuite besoin d'un module pour permettre aux gens de rejoindre le réseau. C'est le rôle de l'exemple 26-5, *signup.php*. Il s'agit d'un programme assez long mais nous en avons déjà vu une grande partie précédemment.

Examinons surtout le bloc de fin en HTML. C'est un simple formulaire qui permet d'entrer un nom d'utilisateur et un mot de passe. Notez toutefois la présence de la section `<span>` vide d'identifiant `'info'`. Elle constitue la destination de l'appel Ajax de ce programme qui vérifie que le nom d'utilisateur souhaité est disponible. Reportez-vous au chapitre 18 pour une description complète de son fonctionnement.

Vérifier la disponibilité d'un nom d'utilisateur

Revenons au début du programme. Vous voyez un bloc de JavaScript qui commence par la fonction `checkUser` (pour vérifier l'utilisateur). Celle-ci est appelée par l'évènement JavaScript `onBlur` lorsque la cible de saisie quitte le champ du nom d'utilisateur du formulaire. La fonction règle le contenu de la section `span` évoquée ci-dessus (d'identifiant `info`) à une chaine vide, ce qui la vide si elle contenait une valeur auparavant.

Ensuite, une requête au programme *checkuser.php* rapporte la disponibilité ou non du nom d'utilisateur `user`. Le résultat renvoyé par l'appel Ajax, un message convivial, vient se placer dans la section span `info`.

Après la section de JavaScript, apparait du code PHP, que vous devriez reconnaitre puisqu'il résulte de la discussion sur la validation de formulaire au chapitre 16. Cette section utilise aussi la fonction `sanitizeString` pour supprimer tout caractère malveillant éventuel avant de rechercher le nom d'utilisateur dans la base de données et, s'il n'est pas déjà pris, insère les nouveaux nom d'utilisateur et mot de passe dans la table.

Se connecter

Dès qu'il s'est inscrit correctement, l'utilisateur est invité à s'identifier pour se connecter. Une réponse plus fluide à ce stade consisterait à connecter automatiquement l'utilisateur dès qu'il s'est inscrit mais pour éviter de compliquer le code à l'excès, nous conservons à part les modules d'inscription et d'identification. N'hésitez pas à implémenter cette fonctionnalité si vous le souhaitez.

Ce fichier utilise la classe CSS `fieldname` pour présenter les champs de formulaire, pour les aligner proprement en colonnes. Lors du chargement dans un navigateur (aux côtés de *checkuser.php*, que nous verrons plus loin), ce programme donne un résultat comparable à celui de la figure 26-2, qui illustre l'appel Ajax ayant identifié que le nom d'utilisateur *Robin* est disponible. Si vous préférez que le champ de mot de passe n'affiche que des astérisques, alors remplacez son type `text` par le type `password`.

Exemple 26-5. S'inscrire : signup.php

```
<?php
  require_once 'header.php';

  echo <<<_END
  <script>
    function checkUser(user)
    { // Vérifier la disponibilité du nom d'utilisateur
      if (user.value == '')
      {
        O('info').innerHTML = ''
        return
      }
      // Construire la requête Ajax
      params  = "user=" + user.value
      request = new ajaxRequest()
      request.open("POST", "checkuser.php", true)
      request.setRequestHeader("Content-type", "application/x-www-form-urlencoded")
      request.setRequestHeader("Content-length", params.length)
      request.setRequestHeader("Connection", "close")

      request.onreadystatechange = function()
      {
        if (this.readyState == 4)
          if (this.status == 200)
            if (this.responseText != null)
              O('info').innerHTML = this.responseText
      }
      request.send(params)
    }

    function ajaxRequest()
    {
      try { var request = new XMLHttpRequest() }
      catch(e1) {
        try { request = new ActiveXObject("Msxml2.XMLHTTP") }
        catch(e2) {
          try { request = new ActiveXObject("Microsoft.XMLHTTP") }
          catch(e3) {
            request = false
      } } }
      return request
    }
  </script>
  <div class='main'><h3>Entrez vos détails d'inscription</h3>
_END;

  $error = $user = $pass = "";
  if (isset($_SESSION['user'])) destroySession();

  if (isset($_POST['user']))
  {
    $user = sanitizeString($_POST['user']);
    $pass = sanitizeString($_POST['pass']);

    if ($user == "" || $pass == "")
      $error = "Tous les champs ne sont pas remplis<br><br>";
    else
    {
      $result = queryMysql("SELECT * FROM members WHERE user='$user'");

      if ($result->num_rows)
        $error = "Ce nom d'utilisateur existe déjà<br><br>";
      else
      {
        queryMysql("INSERT INTO members VALUES('$user', '$pass')");
        die("<h4>Compte créé</h4>Identifiez-vous à nouveau.<br><br>");
      }
    }
  }

  echo <<<_END
    <form method='post' action='signup.php'>$error
    <span class='fieldname'>Nom d'utilisateur</span>
    <input type='text' maxlength='16' name='user' value='$user'
```

```
      onBlur='checkUser(this)'><span id='info'></span><br>
    <span class='fieldname'>Mot de passe</span>
    <input type='text' maxlength='16' name='pass'
      value='$pass'><br>
_END;
?>

    <span class='fieldname'> </span>
    <input type='submit' value='Je m&apos;inscris'>
    </form></div><br>
  </body>
</html>
```

Figure 26-2. La page d'inscription d'un nouvel utilisateur

Sur un serveur de production, je conseille de ne jamais enregistrer les mots de passe en clair comme je l'ai fait ici (pour des raisons de simplicité et de manque de place). Au lieu de cela, « salez-les », faites-les passer par une fonction de hachage à sens unique, puis enregistrez-les. Reportez-vous au chapitre 12 pour les détails de la procédure.

Vérifier un utilisateur : checkuser.php

Pour que le programme *signup.php* fonctionne, il a besoin de son pendant, *checkuser.php*, qui examine la base de données et vérifie la disponibilité d'un nom d'utilisateur. Il renvoie une chaine qui indique s'il est disponible ou s'il est déjà pris. Comme il se base sur les fonctions `sanitizeString` et `queryMysql`, le programme inclut d'abord le fichier *functions.php*.

Ensuite, si la variable `$_POST user` possède une valeur, la fonction la recherche dans la base de données et, s'il existe un tel nom d'utilisateur identique, le programme renvoie « Nom d'utilisateur déjà pris » ou, s'il n'en existe pas, « Nom d'utilisateur disponible ». L'examen du résultat de la fonction `mysql_num_rows` suffit amplement pour cela car elle renvoie `0` pour non trouvé et `1` pour trouvé.

Les entités HTML `&#x2718;` et `&#x2714;` servent également de préfixe à la chaine renvoyée pour afficher une croix ou une coche.

Exemple 26-6. Vérifier un utilisateur : checkuser.php

```
<?php
  require_once 'functions.php';

  if (isset($_POST['user']))
  {
    $user   = sanitizeString($_POST['user']);
    $result = queryMysql("SELECT * FROM members WHERE user='$user'");

    if ($result->num_rows)
      echo  "<span class='taken'> &#x2718; " .
            "Nom d'utilisateur déjà pris</span>";
    else
      echo "<span class='available'> &#x2714; " .
           "Nom d'utilisateur disponible</span>";
  }
?>
```

S'identifier : login.php

À partir du moment où les utilisateurs sont en mesure de s'inscrire sur le site, l'exemple 26-7, *login.php*, fournit le code nécessaire pour leur permettre de s'identifier et de se connecter. Comme la page d'inscription, celle-ci présente un formulaire HTML simple et un peu de contrôle d'erreur, en plus de l'appel à la fonction `sanitizeString` avant d'interroger la base de données MySQL.

La principale particularité à noter ici réside dans le fait qu'à la réussite des vérifications du nom d'utilisateur et du mot de passe, les variables de session `user` et `pass` reçoivent les valeurs correspondantes. Tant que la session courante demeure active, ces variables demeurent disponibles à tous les programmes du projet, ce qui leur permet de donner automatiquement l'accès aux utilisateurs connectés.

Examinez également l'utilisation de la fonction `die` lors de la réussite de l'identification. Sa présence se justifie par le fait qu'elle combine en une seule les commandes `echo` et `exit`, ce qui épargne une ligne de code. Pour la mise en style, ce programme (comme la plupart des autres fichiers) applique la classe `main` pour mettre en retrait le contenu par rapport au bord gauche de la fenêtre de navigation.

Lorsque vous appelez ce programme dans votre navigateur, il prend l'allure de la figure 26-3. Remarquez que le type d'entrée `password` sert ici à masquer le mot de passe à l'aide d'astérisques pour éviter que quiconque regardant par-dessus l'épaule de l'utilisateur voie son mot de passe.

Exemple 26-7. S'identifier : login.php

```
<?php
  require_once 'header.php';
  echo "<div class='main'><h3>Entrez vos informations de connexion</h3>";
  $error = $user = $pass = "";

  if (isset($_POST['user']))
  {
    $user = sanitizeString($_POST['user']);
    $pass = sanitizeString($_POST['pass']);

    if ($user == "" || $pass == "")
        $error = "Tous les champs ne sont pas remplis<br>";
    else
    {
      $result = queryMySQL("SELECT user,pass FROM members
        WHERE user='$user' AND pass='$pass'");

      if ($result->num_rows == 0)
      {
        $error = "<span class='error'>Utilisateur ou mot de passe
                  non valide</span><br><br>";
      }
      else
      {
        $_SESSION['user'] = $user;
        $_SESSION['pass'] = $pass;
        die("Vous êtes connecté. Cliquez <a href='members.php?view=$user'>" .
            "ici</a> pour continuer.<br><br>");
      }
    }
  }

  echo <<<_END
    <form method='post' action='login.php'>$error
    <span class='fieldname'>Utilisateur</span><input type='text'
      maxlength='16' name='user' value='$user'><br>
    <span class='fieldname'>Mot de passe</span><input type='password'
      maxlength='16' name='pass' value='$pass'>
_END;
?>
```



```
    <br>
    <span class='fieldname'> </span>
    <input type='submit' value='Me connecter'>
    </form><br></div>
  </body>
</html>
```

Figure 26-3. La page d'identification et de connexion de l'utilisateur

Définition du profil : profile.php

Une des premières choses que les nouveaux utilisateurs veulent faire après s'être inscrits et connectés consiste à créer leur profil, ce dont se charge l'exemple 26-8, *profile.php*. Vous trouverez certainement du code intéressant dans celui-ci, notamment des routines pour téléverser, redimensionner et rendre des images plus nettes.

Commençons par l'examen de la partie HTML principale en fin de code. Celle-ci ressemble aux formulaires que nous venons de voir mais, cette fois, le formulaire porte le paramètre `enctype='multipart/form-data'`. Ceci nous permet d'envoyer plusieurs types de données à la fois, ce qui permet la publication d'une image en même temps que du texte. Il y a aussi un type d'entrée `file`, qui provoque l'affichage d'un bouton *Parcourir* pour que l'utilisateur puisse sélectionner le fichier à téléverser.

Lors de la soumission du formulaire, le code du début du programme s'exécute. Sa première tâche consiste à s'assurer qu'un utilisateur est connecté avant de permettre l'exécution de la suite du programme. Ce n'est qu'ensuite que la page s'affiche.

Ajouter le texte « À propos de moi »

Ensuite a lieu la vérification de la variable Post `text` pour déterminer si du texte a été publié dans le programme. Dans l'affirmative, il est aseptisé et toutes les longues séquences d'espaces (y compris les retours charriot et les nouvelles lignes) sont remplacées par une seule espace. Cette fonction intègre une double vérification de sécurité, pour garantir que l'utilisateur existe réellement dans la base de données et qu'aucune tentative de piratage ne réussisse avant l'insertion de ce texte dans la base de données, où il devient l'information « À propos de moi » de l'utilisateur.

Si aucun texte n'a été publié, le programme interroge la base de données pour récupérer un éventuel texte existant déjà pour préremplir la zone de texte étendue (`textarea`) afin que l'utilisateur puisse le modifier.

Ajouter une image de profil

Ensuite, nous passons à la section qui teste la variable système `$_FILES` pour vérifier si une image a été téléversée. Si c'est le cas, nous créons une variable chaine nommée `$saveto` à partir du nom de l'utilisateur, suivi de la valeur *.jpg*. Ainsi, si l'utilisateur Gilles dépose une image, `$saveto` reçoit la valeur *Gilles.jpg*. C'est le nom de fichier sous lequel l'image reçue sera enregistrée dans le profil de l'utilisateur.

À la suite de ceci, nous examinons le type de l'image et nous ne l'acceptons que si elle est de type *jpeg*, *png* ou *gif*. En cas de réussite, la variable `$src` reçoit l'image reçue à l'aide d'une des fonctions `imagecreatefrom` selon le type d'image reçue. L'image se trouve actuellement dans un format brut que PHP peut traiter. Si l'image n'est pas d'un type accepté, nous attribuons la valeur `FALSE` au drapeau `$typeok`, ce qui empêche le traitement de la section finale du code de chargement de l'image.

Traiter l'image

Nous stockons d'abord les dimensions de l'image dans les variables `$w` et `$h` à l'aide de l'instruction suivante, qui représente un moyen rapide d'affecter des valeurs d'un tableau à des variables distinctes :

```
list($w, $h) = getimagesize($saveto);
```

Ensuite, à l'aide de la valeur de `$max` (définie à `100`), nous calculons de nouvelles dimensions qui produisent une image de même rapport largeur sur hauteur, mais avec aucune dimension supérieure à 100 pixels. Ces dimensions viennent peupler les variables `$tw` et `$th`. Pour des vignettes plus petites ou plus grandes, modifiez la valeur de `$max`.

Un appel de la fonction `imagecreatetruecolor` permet ensuite de créer un canevas vide de `$tw` de large et `$th` de haut dans `$tmp`. L'appel de la fonction `imagecopyresampled` qui suit ré-échantillonne l'image de `$src` pour la déposer dans la nouvelle variable `$tmp`. Le ré-échantillonnage conduit parfois à une copie légèrement floue. Par conséquent, le code qui suit fait appel à la fonction `imagecovolution` pour rendre l'image un peu plus nette.

Pour finir, nous enregistrons l'image sous forme d'un fichier *jpeg* à l'emplacement défini par la variable `$saveto` et nous supprimons les canevas de l'image initiale et de sa copie redimensionnée à l'aide de la fonction `imagedestroy`, pour restituer la mémoire devenue inutile.

Afficher le profil courant

Dernière opération et non la moindre, afin que l'utilisateur puisse voir son profil avant de le modifier, nous appelons la fonction `showProfile` du fichier *functions.php* avant de renvoyer le formulaire en HTML. Si aucun profil n'existe encore, rien ne s'affiche.

Lorsqu'une image de profil s'affiche, nous appliquons du CSS pour lui donner une bordure, de l'ombre et une marge à sa droite, pour séparer l'image et le texte de profil. Le chargement de l'exemple 26-8 donne un résultat comparable à celui de la figure 26-4, où la zone `textarea` a été remplie d'un texte « À propos de moi ».

Exemple 26-8. Définition du profil : profile.php

```
<?php
  require_once 'header.php';

  if (!$loggedin) die();

  echo "<div class='main'><h3>Votre profil</h3>";

  $result = queryMysql("SELECT * FROM profiles WHERE user='$user'");

  if (isset($_POST['text']))
  {
    $text = sanitizeString($_POST['text']);
    $text = preg_replace('/\s\s+/', ' ', $text);

    if ($result->num_rows)
        queryMysql("UPDATE profiles SET text='$text' where user='$user'");
    else queryMysql("INSERT INTO profiles VALUES('$user', '$text')");
  }
  else
  {
    if ($result->num_rows)
    {
      $row  = $result->fetch_array(MYSQLI_ASSOC);
      $text = stripslashes($row['text']);
```

```
  }
  else $text = "";
}

$text = stripslashes(preg_replace('/\s\s+/', ' ', $text));

if (isset($_FILES['image']['name']))
{
  $saveto = "$user.jpg";
  move_uploaded_file($_FILES['image']['tmp_name'], $saveto);
  $typeok = TRUE;

  switch($_FILES['image']['type'])
  {
    case "image/gif":   $src = imagecreatefromgif($saveto); break;
    case "image/jpeg":  // Images jpeg normal et progressif
    case "image/pjpeg": $src = imagecreatefromjpeg($saveto); break;
    case "image/png":   $src = imagecreatefrompng($saveto); break;
    default:            $typeok = FALSE; break;
  }

  if ($typeok)
  {
    list($w, $h) = getimagesize($saveto);

    $max = 100;
    $tw  = $w;
    $th  = $h;

    if ($w > $h && $max < $w)
    {
      $th = $max / $w * $h;
      $tw = $max;
    }
    elseif ($h > $w && $max < $h)
    {
      $tw = $max / $h * $w;
      $th = $max;
    }
    elseif ($max < $w)
    {
      $tw = $th = $max;
    }

    $tmp = imagecreatetruecolor($tw, $th);
    imagecopyresampled($tmp, $src, 0, 0, 0, 0, $tw, $th, $w, $h);
    imageconvolution($tmp, array(array(-1, -1, -1),
      array(-1, 16, -1), array(-1, -1, -1)), 8, 0);
    imagejpeg($tmp, $saveto);
    imagedestroy($tmp);
    imagedestroy($src);
  }
}
```



```
  showProfile($user);

  echo <<<_END
    <form method='post' action='profile.php' enctype='multipart/form-data'>
    <h3>Entrez ou modifiez votre texte de profil et une image</h3>
    <textarea name='text' cols='50' rows='3'>$text</textarea><br>
_END;
?>

    Image: <input type='file' name='image' size='14'>
    <input type='submit' value='Enregistrer le profil'>
    </form></div><br>
  </body>
</html>
```

Figure 26-4. Modification d'un profil d'utilisateur

Les membres : members.php

Le code de l'exemple 26-9, *members.php*, permet aux utilisateurs de trouver d'autres membres et de les ajouter dans la liste des amis (ou de les supprimer s'ils font déjà partie des amis). Ce programme possède deux modes : le premier énumère tous les membres et leurs relations avec l'utilisateur, tandis que le second affiche le profil d'un membre.

Afficher le profil d'un utilisateur

Le code du second mode vient en premier lieu et teste le contenu de la variable Get `view`. Si elle existe, l'utilisateur souhaite voir le profil de quelqu'un, ce que le programme réalise avec la fonction `showProfile`, avec des liens vers les amis de l'utilisateur et leurs messages.

Ajouter et supprimer des amis

Ensuite, le programme teste deux variables Get, `add` et `remove`. Si l'une d'elles est définie, alors l'utilisateur souhaite ajouter ou supprimer un ami, respectivement. Pour réaliser cela, nous recherchons l'utilisateur dans la table MySQL `friends` et soit nous ajoutons un nom d'utilisateur ami, soit nous le supprimons de la table.

Bien entendu, chaque variable publiée passe à la moulinette de `sanitizeString` pour garantir qu'elle peut être utilisée dans MySQL en toute sécurité.

Énumérer tous les membres

La dernière section du code envoie une requête SQL pour énumérer tous les noms d'utilisateurs. Le code place le numéro renvoyé dans la variable `$num` avant de sortir l'entête de page.

Une boucle `for` itère parmi tous les membres pour en récupérer les détails et rechercher ensuite dans la table `friends` pour voir s'ils sont soit suivis par l'utilisateur ou s'ils suivent ce dernier. Si une personne est à la fois suivie et un suiveur, elle est considérée comme un ami mutuel.

La variable `$t1` est différente de zéro quand l'utilisateur suit un autre membre et `$t2` est différente de zéro quand un autre membre suit l'utilisateur. En fonction de ces valeurs, un texte vient s'afficher après chaque nom d'utilisateur pour indiquer, s'il y a lieu, la relation avec l'utilisateur courant.

Des icônes s'affichent également pour représenter les relations. Une flèche à double pointe (entité HTML `&harr ;`) indique que les utilisateurs sont des amis mutuels, la flèche pointée vers la gauche (`&larr ;`) indique que l'utilisateur suit un autre membre et une flèche pointée vers la droite (`&rarr ;`) indique qu'un autre membre suit l'utilisateur courant.

Pour terminer, selon que l'utilisateur suit un autre membre, nous fournissons un lien pour permettre d'ajouter ou de supprimer tel membre en tant qu'ami.

Lors de l'appel de l'exemple 26-9 dans un navigateur, il produit des résultats illustrés par la figure 26-5. Voyez comment l'utilisateur est invité à « suivre » un membre qui ne le suit pas, mais si le membre du réseau suit déjà l'utilisateur, un lien nommé « récip. » est proposé pour rendre la réciproque de la relation. Dans le cas où l'utilisateur suit déjà un autre membre, l'utilisateur peut cliquer sur « suppr. » pour terminer le suivi.

Exemple 26-9. Les membres : members.php

```
<?php
  require_once 'header.php';

  if (!$loggedin) die();

  echo "<div class='main'>";

  if (isset($_GET['view']))
  {
    $view = sanitizeString($_GET['view']);

    if ($view == $user)
    {
      $name = "Vous";
      echo "<h3>Votre profil</h3>";
      showProfile($view);
      echo "<a class='button' href='messages.php?view=$view'>" .
           "Voir vos messages</a><br><br>";
    }
    else
    {
      $name = "$view";
      echo "<h3>Profil de $name</h3>";
      showProfile($view);
      echo "<a class='button' href='messages.php?view=$view'>" .
           "Voir les messages de $name</a><br><br>";
    }
    die("</div></body></html>");
  }

  if (isset($_GET['add']))
  {
    $add = sanitizeString($_GET['add']);

    $result = queryMysql("SELECT * FROM friends WHERE user='$add' AND
friend='$user'");
    if (!$result->num_rows)
      queryMysql("INSERT INTO friends VALUES ('$add', '$user')");
  }
  elseif (isset($_GET['remove']))
  {
```

```
    $remove = sanitizeString($_GET['remove']);
    queryMysql("DELETE FROM friends WHERE user='$remove' AND friend='$user'");
  }

  $result = queryMysql("SELECT user FROM members ORDER BY user");
  $num    = $result->num_rows;

  echo "<h3>Autres membres</h3><ul>";

  for ($j = 0 ; $j < $num ; ++$j)
  {

    $row = $result->fetch_array(MYSQLI_ASSOC);
    if ($row['user'] == $user) continue;

    echo "<li><a href='members.php?view=" .
      $row['user'] . "'>" . $row['user'] . "</a>";
    $follow = "suivre";

    $result1 = queryMysql("SELECT * FROM friends WHERE
      user='" . $row['user'] . "' AND friend='$user'");
    $t1      = $result1->num_rows;
    $result1 = queryMysql("SELECT * FROM friends WHERE
      user='$user' AND friend='" . $row['user'] . "'");
    $t2      = $result1->num_rows;

    if (($t1 + $t2) > 1) echo " &harr; est un ami mutuel";
    elseif ($t1)         echo " &larr; vous le suivez";
    elseif ($t2)       { echo " &rarr; vous suit";
      $follow = "récip."; }

    if (!$t1) echo " [<a href='members.php?add="    .$row['user'] . "'>$follow</a>]";
    else      echo " [<a href='members.php?remove=".$row['user'] . "'>suppr.</a>]";
  }
?>

    </ul></div>
  </body>
</html>
```


Figure 26-5. Utilisation du module des membres

Des milliers d'utilisateurs, voire des centaines de milliers, peuvent figurer sur un serveur de production. Par conséquent, vous devrez certainement modifier ce programme pour y ajouter une recherche sur le texte « À propos de moi », prendre en charge la présentation par pages des sorties à l'écran et ainsi de suite.

Amis : friends.php

Le module qui affiche les amis et les suiveurs d'un utilisateur figure dans l'exemple 26-10, *friends.php*. Le programme interroge la table `friends` de la même manière que le programme *members.php* mais seulement pour un seul utilisateur. Il affiche ensuite les amis mutuels et les suiveurs de l'utilisateur ainsi que les personnes qu'il suit.

Tous les suiveurs sont mémorisés dans un tableau nommé `$followers`, et toutes les personnes suivies viennent se placer dans un tableau nommé `$following`. Un extrait de code particulier permet ensuite d'extraire tous ceux à la fois suivi par l'utilisateur et qui le suivent, comme suit :

```
$mutual = array_intersect($followers, $following);
```

La fonction `array_intersect` extrait tous les membres communs des deux tableaux et renvoie un nouveau tableau, dans la variable `$mutual`, qui ne contient que les personnes recherchées. À partir de là, il est possible d'exploiter la fonction `array_diff` avec les tableaux `$followers` et `$following` pour ne conserver que les personnes qui ne sont pas des amis réciproques, comme suit :

```
$followers = array_diff($followers, $mutual);
$following = array_diff($following, $mutual);
```

Nous obtenons ainsi le tableau `$mutual` qui ne contient que les amis réciproques, `$followers` avec seulement les suiveurs (mais pas les amis mutuels) et `$following`, avec seulement les personnes suivies (sans les amis mutuels).

Forts de ces tableaux, il suffit ensuite d'afficher chaque catégorie de membres, comme illustré à la figure 26-6. La fonction PHP `sizeof` renvoie le nombre d'élément dans un tableau, ce qui permet de déclencher du code quand cette taille diffère de zéro (donc des amis de ce type existent). Remarquez l'utilisation des variables `$name1`, `$name2` et `$name3` aux endroits appropriés pour que le code puisse indiquer que vous lisez la liste de vos propres amis, à l'aide des mots *Vos* et *Vous*, au lieu d'afficher simplement le nom d'utilisateur. La ligne de commentaire, dé-commentée, permet si vous le souhaitez, d'afficher les informations de profil d'utilisateur sur cet écran.

Exemple 26-10. Amis : friends.php

```
<?php
  require_once 'header.php';

  if (!$loggedin) die();

  if (isset($_GET['view'])) $view = sanitizeString($_GET['view']);
  else                      $view = $user;

  if ($view == $user)
  {
    $name1 = $name2 = "Vos";
    $name3 =          "Vous";
  }
  else
  {
    $name1 = "<a href='members.php?view=$view'>$view</a>'s";
    $name2 = "$view:";
    $name3 = "$view";
  }

  echo "<div class='main'>";

  // Ôtez le commentaire de cette ligne pour afficher le profil d'utilisateur
  // showProfile($view);
```



```
  $followers = array();
  $following = array();

  $result = queryMysql("SELECT * FROM friends WHERE user='$view'");
  $num    = $result->num_rows;

  for ($j = 0 ; $j < $num ; ++$j)
  {
    $row          = $result->fetch_array(MYSQLI_ASSOC);
    $followers[$j] = $row['friend'];
  }

  $result = queryMysql("SELECT * FROM friends WHERE friend='$view'");
  $num    = $result->num_rows;

  for ($j = 0 ; $j < $num ; ++$j)
  {
    $row          = $result->fetch_array(MYSQLI_ASSOC);
    $following[$j] = $row['user'];
  }

  $mutual    = array_intersect($followers, $following);
  $followers = array_diff($followers, $mutual);
  $following = array_diff($following, $mutual);
  $friends   = FALSE;

  if (sizeof($mutual))
  { // L'opérateur ternaire permet de vous distinguer de l'ami
    echo "<span class='subhead'>". ($name2 == 'Vos' ? "Vos amis réciproques" :
         "Amis réciproques de $name2") . "</span><ul>";
    foreach($mutual as $friend)
      echo "<li><a href='members.php?view=$friend'>$friend</a>";
    echo "</ul>";
    $friends = TRUE;
  }

  if (sizeof($followers))
  { // Suiveurs
    echo "<span class='subhead'>". ($name2 == 'Vos' ? "Vos suiveurs" :
         "Suiveurs de $name2") . "</span><ul>";
    foreach($followers as $friend)
      echo "<li><a href='members.php?view=$friend'>$friend</a>";
    echo "</ul>";
    $friends = TRUE;
  }

  if (sizeof($following))
  { // Suivis
    echo "<span class='subhead'>". ($name3 == 'Vous' ? "Vous suivez" :
         "$name3 suit") . "</span><ul>";
    foreach($following as $friend)
      echo "<li><a href='members.php?view=$friend'>$friend</a>";
    echo "</ul>";
    $friends = TRUE;
  }
```

```
  if (!$friends) echo "<br>Vous n'avez pas encore d'ami.<br><br>";

  echo "<a class='button' href='messages.php?view=$view'>" .
       "Voir " . strtolower($name2) . " messages</a>";
?>

    </div><br>
  </body>
</html>
```

Figure 26-6. Affichage des amis et suiveurs de l'utilisateur

Messages : messages.php

L'exemple 26-11 reprend le dernier des modules principaux, *messages.php*. Le programme commence par vérifier si un message a été publié dans la variable `text`. Dans l'affirmative, il l'insère dans la table `messages`. En même temps, il mémorise la valeur de `pm`, qui indique si le message est privé ou public. Un `0` représente un message public et un `1` qu'il est privé.

Le programme affiche ensuite le profil de l'utilisateur et un formulaire d'entrée de message, ainsi que des boutons à option pour choisir le mode de message public ou privé. Vient ensuite l'affichage de tous les messages, selon qu'ils soient publics ou privés. S'ils sont publics, tous les utilisateurs peuvent les voir, tandis que les messages privés ne sont visibles que par leur expéditeur (*sender*) et leur destinataire (*recipient*). Tout ceci est géré à l'aide de deux requêtes de la base de données MySQL. En outre, lorsqu'un message est privé, il est précédé des mots *a chuchoté* et présenté en italiques.

Enfin, le programme affiche deux liens pour actualiser les messages dans le cas où un autre utilisateur en aurait publié un entretemps, et pour voir les amis de l'utilisateur. L'astuce des variables `$name1` et `$name2` sert à nouveau ici pour que, lorsque vous voyez vos propres messages, le mot *Vos* s'affiche à la place du nom de l'utilisateur.

La figure 26-7 illustre les résultats produits par le programme dans un navigateur. Remarquez que les utilisateurs qui visualisent leurs propres messages ont la possibilité de les effacer à l'aide d'un lien.

Exemple 26-11. Messages : messages.php

```
<?php // Exemple 26-11 : messages.php
  require_once 'header.php';

  if (!$loggedin) die();

  if (isset($_GET['view'])) $view = sanitizeString($_GET['view']);
  else                      $view = $user;

  if (isset($_POST['text']))
  {
    $text = sanitizeString($_POST['text']);

    if ($text != "")
    {
      $pm   = substr(sanitizeString($_POST['pm']),0,1);
      $time = time();
      queryMysql("INSERT INTO messages VALUES(NULL, '$user',
        '$view', '$pm', $time, '$text')");
    }
  }

  if ($view != "")
  {
    if ($view == $user) $name1 = $name2 = "Vos";
    else
    {
      $name1 = "<a href='members.php?view=$view'>$view</a> -";
      $name2 = "$view -";
    }
```

```
    echo "<div class='main'><h3>$name1 messages</h3>";
    showProfile($view);

    echo <<<_END
      <form method='post' action='messages.php?view=$view'>
      Entrez un message ici :<br>
      <textarea name='text' cols='40' rows='3'></textarea><br>
      Public<input type='radio' name='pm' value='0' checked='checked'>
      Privé<input type='radio' name='pm' value='1'>
      <input type='submit' value='Publier'></form><br>
_END;

    if (isset($_GET['erase']))
    {
      $erase = sanitizeString($_GET['erase']);
      queryMysql("DELETE FROM messages WHERE id=$erase AND recip='$user'");
    }

    $query  = "SELECT * FROM messages WHERE recip='$view' ORDER BY time DESC";
    $result = queryMysql($query);
    $num    = $result->num_rows;

    for ($j = 0 ; $j < $num ; ++$j)
    {
      $row = $result->fetch_array(MYSQLI_ASSOC);

      if ($row['pm'] == 0 || $row['auth'] == $user || $row['recip'] == $user)
      {
        echo date('d/m \à H:i', $row['time']);
        echo ", <a href='messages.php?view=" . $row['auth'] . "'>" . $row['auth']. "</
a> ";

        if ($row['pm'] == 0)
          echo "a écrit : &quot;" . $row['message'] . "&quot; ";
        else
          echo "a chuchoté : <span class='whisper'>&quot;" .
            $row['message']. "&quot;</span> ";

        if ($row['recip'] == $user)
          echo "[<a href='messages.php?view=$view" .
               "&erase=" . $row['id'] . "'>suppr.</a>]";

        echo "<br>";
      }
    }
  }

  if (!$num) echo "<br><span class='info'>Aucun message</span><br><br>";

  echo "<br><a class='button' href='messages.php?view=$view'>Actualiser</a>";
?>
```



```
    </div><br>
  </body>
</html>
```

Figure 26-7. Le module des messages

Déconnexion : logout.php

L'exemple 26-12 contient le code de notre dernier ingrédient de la recette de réseau social, *logout.php*, la page de déconnexion qui clôture une session et efface les données associées, ainsi que les cookies. La figure 26-8 illustre le résultat de l'appel de ce programme dans un navigateur. Le programme se voit invité à cliquer sur un lien pour l'amener à la page d'accueil avant identification, c'est-à-dire sans les liens du haut de la fenêtre, réservés aux membres connectés. Bien entendu, n'hésitez pas à rédiger du code JavaScript ou PHP de redirection pour mettre ceci en œuvre, ce qui constitue une excellente idée d'ailleurs, car cela permet d'assurer une procédure propre à la déconnexion.

Exemple 26-12. Déconnexion : logout.php

```
<?php
  require_once 'header.php';

  if (isset($_SESSION['user']))
  {
    destroySession();
    echo "<div class='main'>Vous êtes déconnecté. Cliquez " .
         "<a href='index.php'>ici</a> pour actualiser la page.";
  }
  else echo "<div class='main'><br>" .
            "Vous ne pouvez vous déconnecter car vous n'êtes pas
            encore connecté";
?>

    <br><br></div>
  </body>
</html>
```

Figure 26-8. La page de déconnexion

Feuille de style : styles.css

L'exemple 26-13 contient la feuille de styles utilisée dans le projet. Celle-ci comporte une série de déclarations, dont les rôles sont les suivants :

`*`
: Règle la famille et la taille de police de l'ensemble du projet à l'aide du sélecteur universel.

`body`
: Définit la largeur de la fenêtre du projet, la centre horizontalement, précise une couleur de fond et une bordure.

`html`
: Règle la couleur de fond de la section HTML.

`img`
: Donne à toutes les images une bordure, une ombre et une marge à droite.

`li a` *et* `.button`
: Supprime le soulignement des hyperliens de toutes les balises `<a>` présentes dans un élément `<li>` et de tous les éléments qui emploient la classe `button`.

`li a:hover` *et* `.button:hover`
: Définit la couleur du texte au survol de la souris des liens sous des éléments `<li>` et des éléments de classe `button`.

`.appname`
: Règle les propriétés de l'entête (qui utilise la classe `appname` du nom d'application), notamment le centrage, les couleurs d'arrière-plan et de police de texte, la famille et la taille de police, ainsi que la marge intérieure.

`.fieldname`
: Rend d'abord flottants les éléments de classe `fieldname` (étiquettes des champs de formulaire) et en définit la largeur.

`.main`
: Cette classe applique un retrait aux éléments qui l'utilisent.

`.info`
: Cette classe sert à l'affichage d'informations importantes. Elle précise la couleur de fond et de texte, applique une bordure et une marge intérieure, puis place en retrait les éléments qui l'utilisent.

`.menu li` *et* `.button`
: Ces déclarations garantissent que tous les éléments `<li>` et de classe `button` s'affichent les uns à la suite des autres, possèdent une marge intérieure, reçoivent une bordure, une couleur de fond et de texte, une marge droite, des coins de bordures arrondis et une ombre, pour leur donner l'allure de boutons.

`.subhead`
: Met en évidence des sections de texte.

`.taken`, `.available`, `.error` *et* `.whisper`
: (Respectivement pour les éléments pris, disponibles, d'erreur et chuchoté.) Ces déclarations définissent les couleurs et styles de police à utiliser pour ces différentes informations.

`#logo`
: Ces règles appliquent un style au texte du logo en guise de solution de repli pour les navigateurs non compatibles avec HTML5, où il est impossible de créer un canevas.

Exemple 26-13. Feuille de style : styles.css

```
* {
  font-family:verdana,sans-serif;
  font-size  :14pt;
}

body {
  width     :700px;
  margin    :20px auto;
  background:#f8f8f8;
  border    :1px solid #888;
}

html {
  background:#fff
}

img {
  border            :1px solid black;
  margin-right      :15px;
  -moz-box-shadow   :2px 2px 2px #888;
  -webkit-box-shadow:2px 2px 2px #888;
  box-shadow        :2px 2px 2px #888;
}

li a, .button {
  text-decoration:none;
}

li a:hover, .button:hover {
  color:green;
}

.appname {
  text-align :center;
  background :#eb8;
  color      :#40d;
  font-family:helvetica;
  font-size  :20pt;
  padding    :4px;
}

.fieldname {
  float:left;
  width:140px;
}

.main {
  margin-left:40px;
}
```



```
.info {
  background :lightgreen;
  color      :blue;
        border     :1px solid green;
        padding    :5px 10px;
  margin-left:40px;
}

.menu li, .button {
  display           :inline;
  padding           :4px 6px;
  border            :1px solid #777;
  background        :#ddd;
  color             :#d04;
  margin-right      :8px;
  border-radius     :5px;
  -moz-box-shadow   :2px 2px 2px #888;
  -webkit-box-shadow:2px 2px 2px #888;
  box-shadow        :2px 2px 2px #888;
}

.subhead {
  font-weight:bold;
}

.taken, .error {
  color:red;
}

.available {
  color:green;
}

.whisper {
  font-style:italic;
  color     :#006600;
}

#logo {
  font-family:Georgia;
  font-weight:bold;
  font-style:italic;
  font-size :97px;
}
```

Code côté client : javascript.js

Pour terminer, voici le fichier *javascript.js* de l'exemple 26-14, qui contient les fonctions `O`, `S` et `C` utilisées tout au long du livre, en plus d'un peu de code pour dessiner le logo du site dans un canevas HTML5, comme évoqué au chapitre 23.

Exemple 26-14. Code côté client : javascript.js

```
canvas               = O('logo')
context              = canvas.getContext('2d')
context.font         = 'bold italic 97px Georgia'
context.textBaseline = 'top'
image                = new Image()
image.src            = 'robin.gif'

image.onload = function()
{
  gradient = context.createLinearGradient(0, 0, 0, 89)
  gradient.addColorStop(0.00, '#faa')
  gradient.addColorStop(0.66, '#f00')
  context.fillStyle = gradient
  context.fillText(  "Nid de R  bin", 0, 0)
  context.strokeText("Nid de R  bin", 0, 0)
  context.drawImage(image, 410, 22)
}

function O(obj)
{
  if (typeof obj == 'object') return obj
  else return document.getElementById(obj)
}

function S(obj)
{
  return O(obj).style
}

function C(name)
{
  var elements = document.getElementsByTagName('*')
  var objects  = []
  for (var i = 0 ; i < elements.length ; ++i)
    if (elements[i].className == name)
      objects.push(elements[i])
  return objects
}
```

Voilà ! Si vous avez pris la peine de rédiger patiemment tout ce code et les nombreux exemples de ce livre, ou simplement téléchargé les codes sources du site d'accompagnement de ce livre pour les examiner attentivement, alors je suis ravi de vous avoir aidé à les assimiler. Quoi qu'il en soit, je vous remercie d'avoir lu ce livre.

Toutefois, avant que nous ne nous quittions, que vous n'exploitiez vos nouvelles compétences du web au sens large, je vous propose de parcourir les annexes suivantes car elles contiennent de nombreuses informations qui devraient vous être utiles.

ANNEXE A

Réponses aux questions en fin de chapitre

Réponses du chapitre 1

1. Les quatre (au moins) composants nécessaires pour créer une page web totalement dynamique sont un serveur web (par ex. Apache), un langage de script côté serveur (PHP), une base de données (MySQL) et un langage de script côté clients (JavaScript).
2. L'acronyme HTML correspond à l'anglais *HyperText Markup Language*, qui signifie langage de balisage hypertexte. Il désigne la page web en elle-même, y compris le texte et les balises qu'elle contient.
3. Comme à peu près tous les moteurs de base de données, MySQL accepte des commandes en SQL (*Structured Query Language*), un langage de requête structuré, qui permet à tout utilisateur, y compris un programme en PHP, de communiquer avec MySQL.
4. PHP agit du côté du serveur, tandis que JavaScript fonctionne du côté du client. PHP communique avec une base de données pour stocker et retrouver des données, mais il ne peut modifier rapidement et dynamiquement la page web chez l'utilisateur. JavaScript présente l'inverse de ces avantages et inconvénients.
5. CSS correspond à *Cascading style sheets*, ou feuilles de style en cascade : CSS définit les règles de style et de disposition appliquées aux éléments d'un document HTML.
6. Les nouveaux éléments les plus intéressants de HTML5 sont probablement `<audio>`, `<video>` et `<canvas>`, mais il y en a bien d'autres, comme `<article>`, `<summary>`, `<footer>`, …
7. Certaines de ces technologies sont contrôlées par des sociétés qui acceptent les rapports de bogue et réparent les erreurs comme toute société éditrice de logiciels. Mais les logiciels en code source ouvert dépendent aussi d'une communauté, donc un rapport de bogue peut aussi être traité par un utilisateur qui comprend suffisamment bien le programme. Peut-être qu'un jour vous aussi serez amené à corriger des bogues dans un outil en source ouverte.

Réponses du chapitre 2

1. Le W de WAMP signifie « Windows, puis les autres lettres signifient Apache, MySQL et PHP ». Le M de MAMP désigne le Mac au lieu de Windows et le L de LAMP signifie Linux. Ils désignent tous une solution complète d'hébergement de pages web dynamiques.
2. L'adresse IP 127.0.0.1 et l'URL *http://localhost* sont deux manières différentes de faire référence à l'ordinateur local mais elles reviennent au même. Lorsque le WAMP, le MAMP ou le LAMP est correctement configuré, vous pouvez entrer indifféremment l'un ou l'autre dans la barre d'adresse du navigateur pour accéder à la page par défaut du serveur local.
3. FTP signifie *File transfer protocol*, c'est-à-dire protocole de transfert de fichier. Un programme de transfert FTP, ou simplement programme FTP, est nécessaire pour échanger des fichiers entre un client et un serveur, en particulier lorsque ce dernier se situe à distance.
4. Lorsque vous travaillez à distance sur un serveur web, il est nécessaire d'envoyer les fichiers en FTP vers ce serveur pour les mettre à jour, ce qui ralentit significativement le développement, surtout si ces mises à jour sont fréquentes au cours d'une session de travail.
5. Les éditeurs de programmes spécialisés offrent une intelligence certaine et permettent de mettre en évidence des erreurs dans le code des programmes, avant même de l'exécuter.

Réponses du chapitre 3

1. Les balises qui marquent le début et la fin du code PHP à interpréter sont `<?php ... ?>`, que l'on peut abréger en `<? ... ?>`, mais cette pratique est déconseillée.
2. La marque `//` indique le début d'un commentaire sur une seule ligne, tandis que `/* ... */` indique un commentaire réparti sur plusieurs lignes.
3. Toutes les instructions PHP doivent se terminer par un point-virgule (`;`).
4. À l'exception des constantes, toutes les variables PHP débutent par un `$`.
5. Une variable contient une valeur qui peut être une chaine, un nombre ou toute autre donnée.
6. L'expression `$variable = 1` est une instruction d'affectation, tandis que `$variable == 1` est une comparaison, qui fait appel à l'opérateur de comparaison `==`. Utilisez `$variable = 1` pour définir la valeur de `$variable`. Utilisez `$variable == 1` plus loin dans le programme pour vérifier si `$variable` est égale à `1`. Si, par inadvertance, vous écrivez `$variable = 1` alors que vous vouliez effectuer une comparaison, cela a deux conséquences indésirables : vous imposez la valeur de `$variable` à `1`, et vous renvoyez toujours vrai (la valeur `true`), quelle que soit la valeur précédente de la variable.

7. Le tiret est réservé à l'opérateur de soustraction. Si le tiret était aussi autorisé dans les noms de variables, alors une expression telle que `$utilisateur-courant` serait plus difficile à interpréter et, dans tous les cas, rendrait les programmes ambigus.
8. Les noms de variables sont sensibles à la casse. Ainsi, `$Cette_variable` n'est pas la même que `$cette_variable`.
9. Il n'est pas permis d'utiliser d'espace dans les noms de variables parce que cela induirait l'interpréteur PHP en erreur. À la place, utilisez le caractère de soulignement (`_`).
10. Pour convertir un type de variable en un autre, il suffit d'y faire référence et PHP le convertit automatiquement.
11. Il n'y a pas de différence entre `++$j` et `$j++`, sauf quand il s'agit de les utiliser dans un test, de les affecter à une autre variable, ou de les passer en paramètre d'une fonction. Dans ces cas-là, `++$j` incrémente `$j` avant le test ou les autres opérations, tandis que `$j++` effectue l'opération, puis incrémente `$j`.
12. D'une manière générale, les opérateurs logiques `&&` et `and` sont interchangeables. Ils ne le sont plus lorsque la priorité, la « préséance », importe, auquel cas `&&` a une haute préséance, tandis que `and` en a une faible.
13. Pour utiliser plusieurs lignes dans une chaine, entourez-la de guillemets verticaux ou utilisez la construction avec les balises `<<<_END ... _END;` pour créer un `echo` ou une affectation sur plusieurs lignes. La balise de fermeture `_END;` doit être obligatoirement au début d'une ligne et rien, pas même un commentaire, ne peut figurer sur cette ligne à la suite du point-virgule final.
14. Il est interdit de redéfinir une constante parce que, par définition, elle conserve sa valeur jusqu'à la fin du programme.
15. La barre oblique inverse avant l'apostrophe (`\'`) ou un guillemet (`\"`) font échapper ces caractères à l'analyse par l'interpréteur PHP, ce qui explique qu'on désigne `\` de « caractère d'échappement » et `\'` ou `\"` de « séquence d'échappement ».
16. Les commandes `echo` et `print` sont semblables dans la mesure où ce sont toutes deux des constructions, mais `print` se comporte comme une fonction PHP et ne prend qu'un seul argument, tandis qu'`echo` fait partie intégrante du langage et accepte plusieurs arguments.
17. Le but d'une fonction est de séparer des sections discrètes de code pour définir des sections propres, délimitées, autonomes, qu'il est ensuite possible d'appeler par un seul nom de fonction.
18. Pour rendre une variable accessible à toutes les parties d'un programme PHP, il suffit de la déclarer comme globale, avec le mot-clé `global`.
19. Si une fonction génère des données, il est possible de les passer au reste du programme, soit en les retournant comme résultat de la fonction, soit en modifiant des variables globales.
20. Le résultat de la combinaison d'une chaine et d'un nombre est une autre chaine.

Réponses du chapitre 4

1. En PHP, `TRUE` représente la valeur `1`, tandis que `FALSE` équivaut à `NULL`, que l'on peut se représenter par « rien » et dont le résultat est une chaine vide.
2. Les formes les plus simples d'expression sont les valeurs littérales (telles que les nombres et les chaines) et les variables, qui s'évaluent simplement à elles-mêmes.
3. La différence entre les opérateurs unaires, binaires et ternaire se situe dans le nombre d'opérandes que chacun exige, soit respectivement un, deux et trois.
4. Le meilleur moyen d'imposer un ordre de préséance consiste à placer des parenthèses autour des sous-expressions auxquelles il faut accorder la préséance.
5. L'*associativité d'un opérateur* désigne la direction dans laquelle s'effectue son traitement, soit de gauche à droite, ou de droite à gauche.
6. L'opérateur d'identité (`===` ou son inverse, `!==`) intervient lorsqu'il faut contourner le changement automatique de type de PHP (appelé aussi *conversion implicite de type*) et comparer telles quelles deux valeurs.
7. Les trois types d'instructions conditionnelles sont `if`, `switch` et l'opérateur `?:`.
8. Pour éluder la suite de l'itération courante d'une boucle et passer directement à l'itération suivante, utilisez l'instruction `continue`.
9. La boucle à base d'instruction `for` est plus puissante qu'une boucle `while`, parce qu'elle prend en charge deux paramètres supplémentaires pour contrôler la gestion de la boucle.
10. La plupart des expressions conditionnelles dans les instructions `if` et `while` sont littérales (ou booléennes) et de ce fait, déclenchent l'exécution lorsqu'elles s'évaluent à `TRUE`. Les expressions numériques déclenchent l'exécution quand elles s'évaluent à des nombres différents de zéro. Les expressions de chaine déclenchent l'exécution quand leur évaluation donne une chaine non vide. Une valeur `NULL` est évaluée à `FALSE` et ne déclenche pas l'exécution.

Réponses du chapitre 5

1. L'utilisation des fonctions évite la nécessité de recopier plusieurs fois des portions de code semblables, car elles permettent de regrouper des ensembles d'instructions sous un nom et ensuite d'appeler ce nom dans le reste du code.
2. Par défaut, une fonction ne renvoie qu'une seule valeur. Cependant, le renvoi d'un tableau de valeurs, l'utilisation des références et de variables globales permettent de retourner n'importe quel nombre de valeurs.
3. Lorsque vous référencez une variable par son nom, par exemple en attribuant sa valeur à une autre variable ou en passant sa valeur à une fonction, sa valeur est simplement copiée. L'original ne change pas quand la copie subit une modification. En revanche, lors d'une référence à une variable, seule la référence à (ou un pointeur vers) la variable sert dans l'opération, de sorte qu'une seule valeur est référencée par plusieurs noms. En conséquence, une modification de la valeur de cette référence modifie aussi l'original.

4. La *portée* désigne les parties d'un programme qui peuvent accéder à une variable. Par exemple, les variables de portée globale sont accessibles à partir de n'importe quel endroit dans un programme PHP.
5. Pour insérer un fichier de programme dans un autre, utilisez les directives `include` (inclure) ou `require` (exiger), ou mieux, leurs variantes plus sures, `include_once` ou `require_once`.
6. Une fonction est un ensemble d'instructions désigné par un nom, qui reçoit et renvoie des valeurs. Un objet peut contenir aucune ou plusieurs fonctions (appelées *méthodes*), ainsi que des variables (appelées *propriétés*), le tout rassemblé sous une seule unité.
7. Pour créer un nouvel objet PHP, utilisez le mot-clé `new`, comme suit :

```
$objet = new Classe;
```

8. Pour créer une sous-classe, utilisez le mot-clé `extends` selon la syntaxe suivante :

```
class SousClasse extends ClasseParent ...
```

9. Pour appeler une portion de code d'initialisation lors de la création d'un objet, créez une méthode de constructeur nommée `__construct` dans la classe et placez-y ledit code d'initialisation.
10. La déclaration des propriétés au sein d'une classe n'est pas obligatoire car elles sont automatiquement déclarées de manière implicite lors de leur première utilisation. Cependant, les bonnes pratiques de la programmation considèrent qu'il faut toujours les déclarer de façon explicite pour deux raisons : la lisibilité du code et les nécessités du débogage, en particulier si d'autres personnes sont appelées à intervenir sur votre code.

Réponses du chapitre 6

1. Un tableau numérique peut être indicé de manière numérique à l'aide de nombres ou de variables numériques. Un tableau associatif utilise des identifiants alphanumériques pour identifier les éléments.
2. Le principal avantage du mot-clé `array` réside dans le fait qu'il permet d'affecter plusieurs valeurs en même temps dans un tableau sans répéter le nom du tableau.
3. La fonction `each` et la construction de boucle `foreach...as` renvoient toutes deux des éléments d'un tableau; toutes deux commencent au début et incrémentent un pointeur pour garantir le renvoi de l'élément suivant, et toutes deux retournent `FALSE` quand elles atteignent la fin du tableau. La différence se situe dans le renvoi par `each` d'un seul élément, ce qui explique qu'il faille généralement l'intégrer dans une boucle. La construction `foreach...as` est déjà une boucle, qui s'exécute avec répétition jusqu'à la fin du tableau ou une rupture explicite de cette boucle.
4. Pour créer un tableau multidimensionnel, attribuez des tableaux aux éléments d'un tableau.
5. La fonction `count` permet de dénombrer les éléments d'un tableau.

6. Le but de la fonction `explode` est d'extraire des sections d'une chaine séparées par un identifiant pour, par exemple, extraire des mots séparés par une espace dans une phrase.
7. Pour réinitialiser le pointeur interne de PHP dans un tableau et le ramener au tout premier élément, appelez la fonction `reset`.

Réponses du chapitre 7

1. Le spécificateur de conversion pour afficher un nombre en virgule flottante est `%f`.
2. Pour prendre la chaine `"Joyeux anniversaire"` et sortir la chaine `"**Joyeux"`, l'instruction `printf` s'écrit comme suit :

```
printf("%'*8.6s", "Joyeux anniversaire");
```

3. Il existe la fonction alternative `sprintf` pour affecter le résultat de sortie de `printf` à une variable au lieu de l'afficher dans le navigateur.
4. Pour créer un *timestamp* Unix correspondant au 2 mai 2016 à 7 h 11 du matin, utilisez la commande suivante :

```
$timestamp = mktime(7, 11, 0, 5, 2, 2016);
```

5. Le mode d'accès au fichier `w+` permet, dans `fopen`, d'ouvrir un fichier en écriture et lecture, avec le fichier tronqué et le pointeur de fichier placé au début du fichier.
6. La commande PHP pour supprimer le fichier *fichier.txt* dans le dossier courant est la suivante :

```
unlink('fichier.txt');
```

7. La fonction `file_get_contents` permet de lire en une seule fois la totalité d'un fichier. Elle permet de lire ce fichier même à travers l'internet, si le nom de fichier donné correspond à une adresse URL.
8. Le tableau associatif superglobal `$_FILES` contient tous les détails relatifs aux fichiers téléversés.
9. La fonction PHP `exec` permet l'exécution de commandes système.
10. En HTML5, il est permis d'utiliser soit le style XHTML de balise (comme `<hr />`), soit le style standard de HTML4 (comme `<hr>`). Le choix de l'un ou l'autre est laissé à la discrétion du style de codage conseillé dans votre entreprise ou votre communauté de développeurs.

Réponses du chapitre 8

1. Le point-virgule sert en SQL à séparer ou à terminer des commandes. Si vous oubliez de l'entrer, MySQL affiche une invite et attend que vous entriez la suite de la commande. (Dans le texte des réponses de ce chapitre, il est éludé car il perturbe la fluidité du texte mais il doit obligatoirement terminer toute commande.)
2. Pour connaitre les bases de données disponibles, entrez la commande `SHOW DATABASES`. Pour connaitre les tables disponibles dans une base de données, utilisez la commande `SHOW TABLES`. (Ces commandes ne sont pas sensibles à la casse.)

3. Pour créer un utilisateur, utilisez la commande `GRANT` comme suit :

```
GRANT PRIVILEGES ON nouvellebase.* TO 'nouvelutilisateur'@'localhost'
  IDENTIFIED BY ' nouveaupasse';
```

4. Pour connaitre la structure d'une table, entrez *`DESCRIBE nomtable`*.
5. Le but d'un index MySQL consiste à réduire de manière substantielle les temps d'accès à une base de données par l'entretien d'index sur une ou plusieurs colonnes clés, ce qui permet ensuite de mener des recherches beaucoup plus rapides sur ces colonnes et de repérer des lignes de données dans la table.
6. Un index `FULLTEXT` autorise des requêtes en langage naturel pour trouver des mots-clés, où qu'ils figurent dans les colonnes indexées en `FULLTEXT`, d'une manière comparable à celle des moteurs de recherche sur l'internet.
7. Un *mot vide*, ou *stopword*, est un mot si commun qu'il est considéré comme sans intérêt dans un index `FULLTEXT` et dans les recherches sur cet index. Il participe toutefois à une recherche lorsqu'il fait partie d'une chaine plus longue entourée de guillemets verticaux.
8. Fondamentalement, `SELECT DISTINCT` n'affecte que l'affichage et produit une seule ligne pour ignorer les doublons. `GROUP BY` n'ignore aucune ligne mais combine toutes les lignes de données qui possèdent la même donnée dans la colonne de regroupement. Par conséquent, `GROUP BY` s'avère utile pour effectuer une opération telle que celle de comptage, avec `COUNT`, sur des groupes de lignes. `SELECT DISTINCT` n'a aucun intérêt dans ce cas précis.
9. Pour ne retourner que les lignes qui contiennent *Langhorne* n'importe où dans la colonne *auteur* de la table *classiques*, utilisez une commande telle que la suivante :

```
SELECT * FROM classiques WHERE auteur LIKE "%Langhorne%";
```

10. Pour joindre deux tables, elles doivent posséder en commun au moins une colonne, telle qu'un numéro d'identifiant, *ID* ou, dans le cas des tables *classiques* et *clients*, la colonne *isbn*.

Réponses du chapitre 9

1. Le terme *relation* fait référence à la liaison entre deux données qui affichent une certaine association, comme un livre avec son auteur, ou un livre avec les clients qui l'ont acheté. Un serveur de base de données relationnelle comme MySQL est spécialisé dans le stockage et l'obtention de données selon de telles relations.
2. Le processus de suppression de données redondantes et d'optimisation des tables s'appelle la *normalisation*.
3. Les trois règles de la première forme normale sont les suivantes :
   - Il ne faut aucune répétition de colonnes contenant le même genre de données.
   - Toutes les colonnes ne doivent contenir qu'une seule valeur par colonne.
   - Une clé primaire doit exister pour identifier de manière unique chaque ligne de données.

4. Pour respecter la deuxième forme normale, les colonnes d'une table dont des données se répètent dans plusieurs lignes doivent quitter la table pour se placer dans une table distincte.
5. Dans une relation un-à-plusieurs, la *clé primaire* de la table du côté « un » est reproduite dans une colonne supplémentaire (de *clé étrangère*) de la table du côté « plusieurs ».
6. Pour créer une relation plusieurs-à-plusieurs dans une base de données, il faut créer une table intermédiaire qui contient, en tant que clés étrangères, les clés primaires de deux autres tables. Les autres tables peuvent remonter l'une vers l'autre par l'entremise de cette table intermédiaire.
7. Pour initialiser une transaction dans MySQL, utilisez la commande `BEGIN` ou `START TRANSACTION`. Pour clôturer une transaction et annuler toutes les actions qu'elle a effectuées, utilisez la commande `ROLLBACK`; pour la clôturer et valider toutes les actions, utilisez la commande `COMMIT`.
8. Pour examiner les détails de fonctionnement d'une requête, utilisez la commande `EXPLAIN`.
9. Pour sauvegarder la base de données *publications* dans un fichier nommé *publications.sql*, entrez une commande du genre de la suivante :

```
mysqldump -u utilisateur -pmotdepasse publications > publications.sql
```

Réponses du chapitre 10

1. Pour créer une connexion à une base de données MySQL à l'aide de la `mysqli`, créez une nouvelle instance de la classe `mysqli` et passez-lui en arguments le nom d'hôte, le nom d'utilisateur, le mot de passe et la base de données cible. Lorsqu'elle réussit, la méthode renvoie un objet de connexion.
2. Pour soumettre une requête à MySQL à l'aide de `mysqli`, vérifiez d'abord que vous avez créé un objet de connexion sur une base de données, puis appelez sa méthode `query` et passez-lui la chaine de requête en argument.
3. Lorsqu'une erreur `mysqli` se produit, la propriété `error` de l'objet de connexion contient le message d'erreur associé. Si l'erreur concerne la connexion à la base de données, alors la propriété `connect_error` contient le message d'erreur.
4. Pour déterminer le nombre de lignes renvoyées par une requête `mysqli`, interrogez la propriété `num_rows` de l'objet des résultats renvoyé par la requête.
5. Pour récupérer une ligne déterminée parmi un ensemble de résultats `mysqli`, appelez la méthode `data_seek` de l'objet des résultats et passez-lui en argument le numéro de ligne (à partir de 0); appelez ensuite la méthode `fetch_array` ou toute autre méthode de lecture pour obtenir les données nécessaires.
6. Pour créer des séquences d'échappement dans des chaines, vous pouvez appeler la méthode `real_escape_string` d'un objet de connexion `mysqli`, à laquelle vous passez la chaine à traiter.

7. Si vous négligez la fermeture correcte des objets créés à l'aide des méthodes de `mysqli`, votre programme encourt le risque d'une saturation de la mémoire, en particulier sur des sites web de trafic important. Si vous laissez une erreur de logique dans le flux d'exécution de votre code, la fermeture correcte garantit aussi que vous ne réutilisiez pas accidentellement de vieilles données périmées.

Réponses du chapitre 11

1. Les tableaux associatifs utilisés pour transmettre les données de formulaire à PHP sont `$_GET` pour la méthode Get et `$_POST` pour la méthode Post.
2. Le réglage `register_globals` était activé par défaut dans les versions de PHP antérieures à la 4.2.0. Ce n'était pas souhaitable, parce que cela affectait automatiquement les données des champs d'un formulaire à des variables PHP, ce qui ouvrait une faille de sécurité aux pirates potentiels qui pouvaient ainsi tenter de détourner le code PHP à l'aide de variables initialisées aux valeurs de leur choix.
3. La différence entre une zone de texte et une zone de texte étendu, bien qu'elles acceptent toutes deux du texte, réside dans le fait qu'une zone de texte ne contient qu'une seule ligne, tandis qu'une zone de texte étendu en accepte plusieurs et assure le retour automatique à la ligne suivante.
4. Pour offrir plusieurs choix mutuellement exclusifs dans un formulaire web, utilisez un bouton d'options et non des cases à cocher, qui permettent plusieurs sélections simultanées.
5. Pour soumettre un groupe de sélections dans un formulaire web à l'aide d'un seul nom de champ, utilisez un nom de tableau indiqué par deux crochets, par exemple `choix[]`, au lieu du simple nom de champ habituel. Chaque valeur de champ vient alors se glisser dans une cellule du tableau, dont la longueur est égale au nombre de champs envoyés.
6. Pour envoyer une valeur de champ de formulaire sans que l'utilisateur ne voie le champ, placez-le dans un champ masqué, c'est-à-dire avec l'attribut `type="hidden"`.
7. Il est possible d'encapsuler un élément de formulaire avec une prise en charge d'un texte ou d'un graphisme, pour que la totalité de l'élément soit sélectionnable d'un clic de souris, grâce aux balises `<label>` et `</label>`.
8. Pour convertir du HTML dans un format qui puisse s'afficher sans être interprété comme du code HTML dans un navigateur, utilisez la fonction HTML `htmlentities`.
9. Pour suggérer des informations à l'utilisateur pour remplir un champ, à partir de valeurs qu'il a déjà entrées précédemment, utilisez l'attribut `autocomplete`, qui affiche des valeurs possibles pour ce champ. Cette fonctionnalité s'appelle la *saisie semi-automatique*.
10. Pour empêcher l'envoi d'un formulaire alors que des données y manquent, appliquez l'attribut `required` aux entrées indispensables.

Réponses du chapitre 12

1. Le transfert des cookies doit se faire avant tout code HTML d'une page web, parce qu'ils sont envoyés par l'intermédiaire des entêtes de pages.
2. Pour enregistrer un cookie dans un navigateur web, utilisez la fonction `set_cookie`.
3. Pour détruire un cookie, il faut le réexpédier à l'aide de `set_cookie` mais avec une date d'expiration située assez loin dans le passé.
4. Par l'entremise d'une authentification HTTP, le nom d'utilisateur et le mot de passe sont enregistrés respectivement dans `$_SERVER['PHP_AUTH_USER']` et `$_SERVER['PHP_AUTH_PW']`.
5. La fonction de hachage `hash` représente une puissante mesure de sécurité, parce que c'est une fonction dite à sens unique, capable de convertir une chaine en un nombre hexadécimal de 32 caractères qu'il est impossible de convertir dans l'autre sens, donc qu'il est quasi impossible de décrypter.
6. Le *salage* d'une chaine consiste à ajouter des caractères supplémentaires, connus seulement du programmeur et choisis de manière arbitraire, avant la conversion par `hash`. Ce procédé rend quasi impossible la réussite d'une attaque par dictionnaire ou systématique, par force brute.
7. Une session PHP est un groupe de variables unique et spécifique à l'utilisateur courant. Ces variables sont stockées sur le serveur.
8. Pour initialiser une session, utilisez la fonction `session_start`.
9. Le piratage (ou détournement) de session désigne la situation où un pirate découvre d'une manière ou d'une autre un identifiant de session existant et tente d'en prendre le contrôle.
10. La fixation de session est une autre technique de piratage qui consiste à imposer un identifiant de session à un serveur que celui-ci n'a pas créé lui-même.

Réponses du chapitre 13

1. Pour entourer du code JavaScript, utilisez les balises `<script>` et `</script>`.
2. Par défaut, le code JavaScript sort ses informations dans la partie du document où il réside. S'il est dans l'entête (`<head>`), la sortie a lieu dans cet entête ; s'il est dans le corps (`<body>`), la sortie est générée dans le corps.
3. Pour insérer du code JavaScript d'autres sources, il est possible de le copier-coller mais il est plus pratique de l'inclure à l'aide de la balise `<script src='nomdefichier.js'>`.
4. L'équivalent en JavaScript des commandes `echo` et `print` de PHP est la fonction (ou méthode) `document.write`.
5. Pour créer un commentaire en JavaScript, faites-le précéder de `//` pour un commentaire sur une seule ligne, ou entourez le commentaire sur plusieurs lignes de `/*` et `*/`.

6. L'opérateur de concaténation de chaine en JavaScript est le symbole `+`.
7. Au sein d'une fonction en JavaScript, pour déclarer une variable de portée locale, faites-la précéder du mot-clé `var` à l'emplacement de sa première affectation.
8. Pour afficher l'URL affectée au lien d'identifiant `celien` dans tous les principaux navigateurs, vous pouvez utiliser une des commandes suivantes :

```
document.write(document.getElementById('celien').href)
document.write(celien.href)
```

9. Les commandes qui permettent d'aller à la page précédente dans le tableau de l'historique de navigation sont :

```
history.back()
history.go(-1)
```

10. Pour remplacer le document courant par la page principale du site *goulet.ca*, utilisez la commande suivante :

```
document.location.href = 'http://goulet.ca'
```

Réponses du chapitre 14

1. La principale différence notable entre les valeurs booléennes en PHP et en JavaScript réside dans le fait que PHP reconnait les mots-clés `TRUE`, `true`, `FALSE` et `false`, tandis que JavaScript n'accepte que `true` et `false`. De plus, en PHP, `TRUE` possède la valeur `1` et `FALSE` équivaut à `NULL`. En JavaScript, elles sont représentées par `true` et `false`, qui peuvent être renvoyées sous forme de valeurs chaines.
2. Au contraire de PHP, aucun caractère (comme `$`) ne sert à distinguer les noms des variables. Les noms valables en JavaScript peuvent commencer par (et contenir) des lettres en capitale ou bas de casse, ainsi que des soulignements ; les noms de variables peuvent aussi contenir des chiffres, mais pas pour le premier caractère.
3. La différence entre les opérateurs unaires, binaires et ternaire réside dans le nombre d'opérandes que chacun attend, soit un, deux ou trois, respectivement.
4. La meilleure manière d'imposer une préséance aux opérateurs consiste à entourer de parenthèses une portion d'une expression à évaluer en premier lieu.
5. L'opérateur d'identité sert lorsqu'il s'agit d'empêcher JavaScript d'appliquer sa conversion automatique de type.
6. Les formes les plus simples d'expressions sont les littéraux (comme les nombres et les chaines) et les variables qui s'évaluent simplement à eux-mêmes.
7. Les trois types d'instructions conditionnelles sont `if`, `switch` et l'opérateur `?:`.
8. La plupart des expressions conditionnelles dans les instructions `if` et `while` sont constituées de valeurs littérales ou booléennes et, de ce fait, déclenchent une exécution lorsqu'elles s'évaluent à `true`. Les expressions numériques déclenchent une exécution lorsqu'elles s'évaluent à une valeur différente de `0`. Les expressions de chaines déclenchent l'exécution lorsqu'elles s'évaluent à une chaine non vide. Une valeur `NULL` s'évalue à faux, donc ne déclenche aucune exécution.

9. Les boucles `for` sont plus puissantes que les boucles `while`, parce que l'instruction `for` accepte deux paramètres supplémentaires pour contrôler le traitement de boucle.

10. L'instruction `with` prend un objet en paramètre, qui permet de le spécifier une seule fois et, dans le corps de cette instruction, les propriétés et méthodes qui ne font référence à aucun objet sont automatiquement appliquées à cet objet.

Réponses du chapitre 15

1. En JavaScript, les noms des fonctions et des propriétés sont sensibles à la casse, de sorte que les variables `Compteur`, `compteur` et `COMPTEUR` sont toutes différentes.

2. Pour écrire une fonction qui accepte et traite un nombre illimité de paramètres, accédez à ces paramètres par l'entremise du tableau `arguments`, qui constitue un membre de toutes les fonctions.

3. Un moyen de renvoyer plusieurs valeurs à partir d'une fonction consiste à les ranger dans un tableau et de renvoyer ce tableau.

4. Dans la définition d'une classe, le mot clé `this` permet de faire référence à l'objet courant.

5. La définition des méthodes d'une classe ne doit pas nécessairement se situer au sein de la définition de la classe. Lorsqu'une méthode est définie en dehors du constructeur, il faut en revanche obligatoirement affecter le nom de la méthode à l'objet `this` au sein de la classe.

6. La création d'un objet se fait à l'aide du mot clé `new`.

7. Pour rendre une propriété ou une méthode disponible à tous les objets issus d'une classe sans répliquer cette propriété ou méthode dans tous les objets, utilisez le mot clé `prototype` pour créer une seule instance, qui est ensuite passée automatiquement par référence à tous les objets de cette classe.

8. Pour créer un tableau multidimensionnel, placez des sous-tableaux (de sous-tableaux) dans le tableau principal.

9. La syntaxe à utiliser pour créer un tableau associatif est `clé : valeur`, entre accolades, comme suit :

```
tableauassociatif =
{
  "prénom" : "Jean-Louis",
  "nom"    : "Aubert",
  "groupe" : "Téléphone"
}
```

10. L'instruction pour trier un tableau de nombres en ordre numérique décroissant prend l'allure suivante :

```
nombres.sort(function(a, b){ return b - a })
```


Réponses du chapitre 16

1. Vous pouvez envoyer un formulaire en validation avant de le soumettre par l'ajout de l'attribut `onsubmit` à la balise `<form>`. Assurez-vous que la fonction de validation renvoie `true` si le formulaire peut être envoyé en soumission et `false` dans le cas contraire.

2. Pour comparer une chaine à une expression rationnelle en JavaScript, utilisez la méthode `test`.

3. Parmi les expressions rationnelles capables d'identifier des caractères ne faisant pas partie d'un mot, citons `/[^\w]/`, `/[\W]/`, `/[^a-zA-Z0-9_]/` et ainsi de suite.

4. Une expression rationnelle pour identifier les mots *vivre* ou *vitre* pourrait être `/vi[tv]re/`.

5. Une expression rationnelle pour identifier tout mot simple suivi de n'importe quel caractère non-mot pourrait être `/\w+\W/g`.

6. Une fonction JavaScript pour vérifier à l'aide d'expressions rationnelles si le mot *voix* existe dans la chaine *Voix ambigüe d'un cœur* pourrait s'écrire comme suit :

```
document.write(/voix/i.test("Voix ambigüe d'un cœur"))
```

7. Une fonction PHP pour remplacer à l'aide d'expressions rationnelles toutes les occurrences du mot *la*, quelle qu'en soit la casse, dans *La vache saute sur la lune*, par le mot *ma* peut s'écrire comme suit :

```
$s=preg_replace("/la/i", "ma", "La vache saute sur la lune");
```

8. L'attribut HTML qui permet de préremplir un champ de formulaire avec une valeur est `value`, se place dans une balise `<input>` et adopte l'allure `value="valeur"`.

Réponses du chapitre 17

1. Il est nécessaire d'écrire une fonction de création de nouveaux objets `XMLHttpRequest`, parce que les navigateurs de Microsoft utilisent deux méthodes différentes pour les créer, alors que les autres principaux navigateurs en utilisent une troisième. La rédaction d'une fonction qui teste le navigateur en cours d'utilisation permet de garantir que le code fonctionne dans tous les navigateurs principaux.

2. Le but de la construction `try...catch` est d'intercepter une erreur éventuelle dans le code au sein de l'instruction `try`. Si ce code provoque une erreur, l'exécution de la section `catch` évite de générer une erreur.

3. Un objet `XMLHttpRequest` possède six propriétés et six méthodes (voir tableaux 17-1 et 17-2).

4. Un appel Ajax est achevé lorsque la propriété `readyState` d'un objet prend la valeur `4`.

5. Lorsqu'un appel Ajax s'est achevé avec succès, la propriété `status` de l'objet reçoit la valeur `200`.

6. La propriété `responseText` d'un objet `XMLHttpRequest` contient la valeur renvoyée par un appel Ajax réussi.

7. La propriété `responseXML` d'un objet `XMLHttpRequest` contient une arborescence DOM créée à partir du XML renvoyé par un appel Ajax réussi.
8. Pour spécifier une fonction de rappel pour gérer les réponses Ajax, affectez le nom de cette fonction à la propriété `onreadystatechange` de l'objet `XMLHttpRequest`. Il est possible de définir cette fonction à la volée, sous forme d'une fonction anonyme.
9. Pour amorcer une requête Ajax, appeler la méthode `send` de l'objet `XMLHttpRequest`.
10. Les principales différences entre les requêtes Ajax Get et Post sont que les requêtes Get ajoutent les données à l'URL et non en tant que paramètres de la méthode `send`, tandis que les requêtes Post passent les données en paramètres de la méthode `send` et exigent d'envoyer d'abord les entêtes de formulaire corrects.

Réponses du chapitre 18

1. Pour importer une feuille de style dans une autre, utilisez la directive `@import`, comme suit :

   ```
   @import url('styles.css');
   ```

   Dans un document HTML, cette directive vient se placer dans la section `<style>...</style>` de l'entête du document.
2. Pour importer une feuille de style dans un document, vous pouvez utiliser la balise HTML `<link>` :

   ```
   <link rel='stylesheet' type='text/css' href='styles.css'>
   ```
3. Pour intégrer directement un style dans un élément (par exemple `<div>`), utilisez l'attribut `style`, comme suit :

   ```
   <div style='color:blue;'>
   ```
4. La différence entre un identifiant et une classe CSS réside dans le fait qu'un identifiant ne peut s'appliquer qu'à un seul élément dans un document, tandis qu'une classe peut s'appliquer à plusieurs éléments du même document.
5. Dans une déclaration CSS, les noms d'identifiants sont préfixés du caractère `#` (par exemple `#monid`), tandis que les noms de classes prennent le préfixe `.` (par exemple `.maclasse`).
6. En CSS, le point-virgule sert de séparateur entre les déclarations de styles.
7. Pour ajouter un commentaire, entourez-en le texte des indicateurs `/*` d'ouverture et `*/` de fermeture des commentaires.
8. En CSS, pour faire référence à n'importe quel élément, utilisez le sélecteur universel `*`.
9. En CSS, pour sélectionner un groupe d'éléments ou de types d'éléments différents, placez une virgule entre chaque élément, identifiant ou classe.
10. Lorsque deux règles CSS de même préséance sont présentes, pour accorder à l'une une plus grande priorité par rapport à l'autre, ajoutez la déclaration `!important` à la plus prioritaire, comme suit :

    ```
    p { color:#ff0000 !important; }
    ```


Réponses du chapitre 19

1. Les opérateurs CSS3 `^=`, `$=` et `*=` correspondent respectivement au début, à la fin ou à toute portion d'une chaine.
2. La propriété utilisée pour spécifier la taille d'une image d'arrière-plan est `background-size`, et elle s'emploie comme suit :

   ```
   background-size:800px 600px;
   ```
3. La propriété `border-radius` permet de préciser le rayon d'une bordure, comme suit :

   ```
   border-radius:20px;
   ```
4. Pour répartir un texte long sur plusieurs colonnes, la technique consiste à préciser les propriétés `column-count`, `column-gap` et `column-rule` ou leurs variantes spécifiques aux navigateurs, par exemple :

   ```
   column-count    :3;
      column-gap   :1em;
      column-rule  :1px solid black;
   ```
5. Les quatre fonctions qui permettent de définir des couleurs en CSS sont : `hsl`, `hsla`, `rgb` et `rgba`. Par exemple :

   ```
   color:rgba(0%,60%,40%,0.4);
   ```
6. Pour créer une ombre grise sous du texte, décalée de 5 pixels vers le bas et la droite, avec un flou de 3 pixels, utilisez la déclaration suivante :

   ```
   text-shadow:5px 5px 3px #888;
   ```
7. La déclaration suivante permet d'indiquer qu'un texte tronqué doit être suivi de points de suspension ; utilisez une déclaration du genre de la suivante :

   ```
   text-overflow:ellipsis;
   ```
8. Pour inclure une police web de Google dans une page, sélectionnez-la dans *http://google.com/fonts*. Ensuite, si vous choisissez par exemple la police Lobster, incluez-la dans une balise `<link>` telle que la suivante :

   ```
   <link href='http://fonts.googleapis.com/css?family=Lobster'
         rel='stylesheet' type='text/css'>
   ```

   Puis, n'oubliez pas d'y faire référence dans une déclaration CSS comme suit :

   ```
   h1 { font-family:'Lobster', arial, serif; }
   ```
9. La déclaration CSS utilisée pour faire pivoter un objet de 90 degrés est la suivante :

   ```
   transform:rotate(90deg);
   ```
10. Pour déclarer une transition sur un objet afin que cette modification suive une transition immédiate de manière linéaire au cours d'une semi-seconde lorsque n'importe laquelle des propriétés de l'objet change, utilisez cette déclaration :

    ```
    transition:all .5s linear;
    ```

Réponses du chapitre 20

1. La fonction `O` renvoie un objet en fonction de son identifiant, la fonction `S` renvoie la propriété `style` d'un objet et la fonction `C` retourne un tableau avec tous les objets qui relèvent d'une classe CSS donnée.
2. Pour modifier un attribut CSS d'un objet, utilisez la fonction `setAttribute`, comme suit :

   ```
   monobjet.setAttribute('font-size', '16pt')
   ```

   Il est également possible de modifier un attribut directement (et généralement, moyennant l'utilisation de noms de propriétés légèrement modifiés au besoin), comme suit :

   ```
   monobjet.fontSize = '16pt'
   ```
3. Les propriétés qui fournissent la largeur et la hauteur disponibles dans une fenêtre de navigateur sont `window.innerWidth` et `window.innerHeight`.
4. Pour faire en sorte que quelque chose se produise quand la souris survole puis quitte un objet, il suffit d'associer respectivement les évènements `onmouseover` et `onmouseout` à des fonctions ou des actions.
5. Pour créer un nouvel élément, utilisez un code tel que celui-ci :

   ```
   element = document.createElement('span')
   ```

   Pour ajouter le nouvel élément au DOM, faites appel à un code tel que le suivant :

   ```
   document.body.appendChild(element)
   ```
6. Pour rendre un élément invisible, réglez sa propriété `visibility` à `hidden` (masqué) ou à `visible` pour le réafficher. Pour replier un élément de sorte que ses dimensions deviennent nulles, réglez sa propriété `display` à `none` (la valeur `block` est une façon de le restaurer).
7. Pour définir un seul évènement pour qu'il se produise une seule fois à l'avenir, appelez la fonction `setTimeout`, passez-lui le code ou un nom de fonction à exécuter et le délai d'expiration de l'exécution.
8. Pour définir des évènements répétitifs à des intervalles réguliers, utilisez la fonction `setInterval`, passez-lui le code ou le nom de la fonction à exécuter et le délai de répétition en millisecondes.
9. Pour libérer un élément de son emplacement dans une page web et lui permettre de se déplacer, définissez sa propriété position à `relative`, `absolute` ou `fixed`. Pour lui restituer son emplacement initial, définissez la propriété à `static`.
10. Pour réaliser une animation à un taux de rafraichissement de 50 images par seconde, définissez un intervalle entre les interruptions de 20 millisecondes. Pour calculer cette valeur, il suffit de diviser 1 000 millisecondes par le taux de rafraichissement.

Réponses du chapitre 21

1. Le symbole communément utilisé comme méthode de fabrication pour créer un objet jQuery est le `$`. La méthode alternative s'appelle `jQuery`.
2. Pour établir un lien dans une page web à la version minimisée 1.11.2 de jQuery à partir du RDC de Google, l'instruction JavaScript correspondante est la suivante :

   ```
   <script src='http://ajax.googleapis.com/ajax/libs/jquery/
   1.11.2/jquery.min.js'></script>
   ```
3. La méthode de fabrication jQuery `$` accepte des sélecteurs CSS pour constituer un objet avec les éléments identifiés correspondants.
4. Pour connaitre la valeur d'une propriété CSS, utilisez la méthode `css`, en lui fournissant un nom de propriété. Pour définir la valeur de cette propriété, il faut fournir à la méthode le nom de la propriété et une valeur.
5. Pour associer une méthode à l'évènement `click` d'un élément d'identifiant `elem`, pour le masquer lentement, utilisez une instruction du genre de la suivante :

   ```
   $('#elem').click(function() { $(this).hide('slow') } )
   ```
6. La valeur par défaut étant `static`, pour permettre l'animation d'un élément, il faut affecter à sa propriété `position` une des valeurs suivantes : `fixed`, `relative` ou `absolute`.
7. Pour permettre l'exécution immédiate (ou séquentielle dans le cas d'animations) de méthodes, placez-les en chaine, séparées par des points, comme suit :

   ```
   $('#elem').css('color', 'blue').css('background',
   'yellow').slideUp('slow')
   ```
8. Pour retrouver un objet nœud d'élément à partir d'un objet de sélection jQuery, indicez-le entre crochets, par exemple `$('#elem')[0]`, ou utilisez la méthode `get`, comme suit : `$('#elem').get(0)`.
9. Pour imposer l'affichage en gras de l'élément frère qui précède immédiatement celui d'identifiant `nouvelles`, voici un exemple d'instruction :

   ```
   $('#nouvelles').prev().css('font-weight', 'bold')
   ```
10. En jQuery, pour effectuer une demande Ajax Get, utilisez la méthode `$.get`, comme suit :

    ```
    $.get('http://serveur.com/ajax.php?faire=ceci', function(data) {
    alert('Le serveur dit : ' + data) } )
    ```

Réponses du chapitre 22

1. Le nouvel élément de HTML5 qui permet de dessiner des graphismes directement dans une page web s'appelle le canevas et, pour en créer un, utilisez la balise `<canvas>`.
2. Il est nécessaire d'utiliser JavaScript pour accéder à nombre des nouvelles technologies de HTML5, telles que le canevas et la géolocalisation.

3. Pour intégrer de l'audio et de la vidéo dans une page web, utilisez respectivement les balises `<audio>` et `<video>`.
4. En HTML5, le stockage local permet de bien plus grandes possibilités d'accès à l'espace local de l'utilisateur que les cookies, limités par la quantité de données qu'ils peuvent contenir.
5. HTML5 offre la possibilité de créer des traitements web (*web workers*) pour prendre en charge des tâches secondaires à votre place. Ces traitements sont constitués simplement de code JavaScript

Réponses du chapitre 23

1. Pour créer un élément canevas en HTML, utilisez la balise `<canvas>` et donnez-lui un identifiant que JavaScript puisse utiliser pour y accéder, comme suit :

```
<canvas id='moncanvas'>
```

2. Pour donner à JavaScript accès à un élément canevas, assurez-vous de donner à l'élément un identifiant, tel que `moncanvas`, puis utilisez la fonction `document.getElementdById` (ou la fonction `O` du fichier *OSC.js* du site d'accompagnement du livre) pour renvoyer l'objet de l'élément. Enfin, appelez `getContext` sur l'objet pour obtenir le contexte en deux dimensions du canevas, comme suit :

```
canvas = document.getElementById('moncanvas')
context = canvas.getContext('2d')
```

3. Pour débuter un chemin de canevas, utilisez la méthode `beginPath` sur le contexte. Après la création du chemin, clôturez-le à l'aide de la commande `closePath` sur le contexte, comme suit :

```
context.beginPath()
  // Les commandes de création du chemin viennent ici
context.closePath()
```

4. La méthode *toDataURL* permet d'extraire, de copier des données d'un canevas, que l'on peut affecter à la propriété `src` d'un objet image, comme ceci :

```
image.src = canvas.toDataURL()
```

5. Pour créer un remplissage en dégradé (linéaire ou radial) avec plus de deux couleurs, précisez toutes les couleurs requises en tant que couleurs d'arrêt, affectées à un objet dégradé créé au préalable, et affectez à chacune un point de départ sous la forme d'un pourcentage (compris entre `0` et `1`) du dégradé complet, comme suit :

```
gradient.addColorStop(0,    'green')
gradient.addColorStop(0.3,  'red')
gradient.addColorStop(0,79, 'orange')
gradient.addColorStop(1,    'brown')
```

6. Pour ajuster la largeur des traits dessinés, précisez une valeur (en pixels) pour la propriété `lineWidth` du contexte, comme ceci :

```
context.lineWidth = 5
```


7. Pour préciser une section d'un canevas pour que les dessins à venir n'apparaissent qu'au sein de cette zone, créez un chemin et appelez ensuite la méthode `clip`.
8. Une courbe complexe avec deux points d'attraction imaginaire s'appelle une courbe de Bézier. Pour en créer une, appelez la méthode `bezierCurve` et fournissez-lui deux paires de coordonnées *x* et *y* des points d'attraction, suivies d'une paire de coordonnées du point final de la courbe. La courbe est alors créée de l'emplacement courant du dessin jusqu'au point de destination finale.
9. La méthode `getImageData` renvoie un tableau avec les données des pixels précisés, où à chaque pixel correspondent quatre données successives, qui contiennent les valeurs de rouge, de vert, de bleu et du canal alpha.
10. La méthode `transform` prend six arguments (ou paramètres) qui correspondent, dans l'ordre, à : l'échelle horizontale, l'inclinaison horizontale, l'inclinaison verticale, l'échelle verticale, le déplacement horizontal et le déplacement vertical. Les arguments qui correspondent aux opérations de modification d'échelle sont donc les éléments 1 et 4 de cette liste.

Réponses du chapitre 24

1. Pour insérer de l'audio et de la vidéo dans un document HTML5, utilisez respectivement les balises `<audio>` et `<video>`.
2. Pour garantir une diffusion optimale sur toutes les plateformes de contenu audio, utilisez le codec OGG, plus le codec ACC ou MP3, au choix.
3. Pour lire et mettre en pause la lecture d'un contenu multimédia en HTML5, utilisez les méthodes JavaScript `play` et `pause` associées à des boutons, ceci tant pour l'audio que pour la vidéo.
4. Pour prendre en charge du multimédia dans un navigateur non compatible avec HTML5, vous pouvez intégrer un lecteur audio ou vidéo Flash dans n'importe quel élément audio ou vidéo, qui s'active si la lecture en HTML5 n'est pas possible.
5. Pour garantir une diffusion optimale sur toutes les plateformes de contenu vidéo, utilisez le codec MP4/H.264 et, sur Opera, le codec OGG/Theora ou VP8.

Réponses du chapitre 25

1. Pour obtenir des données de géolocalisation d'un navigateur web, appelez la méthode suivante et passez-lui les noms de deux fonctions rédigées pour gérer l'accès et le refus d'accès aux données :

```
navigator.geolocation.getCurrentPosition(autorise, refuse)
```

2. Pour déterminer si un navigateur prend ou non en charge le stockage local, testez la propriété `typeof` de l'objet `localStorage`, comme suit :

```
if (typeof localStorage == 'undefined')
{
  // Stockage local non disponible
}
```

3. Pour effacer toutes les données de stockage local pour le domaine courant, vous pouvez appeler la méthode `localStorage.clear`.

4. Les traitements web communiquent le plus facilement avec un programme principal à l'aide de la méthode `postMessage` pour envoyer des informations, et l'association à l'évènement `onmessage` de l'objet traitement web pour les récupérer.

5. Pour informer un navigateur web que le document peut s'exécuter hors ligne sous forme d'une application web locale, créez un fichier qui sert de manifeste. Dans ce fichier, énumérez tous les fichiers nécessaires à l'application, puis liez le fichier dans la balise `<html>`, comme suit :

   ```
   <html manifest='nonfichier.appcache'>
   ```

6. Pour empêcher l'action prédéfinie, qui bloque le glisser et déposer, dans les évènements qui gèrent ces opérations, appelez la méthode `preventDefault` de l'objet évènement (`event`) dans les gestionnaires d'évènements `ondragover` et `ondrop`.

7. Pour favoriser la sécurité dans la messagerie interdocuments, fournissez toujours un identifiant de domaine lors de la publication de messages, et vérifiez la présence de cet identifiant lors de leur réception, comme suit, pour la publication :

   ```
   postMessage(message, 'http://mondomaine.com')
   ```

   Et, pour la réception :

   ```
   if (event.origin) != 'http://mondomaine.com') // Interdire
   ```

   Il est préférable, en sus, de chiffrer ou de dissimuler les communications pour éviter l'injection et les regards indiscrets.

8. Le but des microdonnées consiste à améliorer la compréhension des informations par des logiciels, tels que les moteurs de recherche.

ANNEXE B

Ressources en ligne

Cette annexe énumère une série de sites utiles où vous pouvez trouver des cours, des tutoriels, des astuces, des utilitaires pour diverses plateformes ou en ligne, ainsi que la matière évoquée dans ce livre et d'autres ressources pour améliorer vos programmes destinés au web.

Sauf mention de [FR] en début de liens, ces ressources sont en anglais.

Sites de ressources sur PHP

- *http://c walkers.com*
- *http://developer.yahoo.com/php/*
- *http://easyphp.org*
- *http://forums.devshed.com*
- *http://free-php.net*
- *http://hotscripts.com/category/php/*
- *http://htmlgoodies.com/beyond/php/*
- *[FR] http://openclassrooms.com/courses?q=PHP*
- *[FR] http://php.developpez.com*
- *http://php.net*
- *http://php.resourceindex.com*
- *http://php-editors.com*
- *http://phpbuilder.com*
- *http://phpfreaks.com*
- *http://phpunit.de*
- *http://w3schools.com/php/*
- *[FR] http://www.apprendre-php.com/tutoriels.html*
- *[FR] http://www.lephpfacile.com/*

- *[FR] http://www.manuelphp.com/*
- *http://www.zend.com/fr*

Sites de ressources sur MySQL

- *http://dev.mysql.com/doc/refman/5.6/en/locale-support.html*
- *[FR] http://fr.planet.mysql.com ou [EN] http://planet.mysql.com*
- *http://launchpad.net/mysql*
- *[FR] http://mysql.developpez.com*
- *[FR] http://openclassrooms.com/courses?q=MySQL*
- *[FR] http://php.net/manual/fr/mysql.php*
- *http://w3schools.com/PHP/php_mysql_intro.asp*
- *[FR] http://www.mysql.fr*
- *[FR] http://www.oracle.com/fr/products/mysql/overview/index.html*

Sites de ressources sur JavaScript

- *[FR] https://developer.mozilla.org/fr/docs/Web/JavaScript*
- *http://dynamicdrive.com*
- *http://javascript.about.com*
- *[FR] http://javascript.developpez.com*
- *http://javascript.internet.com*
- *http://javascriptkit.com*
- *[FR] http://openclassrooms.com/courses?q=JavaScript*
- *http://w3schools.com/JS*
- *http://webreference.com/js*
- *https://www.codeschool.com/paths/javascript*

Sites de ressources sur CSS

- *http://css-discuss.incutio.com/wiki/Print_Stylesheets*
- *[FR] http://css.developpez.com*
- *http://dustindiaz.com/css-shorthand*
- *http://pleeease.io/play/*
- *[FR] http://openclassrooms.com/courses?q=CSS*

- *http://quirksmode.org/css/quirksmode.html*
- *[FR] http://www.css3create.com/Transformations-3D*
- *http://www.cssbasics.com*
- *https://www.freehtmlvalidator.com*
- *[FR] http://www.html5-css3.fr/css3/transformations-3d-css3*

Sites de ressources sur HTML5

- *http://caniuse.com*
- *http://html5test.com*
- *http://html5readiness.com*
- *http://html5demos.com*
- *http://html5doctor.com*
- *http://html5-demos.appspot.com*
- *http://htmlvalidator.com (démo)*
- *http://modernizr.com*
- *[FR] http://openclassrooms.com/courses?q=HTML5*

Sites de ressources sur Ajax

- *http://ajax.asp.net*
- *[FR] http://ajax.developpez.com/cours/*
- *http://ajaxian.com/by/topic/ajax*
- *http://ajaxmatters.com*
- *[FR] https://developer.mozilla.org/fr/docs/AJAX*
- *http://dojotoolkit.org*
- *http://jquery.com*
- *http://mochikit.com*
- *http://mootools.net*
- *[FR] http://openclassrooms.com/courses?q=Ajax*
- *http://openjs.com*
- *http://prototypejs.org*
- *http://sourceforge.net/projects/clean-ajax*
- *http://w3schools.com/Ajax*

Sites de ressources diverses

- *http://onlinewebcheck.com*
- *http://easyphp.org*
- *[FR] http://eclipse.developpez.com*
- *http://eclipse.org*
- *http://editra.org*
- *[FR] http://www.finalclap.com/faq/227-php-editeur-texte-ide*
- *http://fireftp.mozdev.org*
- *[FR] https://notepad-plus-plus.org/fr/*
- *http://mamp.info/en*
- *[FR] http://php.developpez.com/comparatifs/editeurs/*
- *http://programmingforums.org*
- *http://putty.org*
- *http://sourceforge.net/projects/glossword*
- *[FR] https://www.apachefriends.org/fr/index.html*

Sites de ressources d'O'Reilly

- *http://onlamp.com*
- *http://onlamp.com/php*
- *http://onlamp.com/onlamp/general/mysql.csp*
- *http://oreilly.com/ajax*
- *http://oreilly.com/javascript*
- *http://oreilly.com/mysql*
- *http://oreilly.com/php*
- *http://oreillynet.com/javascript*

ANNEXE C

Mots vides FULLTEXT de MySQL

Cette annexe reprend plus de 500 mots vides (*stopwords*) évoqués au chapitre 8, à la section Créer un index FULLTEXT de la page 191. Les mots vides sont des mots dont l'aspect tellement commun les fait considérer comme sans intérêt dans une recherche et dans le stockage dans un index FULLTEXT. En théorie, le fait d'ignorer ces mots ne représente que peu de différence dans les résultats de la plupart des recherches FULLTEXT, mais cela permet de réduire considérablement les bases de données MySQL et d'en améliorer fortement l'efficacité.

Cette liste est en anglais car c'est la liste par défaut des tables INNODB et il n'en existe pas par défaut en français. Néanmoins, si vous voulez définir votre propre liste de mots vides en français, consultez les deux ressources en ligne suivantes :

- Comment définir votre propre liste de mots vides pour une table INNODB (en anglais), après la liste des mots vides anglais par défaut : *http://dev.mysql.com/doc/refman/5.6/en/fulltext-stopwords.html*.
- La liste des mots vides en français (utilisés dans les moteurs de recherche) : *http://www.ranks.nl/stopwords/french*.

La liste suivante énumère ces mots anglais en bas de casse, mais ils s'appliquent aussi aux versions en capitales et en casse mélangée.

A

a's, able, about, above, according, accordingly, across, actually, after, afterwards, again, against, ain't, all, allow, allows, almost, alone, along, already, also, although, always, am, among, amongst, an, and, another, any, anybody, anyhow, anyone, anything, anyway, anyways, anywhere, apart, appear, appreciate, appropriate, are, aren't, around, as, aside, ask, asking, associated, at, available, away, awfully

B

be, became, because, become, becomes, becoming, been, before, beforehand, behind, being, believe, below, beside, besides, best, better, between, beyond, both, brief, but, by

C

c'mon, c's, came, can, can't, cannot, cant, cause, causes, certain, certainly, changes, clearly, co, com, come, comes, concerning, consequently, consider, considering, contain, containing, contains, corresponding, could, couldn't, course, currently

D

definitely, described, despite, did, didn't, different, do, does, doesn't, doing, don't, done, down, downwards, during

E

each, edu, eg, eight, either, else, elsewhere, enough, entirely, especially, et, etc, even, ever, every, everybody, everyone, everything, everywhere, ex, exactly, example, except

F

far, few, fifth, first, five, followed, following, follows, for, former, formerly, forth, four, from, further, furthermore

G

get, gets, getting, given, gives, go, goes, going, gone, got, gotten, greetings

H

had, hadn't, happens, hardly, has, hasn't, have, haven't, having, he, he's, hello, help, hence, her, here, here's, hereafter, hereby, herein, hereupon, hers, herself, hi, him, himself, his, hither, hopefully, how, howbeit, however

I

i'd, i'll, i'm, i've, ie, if, ignored, immediate, in, inasmuch, inc, indeed, indicate, indicated, indicates, inner, insofar, instead, into, inward, is, isn't, it, it'd, it'll, it's, its, itself

J

just

K

keep, keeps, kept, know, knows, known

L

last, lately, later, latter, latterly, least, less, lest, let, let's, like, liked, likely, little, look, looking, looks, ltd

M

mainly, many, may, maybe, me, mean, meanwhile, merely, might, more, moreover, most, mostly, much, must, my, myself

N

name, namely, nd, near, nearly, necessary, need, needs, neither, never, nevertheless, new, next, nine, no, nobody, non, none, noone, nor, normally, not, nothing, novel, now, nowhere

O

obviously, of, off, often, oh, ok, okay, old, on, once, one, ones, only, onto, or, other, others, otherwise, ought, our, ours, ourselves, out, outside, over, overall, own

P

particular, particularly, per, perhaps, placed, please, plus, possible, presumably, probably, provides

Q

que, quite, qv

R

rather, rd, re, really, reasonably, regarding, regardless, regards, relatively, respectively, right

S

said, same, saw, say, saying, says, second, secondly, see, seeing, seem, seemed, seeming, seems, seen, self, selves, sensible, sent, serious, seriously, seven, several, shall, she, should, shouldn't, since, six, so, some, somebody, somehow, someone, something, sometime, sometimes, somewhat, somewhere, soon, sorry, specified, specify, specifying, still, sub, such, sup, sure

T

t's, take, taken, tell, tends, th, than, thank, thanks, thanx, that, that's, thats, the, their, theirs, them, themselves, then, thence, there, there's, thereafter, thereby, therefore, therein, theres, thereupon, these, they, they'd, they'll, they're, they've, think, third, this, thorough, thoroughly, those, though, three, through, throughout, thru, thus, to, together, too, took, toward, towards, tried, tries, truly, try, trying, twice, two

U

un, under, unfortunately, unless, unlikely, until, unto, up, upon, us, use, used, useful, uses, using, usually

V

value, various, very, via, viz, vs

W

want, wants, was, wasn't, way, we, we'd, we'll, we're, we've, welcome, well, went, were, weren't, what, what's, whatever, when, whence, whenever, where, where's, whereafter, whereas, whereby, wherein, whereupon, wherever, whether, which, while, whither, who, who's, whoever, whole, whom, whose, why, will, willing, wish, with, within, without, won't, wonder, would, would, wouldn't

Y

yes, yet, you, you'd, you'll, you're, you've, your, yours, yourself, yourselves

Z

zero

ANNEXE D

Fonctions de MySQL

Le fait que MySQL propose une pléthore de fonctions réduit fortement la vitesse d'exécution de requêtes compliquées, ainsi que leur complexité. Pour de plus amples informations sur les fonctions disponibles, consultez les adresses URL suivantes :

- Fonctions de chaines de caractères : *tinyurl.com/phpstringfuncs*
- Fonctions de date et heure : *tinyurl.com/phpdateandtime*

Cependant, pour une référence rapide, voici quelques-unes des fonctions de MySQL les plus souvent utilisées.

Fonctions de chaines

`CONCAT(chaine1, chaine2, ...)`

Renvoie le résultat de la concaténation de *chaine1*, *chaine2* et des paramètres suivants (ou `NULL` si n'importe lequel des arguments vaut `NULL`). Si au moins un des arguments est binaire, alors le résultat est une chaine binaire; sinon le résultat est une chaine non binaire. Le code suivant renvoie la chaine `"MySQL"` :

```
SELECT CONCAT('My', 'S', 'QL');
```

`CONCAT_WS(séparateur, chaine1, chaine2, ...)`

Celle-ci fonctionne de la même manière que `CONCAT`, sauf qu'elle prend un *séparateur* à insérer entre les éléments à concaténer. Si le séparateur est `NULL`, le résultat est `NULL`, mais des valeurs `NULL` peuvent apparaitre dans les autres arguments, qui seront alors ignorés. Le code suivant retourne la chaine `"Jagger,Mick"` :

```
SELECT CONCAT_WS(',', 'Jagger', 'Mick');
```

`LEFT(chaine, len)`

Revoie les *len* caractères les plus à gauche de la chaine *chaine* (ou `NULL` si un des arguments vaut `NULL`). Le code suivant retourne la chaine `"Chris"` :

```
SELECT LEFT('Christophe Colomb', '5');
```

`RIGHT(chaine, len)`

Revoie les *len* caractères les plus à droite de la chaine *chaine* (ou `NULL` si un des arguments vaut `NULL`). Le code suivant retourne la chaine `"Colomb"` :

```
SELECT RIGHT('Christophe Colomb', '6');
```

MID(*chaine*, *pos*, *len*)
: Renvoie jusque *len* caractères de la chaine *chaine*, à partir de l'emplacement *pos*. Si vous omettez la longueur *len*, alors la fonction renvoie tous les caractères depuis l'emplacement *pos* jusqu'à la fin de la chaine. Vous pouvez utiliser une valeur négative pour *pos*, auquel cas cela représente le caractère placé à *pos* à partir de la fin de la chaine. Le premier emplacement de la chaine est en 1. Ainsi, le code suivant renvoie la chaine "stop":

```
SELECT MID('Christophe Colomb', '5', '4');
```

LENGTH(*chaine*)
: Retourne la longueur en octets de la chaine *chaine*. Remarquez que les caractères sur plusieurs octets comptent aussi pour plusieurs octets. Pour connaître la longueur réelle en caractères d'une chaine, utilisez plutôt la fonction CHAR_LENGTH. Cette instruction renvoie la valeur 15:

```
SELECT LENGTH('Mark Zuckerberg');
```

LPAD(*chaine*, *len*, *chainecal*)
: Renvoie la chaine *chaine* calibrée (*padded*) à la longueur de *len* caractères, quitte à ajouter par la gauche autant de fois le caractère *chainecal* que nécessaire. Si *chaine* est plus longue que la longueur *len* indiquée, alors la fonction tronque la chaine *chaine* par la gauche pour atteindre la longueur *len*. Les lignes de code suivantes revoient ces chaines:

```
Janvier
Février
   Mars
  Avril
    Mai
```

Remarquez l'ajout d'espaces à gauche pour que chaque chaine atteigne 8 caractères de long:

```
SELECT LPAD('Janvier', '8', ' ');
SELECT LPAD('Février', '8', ' ');
SELECT LPAD('Mars', '8', ' ');
SELECT LPAD('Avril', '8', ' ');
SELECT LPAD('Mai', '8', ' ');
```

RPAD(*chaine*, *len*, *chainecal*)
: Cette fonction opère selon le même principe que LPAD mais les caractères de calibrage *chainecal* se placent à droite de la chaine *chaine*. L'instruction suivante renvoie la chaine "Bonjour!!!":

```
SELECT RPAD('Bonjour', '10', '!');
```

LOCATE(*souschaine*, *chaine*, *pos*)
: Renvoie l'emplacement de la première occurrence de la sous-chaine *souschaine* dans la chaine *chaine*. Si le paramètre *pos* est précisé, la recherche débute à l'emplacement *pos* (à partir de 1). Si la fonction ne trouve pas *souschaine* dans *chaine*, elle renvoie la valeur 0. Les lignes de code suivantes renvoient respectivement les

valeurs 6 et 17, parce que le premier appel de la fonction renvoie l'emplacement de la première occurrence du mot *bec*, tandis que le second appel commence à chercher à partir du septième caractère ('e'), donc il renvoie l'emplacement de la deuxième occurrence:

```
SELECT LOCATE('bec', 'Bons becs du Québec');
SELECT LOCATE('bec', 'Bons becs du Québec', 7);
```

LOWER(*chaine*)
: Fonction inverse de UPPER, LOWER renvoie le contenu de la chaine *chaine* avec tous ses caractères convertis en bas de casse. Ainsi, l'instruction suivante renvoie la chaine "reine elizabeth ii":

```
SELECT LOWER('Reine Elizabeth II');
```

UPPER(*chaine*)
: Fonction inverse de LOWER, UPPER renvoie le contenu de la chaine *chaine* avec tous ses caractères convertis en capitales. Ainsi, l'instruction suivante renvoie la chaine "POURQUOI DONC CRIER?":

```
SELECT UPPER("pourquoi donc crier?");
```

QUOTE(*chaine*)
: Renvoie une chaine entourée d'apostrophes, correctement utilisable comme valeur de chaine dans une instruction SQL. La chaine renvoyée est entourée d'apostrophes, avec les apostrophes, les barres obliques inverses, les caractères ASCII NUL et Control-Z précédés d'une barre oblique inverse. Si l'argument *chaine* vaut NULL, alors la valeur renvoyée devient NULL sans guillemets ni apostrophes. L'instruction suivante donne ce résultat:

```
'J\'ai faim'
```

Remarquez le remplacement de l'apostrophe dans le texte par la séquence d'échappement \'.

```
SELECT QUOTE("J'ai faim");
```

REPEAT(*chaine*, *nombre*)
: Renvoie une chaine formée de *nombre* copies de la chaine *chaine*. Si nombre vaut moins de 1, la fonction renvoie une chaine vide. Si l'un des paramètres vaut NULL, alors elle renvoie NULL. L'instruction suivante renvoie les chaines "Ho Ho Ho " et "Joyeux Noël":

```
SELECT REPEAT('Ho ', 3), 'Joyeux Noël';
```

REPLACE(*chaine*, *ceci*, *par_ça*)
: Renvoie la *chaine* où toutes les occurrences de la chaine *ceci* sont remplacées par la chaine *par_ça*. La recherche et le remplacement sont sensibles à la casse de *ceci*. L'instruction suivante renvoie la chaine "Cheeseburger et soda":

```
SELECT REPLACE('Cheeseburger et frites', 'frites', 'soda');
```

`TRIM([spécificateur car_à_retirer FROM] chaine)`

Renvoie la *chaine* avec tous les préfixes ou les suffixes éliminés. Le *spécificateur* prend une des valeurs `BOTH` (les deux), `LEADING` (préfixe) ou `TRAILING` (suffixe). Si vous ne précisez pas le *spécificateur*, la valeur `BOTH` est considérée par défaut. La chaine à éliminer `car_à_retirer` est facultative et si elle est omise, alors le caractère espace est considéré par défaut. Le code suivant renvoie les chaines `"Sans rognage"` et `"Bonjour__"` :

```
SELECT TRIM('   Sans rognage   ');
SELECT TRIM(LEADING '_' FROM '__Bonjour__');
```

`LTRIM(chaine)` et `RTRIM(chaine)`

La fonction `LTRIM` renvoie la chaine dont tous les espaces de gauche ont été éliminés, tandis que `RTRIM` effectue la suppression des espaces à droite. Les instructions suivantes renvoient respectivement `"Rognage    "` et `"    Rognage"` :

```
SELECT LTRIM('   Rognage   ');
SELECT RTRIM('   Rognage   ');
```

Paramètres régionaux en français

MySQL gère aussi des paramètres régionaux dans les dates et heures. Ainsi, au chapitre 8, nous avons vu que les fonctions `DATE` et `TIME` fournissent par défaut des formats de date et heure anglo-saxons. Les noms des jours de semaine ou de mois en anglais ne sont pas toujours très pratiques, puisqu'ils nécessitent un peu de code pour les traduire. Or MySQL possède un paramètre, `lc_time_names`, qui simplifie considérablement cette gestion. Si, dans une console de MySQL (pour y accéder, voir le chapitre 8), vous entrez la commande suivante :

```
SELECT @@lc_time_names;
```

… vous obtenez le résultat suivant, soit la configuration `en_US` :

```
+-------------------+
| @@lc_time_names |
+-------------------+
| en_US           |
+-------------------+
1 row in set (0.00 sec)
```

Dès lors, pour connaitre le nom du jour, vous pouvez entrer la commande suivante, qui affiche par exemple `Tuesday` :

```
SELECT DAYNAME(CURDATE());
```

Pour basculer en français, vous allez devoir imposer un format régional qui dépend de votre pays de consultation. La liste complète de ces formats régionaux et tous les détails pour les mettre en œuvre sont disponibles sur *dev.mysql.com/doc/refman/5.6/en/locale-support.html*. Pour simplifier, retenez que la valeur du paramètre régional francophone est `fr_BE` pour la Belgique, `fr_CA` pour le Canada, `fr_CH` pour la Suisse romande, `fr_FR` pour la France et même `fr_LU` pour le Luxembourg. Ensuite, il suffit d'entrer la commande suivante, par exemple pour le Canada francophone :

```
SET lc_time_names = 'fr_CA';
```

Et, si vous entrez à nouveau la commande suivante, vous obtenez le nom du jour de la semaine en français, `mardi` dans notre exemple) :

```
SELECT DAYNAME(CURDATE());
```

Et voilà ! Dans la liste de fonctions suivantes, nous conservons les paramètres de langue par défaut (jours de semaine, noms de mois, etc.) mais vous savez désormais comment obtenir et modifier les détails en français.

Fonctions de dates

Les dates revêtent une importance capitale pour la plupart des bases de données. Chaque fois que des transactions financières ont lieu, la date de transaction doit être enregistrée, de même que les dates d'expiration des cartes bancaires lorsqu'il s'agit de répéter plusieurs prélèvements par leur entremise, suite à des facturations. C'est une des raisons qui expliquent que MySQL propose une vaste gamme de fonctions pour gérer plus aisément et efficacement les dates.

`CURDATE()`

Renvoie la date courante au format AAAA-MM-JJ ou AAAAMMJJ, selon le contexte de chaine ou numérique d'utilisation de la fonction. À la date du 18 mai 2018, le code suivant renvoie les valeurs `2018-05-18` et `20180518` :

```
SELECT CURDATE();
SELECT CURDATE() + 0;
```

`DATE(expr)`

Extrait la partie date de la date ou d'une expression *expr* de type `DATETIME`. L'instruction suivante renvoie la valeur `1961-05-02` :

```
SELECT DATE('1961-05-02 14:56:23');
```

`DATE_ADD(date, INTERVAL expr unité)`

Renvoie le résultat de l'addition de la *date* et de l'expression *expr* exprimée selon une *unité*. L'argument *date* est la date de début ou une valeur `DATETIME` ; *expr* peut recevoir le symbole – pour des intervalles négatifs. Le tableau D-1 énumère les unités d'intervalles prises en charge et les valeurs *expr* attendues. Remarquez en passant que les exemples du tableau citent, là où elles sont indispensables, les apostrophes qui entourent les valeurs d'*expr*, pour que MySQL puisse les interpréter correctement. Dans le doute, n'hésitez pas à les ajouter systématiquement car cela fonctionnera toujours.

Tableau D-1. Valeurs attendues pour expr

Unité	Valeurs attendue d'expr	Exemple
MICROSECOND	MICROSECONDES	111111
SECOND	SECONDES	11
MINUTE	MINUTES	11
HOUR	HEURES	11
DAYS	JOURS	11
WEEK	SEMAINES	11
MONTH	MOIS	11
QUARTER	TRIMESTRES	1
YEAR	ANNÉES	11
SECOND_MICROSECOND	'SECONDS.MICROSECONDES'	11.22
MINUTE_MICROSECOND	'MINUTES.MICROSECONDES'	11.22
MINUTE_SECOND	'MINUTES:SECONDES'	'11:22'
HOUR_MICROSECOND	'HEURES.MICROSECONDES'	11.22
HOUR_SECOND	'HEURES:MINUTES:SECONDES'	'11:22:33'
HOUR_MINUTE	'HEURES:MINUTES'	'11:22'
DAY_MICROSECOND	'JOURS.MICROSECONDES'	11.22
DAY_SECOND	'JOURS HEURES:MINUTES:SECONDES'	'11 22:33:44'
DAY_MINUTE	'JOURS HEURES:MINUTES'	'11 22:33'
DAY_HOUR	'JOURS HEURES'	'11 22'
YEAR_MONTH	'ANNÉES-MOIS'	'11-2'

La fonction DATE_SUB permet aussi de soustraire des intervalles de dates. Il n'est toutefois pas nécessaire de faire appel aux fonctions DATE_ADD et DATE_SUB car vous pouvez utiliser directement de l'arithmétique de dates en MySQL :

```
SELECT DATE_ADD('1975-01-01', INTERVAL 77 DAY);
SELECT DATE_SUB('1982-07-04', INTERVAL '3-11' YEAR_MONTH);
SELECT '2018-12-31 23:59:59' + INTERVAL 1 SECOND;
SELECT '2000-01-01' - INTERVAL 1 SECOND;
```

Ce code renvoie les valeurs suivantes :

```
1975-03-19
1978-08-04
2019-01-01 00:00:00
1999-12-31 23:59:59
```

Remarquez que les deux dernières instructions utilisent de l'arithmétique directe de dates, sans recourir aux fonctions.

DATE_FORMAT(*date*, *format*)
: Renvoie une valeur de date mise en forme en fonction de la chaine de *format*. Le tableau D-2 énumère les spécificateurs utilisables dans la chaine *format*. Retenez l'obligation d'utiliser le caractère % en préfixe de chacun de ces spécificateurs.

Le code suivant renvoie une date et heure donnée du vendredi 4 mai 2018, à 03:02 (soit Friday May 4th 2018 03:02 AM, puisque c'est en anglais) :

```
SELECT DATE_FORMAT('2018-05-04 03:02:01', '%W %M %D %Y %h:%i %p');
```

Tableau D-2. Spécificateurs de format de DATE_FORMAT

Spécificateur	Description
%a	Nom abrégé du jour de semaine (Sun–Sat)
%b	Nom abrégé du mois (Jan–Dec)
%c	Mois en numérique (0–12)
%D	Jour du mois avec le suffixe anglais (0th, 1st, 2nd, 3rd, ...)
%d	Jour du mois, en numérique (00–31)
%e	Jour du mois, en numérique (0–31)
%f	Microsecondes (000000–999999)
%H	Heure (00–23)
%h	Heure (01–12)
%I	Heure (01–12)
%i	Minutes en numérique (00–59)
%j	Jour de l'année (001–366)
%k	Heure (0–23)
%l	Heure (1–12)
%M	Nom du mois (January–December)
%m	Mois en numérique (00–12)
%p	AM ou PM
%r	Heure en 12 heures (hh:mm:ss suivi d'AM ou de PM)
%S	Secondes (00–59)
%s	Secondes (00–59)
%T	Heure en 24 heures (hh:mm:ss)
%U	Semaine (00–53), où le dimanche est le premier jour de la semaine
%u	Semaine (00–53), où le lundi est le premier jour de la semaine
%V	Semaine (01–53), où le dimanche est le premier jour de la semaine; utilisé avec %X
%v	Semaine (01–53), où le lundi est le premier jour de la semaine; utilisé avec %X
%W	Jour de la semaine en numérique (0 = dimanche; 6 = samedi)
%X	Année où le dimanche est le premier jour de la semaine, en numérique, à quatre chiffres; utilisé avec %V
%x	Année où le lundi est le premier jour de la semaine, en numérique, à quatre chiffres; utilisé avec %v
%Y	Année en numérique, à quatre chiffres
%y	Année en numérique, à deux chiffres
%%	Le caractère littéral %

DAY(*date*)
: Renvoie le jour du mois de la *date* précisée, dans la plage de 1 à 31, ou 0 pour les dates qui possèdent un jour égal à zéro, comme dans 0000-00-00 ou 2018-00-00. La fonction DAYOFMONTH produit exactement le même résultat. L'instruction suivante renvoie la valeur 3 :

```
SELECT DAY('2018-02-03');
```

DAYNAME(*date*)
: Renvoie le nom du jour de la semaine de la *date* précisée. Cette instruction renvoie la chaine "Saturday" :

```
SELECT DAYNAME('2018-02-03');
```

DAYOFWEEK(*date*)
: Renvoie l'indice du jour de semaine de la *date*, avec un nombre compris entre 1 pour le dimanche et 7 pour le samedi. Le code suivant renvoie la valeur 7 :

```
SELECT DAYOFWEEK('2018-02-03');
```

DAYOFYEAR(*date*)
: Retourne le jour de l'année de la date donnée dans la plage de 1 à 366. L'instruction suivante renvoie 34 :

```
SELECT DAYOFYEAR('2018-02-03');
```

LAST_DAY(*date*)
: Renvoie le dernier jour du mois correspondant à la *date* de type DATETIME. Si l'argument s'avère incorrect, la fonction retourne NULL. Le code suivant,

```
SELECT LAST_DAY('2018-02-03');
SELECT LAST_DAY('2018-03-11');
SELECT LAST_DAY('2018-04-26');
```

renvoie les valeurs qui suivent :

```
2018-02-28
2018-03-31
2018-04-30
```

Comme vous pouvez vous y attendre, il renvoie le 28 février, le 31 mars et le 30 avril.

MAKEDATE(*année, jour_de_l_année*)
: Renvoie une date correspondant aux valeurs de l'*année* et du *jour_de_l_année*. Si le *jour_de_l_année* vaut 0, le résultat est NULL. L'instruction suivante retourne la date 2016-10-01 :

```
SELECT MAKEDATE(2018,274);
```

MONTH(*date*)
: Renvoie le mois correspondant à la *date* donnée dans la plage de 1 à 12, pour janvier à décembre. Une *date* avec la partie mois égale à zéro, comme 0000-00-00 ou 2016-00-00, donne 0 comme résultat. L'instruction suivante renvoie la valeur 7 :

```
SELECT MONTH('2018-07-11');
```


MONTHNAME(*date*)
: Renvoie le nom complet du mois de la *date*. La commande suivante renvoie la chaine *"July"* :

```
SELECT MONTHNAME('2018-07-11');
```

SYSDATE()
: Renvoie la date et l'heure courantes au format AAAA-MM-JJ HH:MM:SS ou AAAMMJJHHMMSS, selon le contexte de chaine ou numérique d'utilisation de la fonction. La fonction NOW fonctionne de façon semblable, sauf qu'elle renvoie la date et l'heure du début de l'instruction courante, tandis que SYSDATE renvoie la date et l'heure au moment exact de l'appel de la fonction elle-même. Le 19 décembre 2018, à 19:11:13, le code suivant renvoie les valeurs 2018-12-19 19:11:13 et 20181219191113 :

```
SELECT SYSDATE();
SELECT SYSDATE() + 0;
```

YEAR(*date*)
: Renvoie l'année de la *date* donnée, dans la plage de 1000 à 9999, ou 0 pour la date zéro. Cette instruction renvoie l'année 1999 :

```
SELECT YEAR('1999-08-07');
```

WEEK(*date [, mode]*)
: Renvoie le numéro de la semaine de l'année correspondant à la *date* donnée. Le paramètre facultatif *mode* corrige le numéro de semaine selon les nuances du tableau D-3. La fonction WEEKOFYEAR équivaut à WEEK avec le *mode* égal à 3. L'instruction suivante renvoie le numéro de semaine 14 :

```
SELECT WEEK('2018-04-04', 1);
```

Tableau D-3. Les modes acceptés par la fonction WEEK

Mode	Premier jour de la semaine	Plage	La semaine 1 est la première semaine avec...
0	Dimanche	0-53	un dimanche dans cette année
1	Lundi	0-53	plus de trois jours dans cette année
2	Dimanche	1-53	un dimanche dans cette année
3	Lundi	1-53	plus de trois jours dans cette année
4	Dimanche	0-53	plus de trois jours dans cette année
5	Lundi	0-53	un lundi dans cette année
6	Dimanche	1-53	plus de trois jours dans cette année
7	Lundi	1-53	un lundi dans cette année

WEEKDAY(*date*)
: Renvoie l'indice de jour de la semaine pour la *date* donnée, où 0 correspond au lundi et 6 au dimanche. L'instruction suivante renvoie la valeur 2 :

```
SELECT WEEKDAY('2018-04-04');
```

Fonctions d'heures

Il est souvent nécessaire de faire abstraction des dates pour privilégier l'heure. Dans ce contexte aussi, MySQL propose une pléthore de fonctions pour s'adapter aux cas d'utilisation pratiques.

`CURTIME()`

Renvoie l'heure courante sous la forme HH:MM:SS ou HHMMSS.uuuuuu, selon le contexte de chaine ou numérique d'utilisation de la fonction. La valeur retournée dépend du fuseau horaire courant. Lorsque l'heure courante est `11:56:23`, le code suivant renvoie les valeurs `11:56:23` et `115623.000000` :

```
SELECT CURTIME();
SELECT CURTIME() + 0;
```

`HOUR(temps)`

Renvoie l'heure correspondant au *temps* donné. L'instruction suivante produit la valeur `11` :

```
SELECT HOUR('11:56:23');
```

`MINUTE(temps)`

Renvoie les minutes correspondant au *temps* donné. L'instruction suivante renvoie la valeur `56` :

```
SELECT MINUTE('11:56:23');
```

`SECOND(temps)`

Renvoie les secondes correspondant au *temps* donné. L'instruction suivante renvoie la valeur `23` :

```
SELECT SECOND('11:56:23');
```

`MAKETIME(heures, minutes, secondes)`

Renvoie une valeur de temps construite à partir des arguments *heures*, *minutes* et *secondes*. L'instruction suivante génère un temps de `11:56:23` :

```
SELECT MAKETIME(11, 56, 23);
```

`TIMEDIFF(expr1, expr2)`

Renvoie la différence entre *expr1* et *expr2* (soit *expr1 - expr2*) sous forme d'une valeur de temps. Les *expr1* et *expr2* doivent être des expressions de type `TIME` ou `DATETIME` identiques. L'instruction suivante renvoie la valeur `01:37:38` :

```
SELECT TIMEDIFF('2000-01-01 01:02:03', '1999-12-31 23:24:25');
```

`UNIX_TIMESTAMP([date])`

Lors d'un appel sans l'argument facultatif *date*, cette fonction renvoie le nombre de secondes écoulées depuis 1970-01-01 00:00:00 UTC sous forme d'un entier non signé. Si l'argument *date* est précisé, alors la fonction renvoie le nombre de secondes écoulées entre la date de début en 1970, jusqu'à la *date* donnée. Cette commande ne produit pas la même valeur pour tout le monde car la date donnée est interprétée comme une date-heure locale (en fonction du fuseau horaire de l'utilisateur). Le code suivant renvoie la valeur `946684800` (soit le nombre de secondes écoulées jusqu'au début du nouveau millénaire), suivie d'un `TIMESTAMP` (horodatage) qui représente le temps Unix courant au moment de l'exécution :

```
SELECT UNIX_TIMESTAMP('2000-01-01');
SELECT UNIX_TIMESTAMP();
```

`FROM_UNIXTIME(unix_timestamp [, format])`

Renvoie le paramètre *unix_timestamp* sous forme d'une chaine au format AAAA-MM-JJ HH:MM:SS ou d'un nombre au format AAAAMMJJHHMMSS.uuuuuu, selon que la fonction est appelée dans un contexte de chaine ou numérique. Lorsque vous précisez le paramètre facultatif format, le résultat est présenté en fonction des spécificateurs énumérés au tableau D-2. La valeur précise renvoyée par la fonction dépend du fuseau horaire de l'utilisateur. Le code suivant renvoie les chaines `"2000-01-01 00:00:00"` et `"Saturday January 1st 2000 12:00 AM"` :

```
SELECT FROM_UNIXTIME(946684800);
SELECT FROM_UNIXTIME(946684800  M %D %Y %h:%i %p');
```

La section Paramètres régionaux en français évoquait plus haut l'impact de la définition du paramètre `lc_time_names` sur les noms de jours et de mois. Si vous avez défini ce paramètre à `'fr_FR'`, par exemple, la dernière ligne de code précédente donnerait la chaine `"samedi janvier 1st 2000 01:00 AM"`. Pour obtenir une formulation plus propre en français, utilisez plutôt l'instruction suivante, qui produit la chaine `"Le samedi 1 janvier 2000 à 01:00"` :

```
SELECT FROM_UNIXTIME(946684800, 'Le %W %e %M %Y \à %H:%i');
```

ANNEXE E

Sélecteurs, objets et méthodes de jQuery

Le chapitre 21 vous apporte d'excellentes bases de l'utilisation de la bibliothèque jQuery écrite en JavaScript. Pour vous aider à aller plus loin dans l'utilisation de jQuery et obtenir les meilleurs effets, voici une liste exhaustive des sélecteurs, objets et méthodes qu'elle emprunte, dont ce livre n'a pu présenter nombre parmi eux, faute de place. Vous êtes cependant prêt à les aborder parce que vous en savez suffisamment pour les exploiter correctement.

Conservez toutefois à l'esprit que jQuery est un produit vivant qui ne cesse de connaitre des ajouts de fonctionnalités, de corrections de bogues et d'abandons de fonctionnalités obsolètes. Comme ces détails ne peuvent être abordés ici, pour vous tenir au courant des derniers développements, des informations sur les fonctionnalités dépréciées ou abandonnées et des nouvelles versions de jQuery, consultez *jquery.com* et *api.jquery.com*.

Sélecteurs de jQuery

`('*')`
: Sélectionne tous les éléments.

`('element')`
: Sélectionne tous les éléments de nom de balise `element` donné.

`('#id')`
: Sélectionne un seul élément d'attribut `id` donné.

`('.classe')`
: Sélectionne tous les éléments de classe donnée.

`('selecteur1, selecteur2, selecteurN')`
: Sélectionne les résultats de la combinaison de tous les sélecteurs précisés.

`('ancetre descendant')`
: Sélectionne tous les éléments descendants d'un ancêtre donné.

('*precedent + suivant*')
Sélectionne tous les éléments *suivants* identifiés par suivant immédiatement précédés par un frère *precedent*.

('*precedent ~ freres*')
Sélectionne tous les éléments frères qui suivent l'élément *precedent*, qui possèdent le même parent et sont identifiés par le sélecteur de filtrage *freres*.

('*parent > enfant*')
Sélectionne tous les éléments enfants directs identifiés par *enfant*, des éléments identifiés par parent.

[*nom*]
Sélectionne tous les éléments qui possèdent l'attribut de *nom* précisé, quelle qu'en soit la valeur.

[*nom*|='*valeur*']
Sélectionne les éléments qui possèdent le nom d'attribut *nom* donné avec une valeur, soit égale à la chaine *valeur* donnée, soit qui débute par cette chaine suivie d'un tiret (-).

[*nom**='*valeur*']
Sélectionne les éléments qui possèdent le nom d'attribut *nom* donné avec une valeur contenant la sous-chaine *valeur* donnée.

[*nom*~='*valeur*']
Sélectionne les éléments qui possèdent le nom d'attribut *nom* donné avec une valeur contenant le mot *valeur* délimité par des espaces.

[*nom*$='*valeur*']
Sélectionne les éléments qui possèdent le nom d'attribut *nom* donné avec une valeur qui se termine exactement par la chaine *valeur* donnée. La comparaison est sensible à la casse.

[*nom*='*valeur*']
Sélectionne les éléments qui possèdent le nom d'attribut *nom* donné dont la valeur est exactement égale à une certaine *valeur*.

[*nom*!='*valeur*']
Sélectionne les éléments qui ne possèdent pas l'attribut de *nom* donné ou qui ont cet attribut mais pas avec une certaine *valeur*.

[*nom*^='*valeur*']
Sélectionne les éléments qui possèdent le nom d'attribut *nom* donné avec une valeur qui commence exactement avec la chaine *valeur*.

[*nom*='*valeur*'][*nom2*='*valeur2*']
Identifie tous les éléments qui correspondent simultanément à tous les filtres d'attributs précisés.

:animated
Sélectionne tous les éléments qui participent à une animation au moment de l'exécution du sélecteur.

:button
Sélectionne tous les boutons et les éléments de type button.

:checkbox
Sélectionne tous les éléments de type case à cocher (checkbox).

:checked
Identifie tous les éléments cochés (checked) et sélectionnés (selected).

:contains(*texte*)
Sélectionne tous les éléments qui contiennent le *texte* indiqué.

:disabled
Sélectionne tous les éléments désactivés (disabled).

:empty
Sélectionne tous les éléments qui ne possèdent aucun enfant (dont les nœuds de texte).

:enabled
Sélectionne tous les éléments activés (enabled).

:eq(*n*)
Sélectionne l'élément à l'indice *n* au sein de l'ensemble identifié.

:even
Sélectionne les éléments pairs à partir de l'indice zéro. Voir aussi odd.

:file
Sélectionne tous les éléments qui correspondent à des fichiers (file).

:first-child
Sélectionne tous les éléments qui sont le premier enfant de leur parent.

:first-of-type
Sélectionne tous les éléments, premiers parmi les frères du même nom d'élément.

:first
Sélectionne le premier parmi les éléments identifiés.

:gt(*indice*)
Sélectionne tous les éléments d'indice supérieur à l'*indice* donné dans l'ensemble identifié.

:has(*selecteur*)
Sélectionne les éléments qui contiennent au moins un élément correspondant au *sélecteur* précisé.

`:image`
Sélectionne tous les éléments de type `image`.

`:header`
Sélectionne tous les éléments de titre, comme `h1`, `h2`, `h3` et ainsi de suite.

`:hidden`
Sélectionne tous les éléments cachés (`hidden`).

`:input`
Sélectionne tous les éléments `input`, `textarea`, `select` et `button`.

`:lang(`*`langue`*`)`
Sélectionne tous les éléments de la *`langue`* précisée.

`:last-child`
Sélectionne tous les éléments qui forment le dernier enfant de leur parent.

`:last-of-type`
Sélectionne tous les éléments qui constituent le dernier parmi les frères du même nom d'élément.

`:last`
Sélectionne le dernier parmi les éléments identifiés.

`:lt(`*`indice`*`)`
Sélectionne tous les éléments d'indice inférieur à un *`indice`* donné au sein de l'ensemble identifié.

`:not(`*`selecteur`*`)`
Sélectionne tous les éléments qui ne sont pas identifiés par le *`sélecteur`* donné.

`:nth-child(`*`n`*`)`
Sélectionne tous les éléments qui forment le *n*ième enfant de leur parent.

`:nth-last-child(`*`n`*`)`
Sélectionne tous les éléments qui constituent le *n*ième enfant de leur parent en commençant le comptage par le dernier élément jusqu'au premier.

`:nth-last-of-type(`*`n`*`)`
Sélectionne tous les éléments qui forment le *n*ième frère du même nom d'élément, en commençant le comptage du dernier élément au premier.

`:nth-of-type(`*`n`*`)`
Sélectionne tous les éléments qui sont le *n*ième enfant de leur parent par rapport aux frères de même nom d'élément.

`:odd`
Sélectionne les éléments impairs à partir de l'indice zéro. Voir aussi `even`.

`:only-child`
Sélectionne toutes les éléments qui constituent le seul enfant de leur parent.

`:only-of-type`
Sélectionne tous les éléments qui n'ont pas de frère de même nom d'élément.

`:parent`
Sélectionne tous les éléments qui possèdent au moins un nœud enfant (que ce soit un élément ou du simple texte).

`:password`
Sélectionne tous les éléments de type mot de passe (`password`).

`:radio`
Sélectionne tous les éléments de type bouton à option (`radio`).

`:reset`
Sélectionne tous les éléments de type bouton Annuler (`reset`).

`:root`
Sélectionne l'élément racine (`root`) du document.

`:selected`
Sélectionne tous les éléments sélectionnés (`selected`).

`:submit`
Sélectionne tous les éléments de type bouton Envoyer (`submit`).

`:target`
Sélectionne l'élément cible indiqué par l'identifiant de fragment de l'URL du document.

`:text`
Sélectionne tous les éléments de type `text`.

`:visible`
Sélectionne tous les éléments visibles.

Objets de jQuery

`event.currentTarget`
L'élément DOM courant dans la phase de remontée d'évènement en cascade.

`event.data`
Un objet de données facultatif transmis à une méthode d'évènement lorsque celle-ci possède une association à un gestionnaire en charge de traiter l'évènement.

`event.delegateTarget`
L'élément auquel le gestionnaire d'évènement jQuery actuellement appelé a été associé.

`event.metaKey`
: Précise si la touche Méta a été pressée au moment du déclenchement de l'évènement.

`event.namespace`
: L'espace de nom (*namespace*) indiqué au moment du déclenchement de l'évènement.

`event.pageX`
: L'emplacement de la souris par rapport au bord gauche du document.

`event.pageY`
: L'emplacement de la souris par rapport au bord supérieur du document.

`event.relatedTarget`
: L'autre élément DOM impliqué dans l'évènement s'il existe.

`event.result`
: La dernière valeur renvoyée par un gestionnaire d'évènement lors de son dernier déclenchement par l'évènement courant, à moins que la valeur n'ait pas été définie.

`event.target`
: L'élément DOM qui a amorcé l'évènement.

`event.timeStamp`
: La différence en millisecondes entre l'horodatage de déclenchement de l'évènement par le navigateur et le 1er janvier 1970.

`event.type`
: Décrit la nature de l'évènement.

`event.which`
: Dans le cas d'évènements de clavier ou de souris, cette propriété précise la touche de clavier ou le bouton de souris pressés.

`jquery`
: Cette chaine contient le numéro de version de jQuery.

`length`
: Le nombre d'éléments contenus dans l'objet jQuery.

`jQuery.cssHooks`
: Un *hook* est un point d'insertion de code. Crée un point d'insertion directement dans jQuery pour remplacer la manière de retrouver ou de définir des propriétés CSS déterminées, de normaliser le nommage de propriétés CSS, ou offre un moyen de créer des propriétés personnalisées.

`jQuery.fx.interval`
: L'intervalle ou période (en millisecondes) de déclenchement des animations.

`jQuery.fx.off`
: Désactive globalement toutes les animations.

Méthodes de jQuery

`$`
: Renvoie une collection d'éléments identifiés, soit trouvés dans le DOM en fonction d'un ou plusieurs arguments transmis, soit créés à l'aide d'une chaine HTML transmise.

`add`
: Ajoute des éléments à l'ensemble des éléments identifiés.

`addBack`
: Ajoute le précédent ensemble d'éléments de la pile à l'ensemble courant, facultativement filtrés par un sélecteur.

`addClass`
: Ajoute la ou les classes indiquées à l'ensemble des éléments identifiés.

`after`
: Insère du contenu précisé par le paramètre après chaque élément de l'ensemble des éléments identifiés.

`ajaxComplete`
: Inscrit un gestionnaire à appeler après l'achèvement des requêtes Ajax.

`ajaxError`
: Inscrit un gestionnaire à appeler quand des requêtes s'achèvent par une erreur.

`ajaxSend`
: Associe une fonction à exécuter avant d'envoyer une requête Ajax.

`ajaxStart`
: Inscrit un gestionnaire à appeler lorsque la première requête Ajax débute.

`ajaxStop`
: Inscrit un gestionnaire à appeler lorsque toutes les requêtes Ajax se sont achevées.

`ajaxSuccess`
: Associe une fonction à exécuter chaque fois qu'une requête Ajax s'achève avec succès.

`animate`
: Exécute une animation personnalisée à partir d'un ensemble de propriétés CSS.

`append`
: Insère le contenu précisé par le paramètre à la fin de chaque élément de l'ensemble des éléments identifiés.

`appendTo`
: Insère chaque élément de l'ensemble des éléments identifiés à la fin de la cible.

`attr`
: Détermine la valeur d'un attribut pour le premier élément de l'ensemble des éléments identifiés ou définit un ou plusieurs attributs pour chacun des éléments identifiés.

`before`
: Insère le contenu précisé par le paramètre avant chaque élément de l'ensemble d'éléments identifiés.

`bind`
: Associe un gestionnaire à un évènement pour les éléments.

`blur`
: Associe un gestionnaire à l'évènement JavaScript `blur`, de perte de cible de saisie, ou déclenche cet évènement sur un élément.

`change`
: Associe un gestionnaire à l'évènement JavaScript `change` ou déclenche cet évènement sur un élément.

`children`
: Détermine les enfants de chaque élément de l'ensemble des éléments identifiés, filtrés en option par un sélecteur.

`clearQueue`
: Supprime de la file d'attente (*queue*) tous les objets qui n'ont pas encore été exécutés.

`click`
: Associe un gestionnaire à l'évènement JavaScript `click` ou déclenche cet évènement sur un élément.

`clone`
: Crée une copie en profondeur de l'ensemble d'éléments identifiés.

`closest`
: Pour chaque élément de l'ensemble, récupère le premier élément qui correspond au sélecteur par le test de l'élément lui-même et en remontant le long de ses ancêtres au sein de l'arborescence DOM.

`contents`
: Détermine les enfants de chaque élément de l'ensemble des éléments identifiés, y compris les nœuds de texte et de commentaires.

`css`
: Détermine la valeur d'une propriété de style pour le premier de l'ensemble des éléments identifiés ou définit une ou plusieurs propriétés CSS pour chacun des éléments identifiés.

`data`
: Mémorise des données arbitraires associées aux éléments identifiés ou renvoie la valeur d'un stockage nommé de données pour le premier élément de l'ensemble des éléments identifiés.

`dblclick`
: Associe un gestionnaire à l'évènement JavaScript `dblclick` ou déclenche cet évènement sur un élément.

`callbacks.add`
: Ajoute une fonction de rappel (*callback*) ou une collection de fonctions de rappel à une liste de telles fonctions.

`callbacks.disable`
: Désactive une liste de fonctions de rappel pour les empêcher de s'actionner.

`callbacks.disabled`
: Détermine si une liste de fonctions de rappel a été désactivée.

`callbacks.empty`
: Supprime toutes les fonctions de rappel d'une liste.

`callbacks.fire`
: Appelle toutes les fonctions de rappel avec les arguments donnés.

`callbacks.fired`
: Détermine si les fonctions de rappel ont déjà été appelées au moins une fois.

`callbacks.fireWith`
: Appelle toutes les fonctions de rappel d'une liste avec le contexte et les arguments donnés.

`callbacks.has`
: Détermine si une fonction de rappel donnée est présente dans une liste.

`callbacks.lock`
: Verrouille une liste de fonctions de rappel dans son état courant.

`callbacks.locked`
: Détermine si la liste de fonctions de rappel a été verrouillée.

`callbacks.remove`
: Supprime une fonction de rappel ou une suite de fonctions de rappel d'une liste.

`deferred.always`
: Ajoute des gestionnaires à appeler lorsque l'objet différé (`Deferred`) est soit résolu, soit rejeté.

`deferred.done`
: Ajoute des gestionnaires à appeler lorsque l'objet différé est résolu.

`deferred.fail`
: Ajoute des gestionnaires à appeler lorsque l'objet différé est rejeté.

`deferred.notify`
: Appelle `progressCallbacks` sur un objet différé avec les arguments donnés.

`deferred.notifyWith`
: Appelle `progressCallbacks` sur un objet différé avec le contexte et les arguments donnés.

`deferred.progress`
: Ajoute des gestionnaires à appeler quand un objet différé génère des avis de progression.

`deferred.promise`
: Renvoie la promesse (`Promise`) d'un objet différé.

`deferred.reject`
: Rejette un objet différé et appelle tout `failCallbacks` avec les arguments donnés.

`deferred.rejectWith`
: Rejette un objet différé et appelle tout `failCallbacks` avec le contexte et les arguments donnés.

`deferred.resolve`
: Résout un objet différé et appelle tout `doneCallbacks` avec les arguments donnés.

`deferred.resolveWith`
: Résout un objet différé et appelle tout `doneCallbacks` avec le contexte et les arguments donnés.

`deferred.state`
: Détermine l'état courant d'un objet différé.

`deferred.then`
: Ajoute des gestionnaires à appeler quand l'objet différé est résolu, rejeté ou toujours en cours.

`delay`
: Règle une minuterie pour retarder l'exécution des éléments suivants de la file d'attente.

`delegate`
: Associe un gestionnaire à un ou plusieurs évènements pour tous les éléments identifiés par le sélecteur, maintenant ou à l'avenir, en fonction d'un ensemble déterminé d'éléments racines.

`dequeue`
: Exécute la fonction suivante de la file d'attente (*queue*) pour les éléments identifiés.

`detach`
: Supprime l'ensemble des éléments identifiés pour le DOM.

`each`
: Itère sur un objet jQuery et exécute une fonction pour chaque élément identifié dans l'objet.

`empty`
: Supprime tous les nœuds enfants de l'ensemble des éléments identifiés du DOM.

`end`
: Termine l'opération la plus récente de filtrage dans la séquence courante et renvoie l'ensemble des éléments identifiés à son état précédent.

`eq`
: Réduit l'ensemble des éléments identifiés à celui présent à l'indice indiqué.

`event.isDefaultPrevented`
: Renvoie si `preventDefault` a déjà été appelé sur cet objet d'évènement.

`event.isImmediatePropagationStopped`
: Renvoie si `stopImmediatePropagation` a déjà été appelé sur cet objet d'évènement.

`event.isPropagationStopped`
: Renvoie si `stopPropagation` a déjà été appelé sur cet objet d'évènement.

`event.preventDefault`
: Si cette méthode est appelée dans le gestionnaire d'évènement, l'action par défaut de cet évènement n'est pas déclenchée.

`event.stopImmediatePropagation`
: Empêche l'exécution du reste des gestionnaires et l'évènement de remonter en cascade le long de l'arborescence du DOM.

`event.stopPropagation`
: Empêche l'évènement de remonter en cascade le long de l'arborescence du DOM et interdit toute notification de l'évènement au moindre gestionnaire parent.

`fadeIn`
: Affiche les éléments identifiés avec un fondu enchainé vers l'opacité.

`fadeOut`
: Masque progressivement les éléments identifiés avec un fondu enchainé vers la transparence.

`fadeTo`
: Règle l'opacité des éléments identifiés.

`fadeToggle`
: Affiche ou masque les éléments identifiés, par une animation de leur opacité.

`filter`
: Réduit l'ensemble des éléments identifiés à ceux qui correspondent au sélecteur ou réussissent le test de la fonction.

`find`
: Détermine des descendants de chaque élément dans l'ensemble courant d'éléments identifiés, filtrés par un sélecteur, un objet jQuery ou un élément.

`finish`
: Arrête l'animation en cours d'exécution, supprime toutes les animations placées en file d'attente et achève toutes les animations pour les éléments identifiés.

`first`
: Réduit l'ensemble des éléments identifiés au premier de l'ensemble.

`focus`
: Associe un gestionnaire d'évènement à l'évènement JavaScript `focus`, de réception de la cible de saisie, ou déclenche cet évènement sur un élément. Sélectionne l'élément s'il a actuellement la cible de saisie.

`focusin`
: Associe un gestionnaire d'évènement à l'évènement `focusin` de réception du focus.

`get`
: Récupère les éléments du DOM identifiés par l'objet jQuery.

`has`
: Réduit l'ensemble des éléments identifiés à ceux qui possèdent un descendant qui correspond au sélecteur ou à l'élément DOM.

`hasClass`
: Détermine si au moins un élément identifié est affecté à la classe donnée.

`height`
: Détermine la hauteur courante calculée pour le premier élément dans l'ensemble des éléments identifiés ou définit la hauteur de chaque élément identifié.

`hide`
: Masque les éléments identifiés.

`hover`
: Associe un ou deux gestionnaires aux éléments identifiés, à exécuter quand le pointeur de la souris entre dans, et quitte, les éléments.

`html`
: Détermine le contenu HTML du premier élément dans l'ensemble des éléments identifiés ou définit le contenu HTML de chaque élément identifié.

`index`
: Recherche un élément donné parmi les éléments identifiés.

`innerHeight`
: Détermine la hauteur intérieure courante calculée pour le premier élément dans l'ensemble des éléments identifiés, y compris la marge intérieure mais sans tenir compte de la bordure.

`innerWidth`
: Détermine la largeur intérieure courante calculée pour le premier élément dans l'ensemble des éléments identifiés, y compris la marge intérieure mais sans tenir compte de la bordure, ou définit la largeur intérieure de chaque élément identifié.

`insertAfter`
: Insère chaque élément dans l'ensemble des éléments identifiés après la cible.

`insertBefore`
: Insère chaque élément dans l'ensemble des éléments identifiés avant la cible.

`is`
: Vérifie l'ensemble d'éléments identifiés courant par rapport à un sélecteur, un élément ou un objet jQuery et renvoie `true` si au moins un de ces éléments correspond aux arguments donnés.

`jQuery`
: Renvoie une collection d'éléments identifiés, soit trouvés dans le DOM en fonction d'argument(s) passé(s), soit créés par le passage d'une chaine HTML.

`jQuery.ajax`
: Effectue une requête HTTP asynchrone (Ajax).

`jQuery.ajaxPrefilter`
: Gère les options Ajax personnalisées ou modifie les options existantes avant l'envoi de chaque requête et avant son traitement par Ajax.

`jQuery.ajaxSetup`
: Définit les valeurs par défaut des futures requêtes Ajax. Son utilisation est déconseillée.

`jQuery.ajaxTransport`
: Crée un objet pour traiter la transmission pratique de données Ajax.

`jQuery.Callbacks`
: Un objet de liste de fonctions de rappel à buts multiples qui fournit un moyen puissant de gestion de listes de fonctions de rappel.

`jQuery.contains`
: Vérifie si un élément DOM est un descendant d'un autre élément DOM.

`jQuery.cssHooks`
: Place un point d'insertion (hook) directement dans jQuery pour remplacer la manière de lire ou de définir des propriétés CSS déterminées, pour normaliser le nommage de propriétés CSS ou créer des propriétés personnalisées.

`jQuery.data`
: Mémorise des données arbitraires associées à l'élément donné et (ou) renvoie la valeur qui a été définie.

`jQuery.Deferred`
: Cette fonction de constructeur renvoie un objet utilitaire chainable, avec des méthodes pour inscrire plusieurs fonctions de rappel dans des files d'attente de fonctions de rappel, appeler des files d'attente de fonctions de rappel et relayer l'état de succès ou d'échec de toute fonction synchrone ou asynchrone.

`jQuery.dequeue`
: Exécute la fonction suivante de la file d'attente de l'élément identifié.

`jQuery.each`
: Une fonction générique d'itérateur qui permet d'itérer de manière transparente parmi tant des objets que des tableaux. L'itération dans les tableaux, et les objets ressemblant à des tableaux avec une propriété `length` (tel que l'objet `arguments` d'une fonction), s'effectue à l'aide d'un indice numérique de 0 à `length-1`. L'itération parmi les autres objets s'effectue à l'aide de leurs propriétés nommées.

`jQuery.error`
: Prend une chaine et lève une exception qui contient celle-ci.

`jQuery.extend`
: Fusionne le contenu de deux objets ou plus dans le premier objet.

`jQuery.fn.extend`
: Fusionne le contenu d'un objet dans le prototype jQuery pour fournir de nouvelles méthodes d'instance jQuery.

`jQuery.get`
: Charge des données d'un serveur à l'aide d'une requête HTTP Get.

`jQuery.getJSON`
: Charge des données encodées en JSON à partir du serveur à l'aide d'une requête HTTP Get.

`jQuery.getScript`
: Charge un fichier JavaScript à partir d'un serveur à l'aide d'une requête HTTP Get, puis l'exécute.

`jQuery.globalEval`
: Exécute du code JavaScript de manière globale.

`jQuery.grep`
: Trouve les éléments d'un tableau qui satisfont une fonction de filtrage. Le tableau original ne subit aucune modification.

`jQuery.hasData`
: Détermine si à un élément correspondent des données jQuery associées.

`jQuery.holdReady`
: Retient ou libère l'exécution de l'évènement `ready` de jQuery.

`jQuery.inArray`
: Recherche une valeur déterminée au sein d'un tableau et renvoie son indice (ou `-1` si elle est introuvable).

`jQuery.isArray`
: Détermine si l'argument est un tableau.

`jQuery.isEmptyObject`
: Vérifie si un objet est vide, c'est-à-dire s'il ne contient aucune propriété énumérable.

`jQuery.isFunction`
: Détermine si l'argument passé est un objet fonction de JavaScript.

`jQuery.isNumeric`
: Détermine si son argument est un nombre.

`jQuery.isPlainObject`
: Vérifie si un objet est un objet à part entière, c'est-à-dire créé à l'aide de `{}` ou de `new Objet`).

`jQuery.isWindow`
: Détermine si l'argument est une fenêtre.

`jQuery.isXMLDoc`
: Vérifie si un nœud DOM figure au sein d'un document XML (ou s'il est lui-même un document XML).

`jQuery.makeArray`
: Convertit un objet ressemblant à un tableau en un authentique tableau JavaScript.

`jQuery.map`
: Convertit tous les éléments d'un tableau ou d'un objet en un nouveau tableau d'éléments.

`jQuery.merge`
: Fusionne le contenu de deux tableaux dans le premier tableau.

`jQuery.noConflict`
: Renonce au contrôle par jQuery du nom de variable `$`.

`jQuery.noop`
: Une fonction vide. Le terme `noop` signifie no operation, soit aucune action.

`jQuery.now`
: Renvoie un nombre qui représente le temps courant.

`jQuery.param`
: Crée une représentation sérialisée d'un tableau ou d'un objet adéquat pour son utilisation dans une chaine de requête URL ou Ajax.

`jQuery.parseHTML`
: Parcourt et analyse le contenu d'une chaine pour la scinder et la répartir en un tableau de nœuds DOM.

`jQuery.parseJSON`
: Prend une chaine JSON correctement formée et renvoie l'objet JavaScript résultant.

`jQuery.parseXML`
: Parcourt et analyse le contenu d'une chaine pour la répartir dans un document XML.

`jQuery.post`
: Charge des données d'un serveur à l'aide d'une requête HTTP Post.

`jQuery.queue`
: Affiche ou manipule la file d'attente de fonctions à exécuter sur l'élément identifié.

`jQuery.removeData`
: Supprime une donnée mémorisée préalablement.

`jQuery.trim`
: Supprime l'espace vide du début ou de la fin d'une chaine.

`jQuery.type`
: Détermine la classe JavaScript interne d'un objet.

`jQuery.unique`
: Trie un tableau d'éléments DOM en place et supprime les doublons. Cette méthode ne fonctionne que sur des tableaux d'éléments DOM et non sur des chaines ni des nombres.

`jQuery.when`
: Fournit un moyen d'exécuter des fonctions de rappel basées sur un ou plusieurs objets, généralement des objets différés qui représentent des évènements asynchrones.

`keydown`
: Associe un gestionnaire d'évènement à l'évènement JavaScript `keydown`, de touche enfoncée, ou déclenche cet évènement sur un élément.

`keypress`
: Associe un gestionnaire d'évènement à l'évènement JavaScript `keypress`, de touche pressée, ou déclenche cet évènement sur un élément.

`keyup`
: Associe un gestionnaire d'évènement à l'évènement JavaScript `keyup`, de touche relâchée, ou déclenche cet évènement sur un élément.

`last`
: Réduit l'ensemble des éléments identifiés à l'élément final de l'ensemble.

`load`
: Charge des données d'un serveur et place le HTML restitué dans l'élément identifié.

`map`
: Passe chaque élément de l'ensemble d'éléments identifiés courant à travers une fonction, pour produire un nouvel objet jQuery contenant les valeurs renvoyées.

`mousedown`
: Associe un gestionnaire d'évènement à l'évènement JavaScript `mousedown`, de bouton de souris pressé, ou déclenche cet évènement sur un élément.

`mouseenter`
: Associe un gestionnaire d'évènement à déclencher lorsque la souris entre dans un élément, ou déclenche ce gestionnaire sur un élément.

`mouseleave`
: Associe un gestionnaire d'évènement à déclencher lorsque la souris quitte un élément, ou déclenche ce gestionnaire sur un élément.

`mousemove`
: Associe un gestionnaire d'évènement à l'évènement JavaScript `mousemove`, de déplacement de souris, ou déclenche cet évènement sur un élément.

`mouseout`
: Associe un gestionnaire d'évènement à l'évènement JavaScript `mouseout`, de souris sortante, ou déclenche cet évènement sur un élément.

`mouseover`
: Associe un gestionnaire d'évènement à l'évènement JavaScript `mouseover`, de survol de souris, ou déclenche cet évènement sur un élément.

`mouseup`
: Associe un gestionnaire d'évènement à l'évènement JavaScript `mouseup`, de bouton de souris relâché, ou déclenche cet évènement sur un élément.

`next`
: Prend le frère immédiatement suivant de chaque élément dans l'ensemble des éléments identifiés. Si un sélecteur est fourni, elle ne récupère le frère que s'il correspond à ce sélecteur.

`nextAll`
: Prend tous les frères suivants de chaque élément dans l'ensemble des éléments identifiés, filtrés en option par un sélecteur.

`nextUntil`
: Prend tous les frères suivants de chaque élément jusque, mais non compris, l'élément identifié par le sélecteur, le nœud DOM ou l'objet jQuery passé.

`off`
: Supprime un gestionnaire d'évènement.

`offset`
: Détermine les coordonnées courantes du premier élément ou définit les coordonnées de tous les éléments d'un ensemble d'éléments identifiés, relatives au document.

`offsetParent`
: Détermine l'élément ancêtre le plus proche à l'emplacement indiqué.

`on`
: Associe une fonction de gestionnaire d'évènement pour un ou plusieurs évènements aux éléments sélectionnés.

`one`
: Associe un gestionnaire à un évènement pour les éléments. Le gestionnaire est exécuté maximum une fois par élément et type d'évènement.

`outerHeight`
: Détermine la hauteur calculée courante du premier élément dans l'ensemble des éléments identifiés, en tenant compte de la marge intérieure, de la bordure et éventuellement la marge. Renvoie une représentation numérique (sans le `px`) de la valeur ou `null` si elle est appelée sur un ensemble vide d'éléments.

`outerWidth`
: Détermine la largeur calculée courante du premier élément dans l'ensemble des éléments identifiés, en tenant compte de la marge intérieure et de la bordure.

`parent`
: Détermine le parent de chaque élément dans l'ensemble courant d'éléments identifiés, filtré en option par un sélecteur.

`parents`
: Détermine les ancêtres de chaque élément dans l'ensemble courant d'éléments identifiés, filtrés en option par un sélecteur.

`parentsUntil`
: Détermine les ancêtres de chaque élément dans l'ensemble courant d'éléments identifiés, jusqu'à, mais non compris, l'élément identifié par le sélecteur, le nœud DOM ou l'objet jQuery.

`position`
: Détermine les coordonnées courantes du premier élément dans l'ensemble des éléments identifiés, par rapport au parent décalé.

`prepend`
: Insère un contenu précisé dans le paramètre au contenu de chaque élément dans l'ensemble des éléments identifiés.

`prependTo`
: Insère au début de la cible chaque élément dans l'ensemble des éléments identifiés.

`prev`
: Détermine le frère prédécesseur immédiat de chaque élément dans l'ensemble des éléments identifiés, filtré en option par un sélecteur.

`prevAll`
: Détermine tous les frères précédents de chaque élément dans l'ensemble des éléments identifiés, filtrés en option par un sélecteur.

`prevUntil`
: Détermine tous les frères précédents de chaque élément jusqu'à, mais non compris, l'élément identifié par le sélecteur, le nœud DOM ou l'objet jQuery.

`promise`
: Renvoie un objet promesse (`Promise`) à observer lorsque toutes les actions d'un certain type associées à la collection, placées ou non en file d'attente, se sont achevées.

`prop`
: Détermine la valeur d'une propriété pour le premier élément dans l'ensemble des éléments identifiés ou définit une ou plusieurs propriétés pour chaque élément identifié.

`pushStack`
: Ajoute une collection d'éléments DOM à la pile (*stack*) jQuery.

`queue`
: Affiche ou manipule la file d'attente de fonctions à exécuter sur les éléments identifiés.

`ready`
: Précise la fonction à exécuter lorsque le chargement du DOM est achevé au complet.

`remove`
: Supprime l'ensemble des éléments identifiés du DOM.

`removeAttr`
: Supprime un attribut de tous les éléments dans l'ensemble des éléments identifiés.

`removeClass`
: Supprime une seule classe, plusieurs classes ou toutes les classes de chaque élément dans l'ensemble des éléments identifiés.

`removeData`
: Supprime une donnée mémorisée préalablement.

`removeProp`
: Supprime une propriété de tous les éléments de l'ensemble des éléments identifiés.

`replaceAll`
: Remplace chaque élément cible par l'ensemble des éléments identifiés.

`replaceWith`
: Remplace chaque élément dans l'ensemble des éléments identifiés par le nouveau contenu fourni et renvoie l'ensemble des éléments qui ont été supprimés.

`resize`
: Associe un gestionnaire d'évènement à l'évènement JavaScript `resize`, de redimensionnement, ou déclenche cet évènement sur un élément.

`scroll`
: Associe un gestionnaire d'évènement à l'évènement JavaScript `scroll`, de défilement, ou déclenche cet évènement sur un élément.

`scrollLeft`
: Détermine l'emplacement de la barre de défilement horizontale pour le premier élément dans l'ensemble des éléments identifiés ou définit l'emplacement horizontal de la barre de défilement pour chaque élément identifié.

`scrollTop`
: Détermine l'emplacement de la barre de défilement verticale pour le premier élément dans l'ensemble des éléments identifiés ou définit l'emplacement vertical de la barre de défilement pour chaque élément identifié.

`scroselect`
: Associe un gestionnaire d'évènement à l'évènement JavaScript `select`, de sélection, ou déclenche cet évènement sur un élément.

`serialize`
: Encode un ensemble d'éléments de formulaire sous forme de chaine pour envoi en soumission.

`serializeArray`
: Encode un ensemble d'éléments de formulaire sous forme de tableau de noms et de valeurs.

`show`
: Affiche les éléments identifiés.

`siblings`
: Détermine les frères (*siblings*) de chaque élément dans l'ensemble des éléments identifiés, filtrés en option par un sélecteur.

`slice`
: Réduit l'ensemble des éléments identifiés à un sous-ensemble précisé par une plage d'indices donnée.

`slideDown`
: Affiche les éléments identifiés avec un mouvement de glissement vers le bas (déroulement).

`slideToggle`
: Affiche ou masque les éléments identifiés avec un mouvement de glissement.

`slideUp`
: Masque les éléments identifiés avec un mouvement de glissement vers le haut (enroulement).

`stop`
: Arrête l'animation en cours d'exécution sur les éléments identifiés.

`submit`
: Associe un gestionnaire d'évènement à l'évènement JavaScript `submit`, de soumission de formulaire, ou déclenche cet évènement sur un élément.

`text`
: Détermine le contenu de texte combiné des éléments dans l'ensemble des éléments identifiés, y compris leurs descendants, ou définit le contenu de texte des éléments identifiés.

`toArray`
: Récupère sous forme d'un tableau tous les éléments contenus dans l'ensemble jQuery indiqué.

`toggle`
: Affiche ou masque les éléments identifiés.

`toggleClass`
: Ajoute ou supprime une ou plusieurs classes de chaque élément dans l'ensemble des éléments identifiés, en fonction de la présence de la classe ou de la valeur de l'argument de basculement (*switch*).

`trigger`
: Exécute tous les gestionnaires et comportements associés aux éléments identifiés pour le type d'évènement précisé.

`triggerHandler`
: Exécute tous les gestionnaires associés à un élément pour un évènement donné.

`unbind`
: Supprime des éléments un gestionnaire d'évènement préalablement associé.

`undelegate`
: Supprime un gestionnaire de l'évènement donné pour tous les éléments qui correspondent au sélecteur courant, en fonction d'un ensemble déterminé d'éléments racines.

`unwrap`
: Supprime les parents de l'ensemble des éléments identifiés du DOM, tout en laissant les éléments identifiés à leur place.

`val`
: Détermine la valeur courante du premier élément dans l'ensemble des éléments identifiés ou définit la valeur de chaque élément identifié.

`width`
: Détermine la largeur calculée courante du premier élément dans l'ensemble des éléments identifiés ou définit la largeur de chaque élément identifié.

`wrap`
: Entoure d'une structure HTML chaque élément dans l'ensemble des éléments identifiés.

`wrapAll`
: Entoure d'une structure HTML tous les éléments de l'ensemble des éléments identifiés.

`wrapInner`
: Entoure d'une structure HTML le contenu de chaque élément dans l'ensemble des éléments identifiés.

Index

Symboles

A

B

C

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z