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).

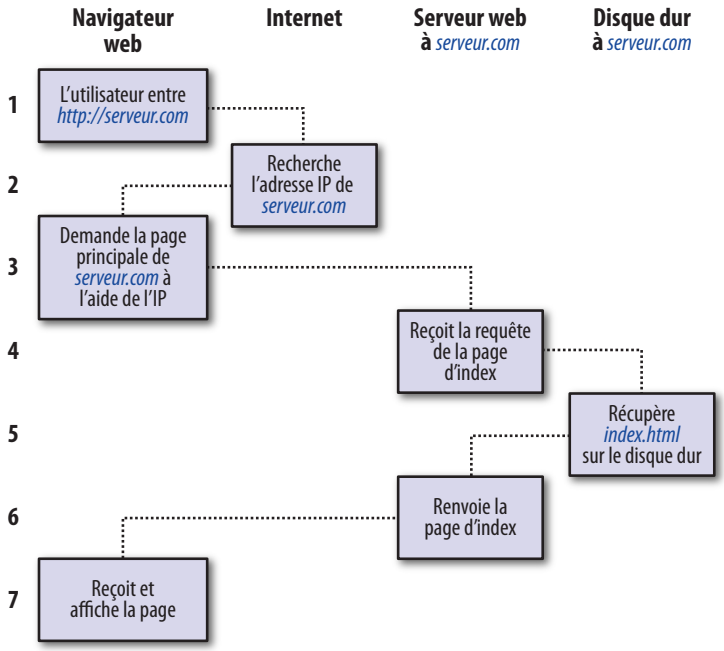

Figure 1-1. La séquence de base d'une requête-réponse entre client et serveur

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.

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).

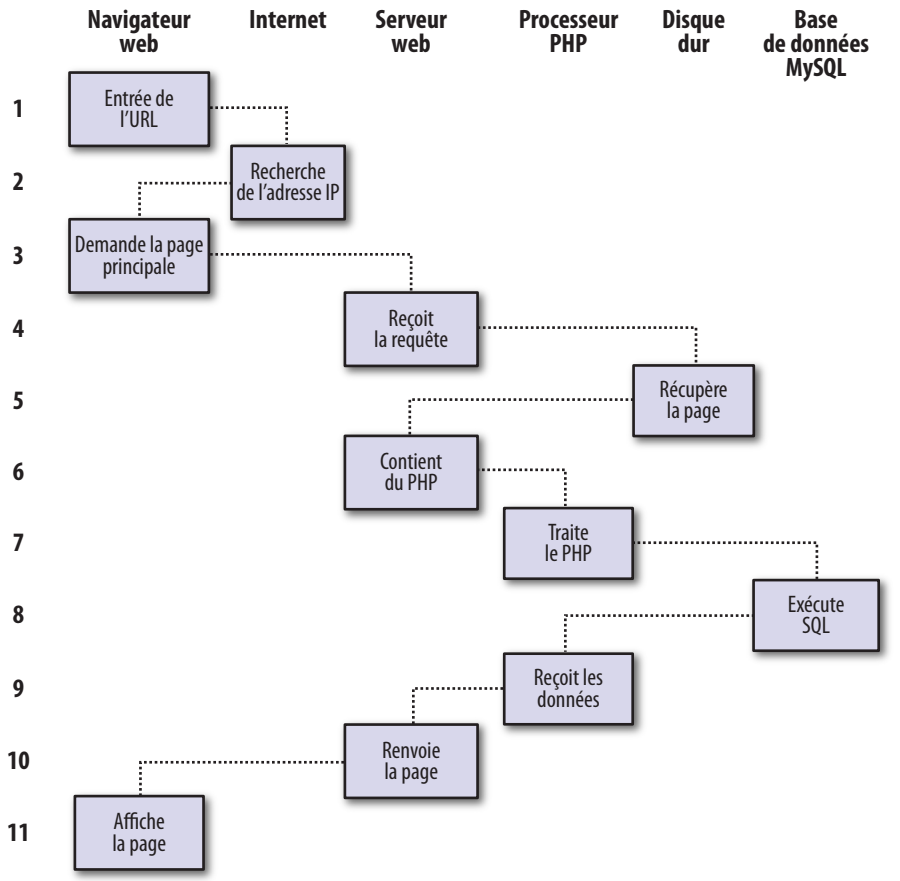

Figure 1-2. La séquence de requête-réponse dynamique entre client et serveur

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 l'environnement de développement 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 *microdata* 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 naitre 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).

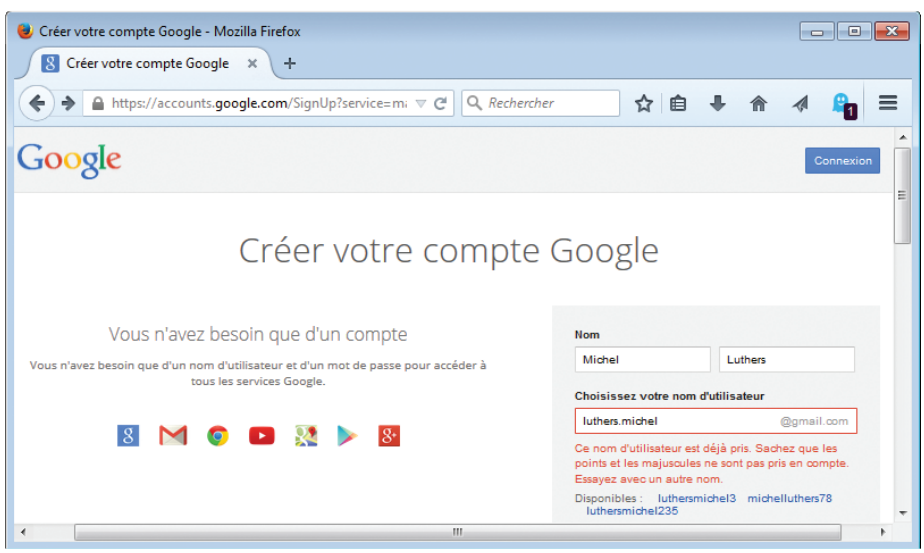

Figure 1-3. Gmail utilise Ajax pour vérifier la disponibilité des noms d'utilisateurs

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.