SMASHING MAGAZINE

HTML5

pour le Webdesign

Conception
Mise en œuvre des CSS3
Intégration des médias
Développement JavaScript

Bill Sanders

DUNOD

Cet ouvrage est la traduction
en langue française, par Dunod éditeur, de l'ouvrage
Smashing HTML5
de Bill Sanders

Couverture : Rachid Maraï

© Dunod, Paris, 2012, pour la traduction française
ISBN 978-2-10-058897-8

Table des matières

Troisième partie – Multimédia : images, sons et vidéos

Introduction

Cet ouvrage est dédié à Jacob Sanders.

En 1992, je suis parti à la découverte d'Internet (on ne parlait pas encore de *surf* à ce moment-là) avec un programme qui utilisait le protocole Gopher. Je me trouvais à El Paso, au Texas, et j'étais capable de consulter les horaires de chemin de fer entre Londres et Cambridge en Angleterre. À cette époque, cela tenait du miracle et c'était incroyable.

Je ne pensais pas que l'on ferait mieux que Gopher sur Internet, et peu de temps après cela est apparu le navigateur Mosaic et le World Wide Web. Netscape Navigator a rapidement supplanté Mosaic et j'ai découvert HTML. J'étais capable de visualiser du texte et des images et je pouvais en plus passer d'une page Web à une autre. En peu de temps, je me suis débrouillé pour créer mes propres pages Web à l'aide d'un éditeur de texte et du nouveau langage de balisage, HTML. Des types du service informatique me créèrent un serveur et je me suis lancé dans l'aventure.

Pendant un moment, c'était comme s'il sortait à peu près tous les ans une nouvelle version de HTML. CSS et JavaScript firent leur apparition et de plus en plus de navigateurs devinrent disponibles. Les progrès étaient continus, mais après HTML4 (qui incluait XHTML), les choses ont semblé stagner. Ces heures sombres de HTML ont duré de 2000 jusqu'à 2008. Puis le W3C (World Wide Web Consortium) publia un document de travail sur HTML5 en 2008. Pourtant après la publication de la norme HTML5 sous une forme provisoire, les choses évoluèrent très lentement jusqu'à ce que je mette la main sur un navigateur HTML5. L'équipe de développement des standards

a travaillé méthodiquement et projetait d'implémenter la version définitive de la norme dans les navigateurs en 2012 !

Puis un jour en 2009 ou en 2010, j'ai entendu parler d'une version bêta d'un navigateur qui prenait en charge HTML5, ou tout du moins certaines de ses fonctionnalités. En 2010, plusieurs navigateurs étaient compatibles HTML5, notamment les navigateurs pour les périphériques mobiles. Des blogs comme `www.smashingmagazine.com` publièrent des articles sur HTML5 et même si tout n'était pas prêt, HTML5 était bien là ! HTML5 s'était échappé du zoo et c'était la course pour produire des navigateurs HTML5. Nous étions officiellement rentrés dans la période de renaissance de HTML. L'excitation était de retour !

HTML5 est un sujet si vaste que j'ai dû choisir une thématique centrale qui englobe l'essence du langage de balisage sans tomber dans une encyclopédie de référence où j'aurais tenté de tout décrire sans rien expliquer. Naturellement, les nouvelles fonctionnalités devaient être le sujet central de l'ouvrage, mais elles n'apparaissent qu'associées avec d'autres balises et les lecteurs qui découvraient HTML pour la première fois avaient besoin de notions fondamentales. Je devais aussi laisser de côté les éléments abandonnés et montrer comment les éléments conservés et les nouveaux éléments travaillent de concert. De plus CSS3 et JavaScript jouent un rôle important, mais ils ne sont abordés que dans la mesure où ils ont un lien avec HTML5. J'ai donc pris la décision de diviser *HTML5 pour le webdesign* en quatre parties qui constituent l'essentiel de HTML5.

Partie I : Le langage du Web

La première partie de l'ouvrage débute par l'étude des différents navigateurs compatibles HTML5 (y compris les navigateurs mobiles) et donne des pistes pour démarrer avec cette nouvelle version de HTML. Elle traite aussi des méthodes de travail avec les différents types de fichiers et de leur organisation pour que la création des pages Web et des sites s'effectue rationnellement. Elle explique comment utiliser les balises (éléments) HTML5 avec les différents attributs que l'on peut leur assigner. Vous apprendrez aussi à exploiter les styles CSS3. À la fin de la première partie, vous saurez utiliser la couleur et les différents codes couleurs avec HTML5 et intégrer des modèles de couleurs à vos sites pour les améliorer.

Partie II : Conceptions de pages et de sites Web

La deuxième partie étudie en détail la création des pages Web et des sites Web. À une certaine époque, la seule préoccupation des concepteurs et des développeurs concernait la manière dont une page s'affichait sur l'écran d'un ordinateur, puis tout à coup, les utilisateurs de téléphones mobiles ont commencé à consulter des pages Web et il a fallu revoir les stratégies de conception pour inclure les utilisateurs de périphériques mobiles. Tout au long de ce livre, vous verrez des pages Web affichées sur des téléphones mobiles comme l'iPhone ou les téléphones Android. Outre les copies d'écran de navigateurs sous Windows 7 et Macintosh OS X, vous verrez donc

des pages affichées dans des navigateurs mobiles comme Mini Opera et Safari, ainsi que d'autres navigateurs mobiles dont vous ne soupçonnez même pas l'existence.

Partie III : Multimédia : images, sons et vidéos

Seul le chapitre 9 qui traite des images couvre un média qui était déjà disponible dans les versions précédentes de HTML. En revanche, les deux autres chapitres qui traitent de l'audio et de la vidéo constituent des nouveautés en HTML5. Certains formats vidéo sont relativement nouveaux et ont été développés pour être utilisés sur le Web. Tous les navigateurs HTML5 n'utilisent pas les mêmes formats vidéo, mais heureusement, HTML5 possède les structures pour vérifier les formats vidéo et trouver ceux qui sont compatibles avec un navigateur donné. Outre les éléments pour la gestion de l'audio et de la vidéo, j'étudie également dans cette partie les nouveaux attributs HTML5 qui optimisent l'emploi des médias sur le Web.

Partie IV : Balises HTML5 dynamiques et un soupçon de JavaScript

L'élément `canvas` constitue l'une des fonctionnalités les plus attendues de HTML5, mais pour en tirer le meilleur parti, vous devez à la fois maîtriser JavaScript et CSS3. Dans cette partie, vous apprenez donc suffisamment de JavaScript associé au DOM (Document Object Model) HTML5 pour travailler efficacement avec l'objet canvas et CSS3. De la même manière, HTML5 offre plusieurs nouveaux attributs de formulaires, mais comme avec la plupart des formulaires, ils ont besoin d'assistance pour traiter l'information. Grâce à JavaScript, vous allez apprendre à enregistrer les données des formulaires dans les nouveaux objets de stockage en HTML5. J'étudie aussi les nouveaux objets `geolocation` et leurs propriétés pour vous montrer comment votre page Web peut automatiquement charger une carte basée sur les coordonnées de votre position (latitude et longitude). Vous trouverez dans cette quatrième partie une quantité de nouveautés qui ajouteront de nombreuses fonctionnalités à vos sites.

À propos de l'auteur

Bill Sanders est l'un des enseignants fondateurs de l'Université de développement et de conception Web multimédia d'Hartford. Il y enseigne HTML5, la conception d'interface et d'information, CSS3, Flash, ActionScript 3.0, ASP.NET, C#, PHP et la vidéo en streaming. Il a écrit de nombreux livres sur la programmation Internet, notamment sur JavaScript et les modèles de conception ActionScript 3.0. Il réside à la campagne dans le Connecticut.

Remerciements

J'ai pris conscience de l'intérêt immédiat de HTML5 grâce à Michael Wilson, Zach Dunn et Nick Greenfield qui ont attiré mon attention sur cette technologie. Ils m'ont aussi présenté à l'équipe de *Smashing Magazine* et m'ont initié à quelques tendances émergentes. Chris Webb de chez Wiley m'a aidé à définir les orientations du livre et, avec Margot Hutchinson de chez Waterside Productions, il m'a permis de finaliser cet ouvrage. Ellie Scott de chez Wiley m'a assisté dans la prise en compte des nombreux détails qui aboutissent à la création d'un livre. Elizabeth Kuball, qui est une correctrice talentueuse, a clarifié et allégé ma prose, alors que Harvey Chute, en tant que relecteur technique, s'est assuré que tout le code fonctionnait correctement et m'a proposé des suggestions d'amélioration. Pour finir, je souhaite remercier mes collègues de l'Université de développement et de conception Web multimédia d'Hartford, John Gray et Brian Dorn, qui m'ont aidé quand j'en avais besoin, notamment ce jour critique où un point-virgule manquant avait fait des ravages dans un programme.

PREMIÈRE PARTIE

Le langage du Web

1

Introduction à HTML5

Objectif

Ce chapitre est une présentation générale des nouveautés de HTML5 (éléments ajoutés, conservés et supprimés). Pour le moment, la préoccupation majeure reste de trouver des navigateurs qui fonctionnent avec HTML5, de connaître ceux qui sont en développement et qui promettent une compatibilité HTML5 ainsi que la manière dont ils ont commencé à implémenter HTML5. Nous allons aussi étudier les nouveaux navigateurs qui sont spécifiquement développés pour les terminaux mobiles, de manière à ce que vous puissiez également tester les pages HTML5 sur ces périphériques. Pour commencer, téléchargez tous les navigateurs HTML5 étudiés dans ce chapitre de telle sorte que vous puissiez savoir ce que les utilisateurs verront quand ils liront une page Web HTML5 que vous aurez créée.

1.1 UTILISATION DES BALISES

La plupart du contenu d'Internet est créé en HTML (HyperText Markup Language). Vous seriez sans doute surpris d'apprendre que plusieurs applications que vous utilisez quotidiennement (par exemple votre traitement de texte) ont aussi été créées avec des langages de balisage. Cependant, comme pour tous les langages informatiques, on ne voit en HTML que le contenu et pas le langage sous-jacent. Le langage fonctionne comme la structure d'un bâtiment : vous savez qu'elle se trouve sous la peinture et les cloisons, mais vous ne pouvez pas la voir. Dans ce livre, je vais exposer en pleine lumière le langage HTML et vous montrer comment l'utiliser pour créer vos propres structures.

Si vous êtes familier des versions précédentes de HTML et XHTML, vous serez capable d'adapter vos connaissances à HTML5. Et si vous êtes totalement novice en matière de HTML, vous trouverez HTML5 assez simple. Au fond, tout ce que vous avez à faire, c'est de placer vos contenus entre une balise d'ouverture et une balise de fermeture, que l'on appelle un conteneur, et qui stylera votre texte ou affichera vos images et vos médias sur la page Web. La figure 1.1 illustre le concept de conteneur :

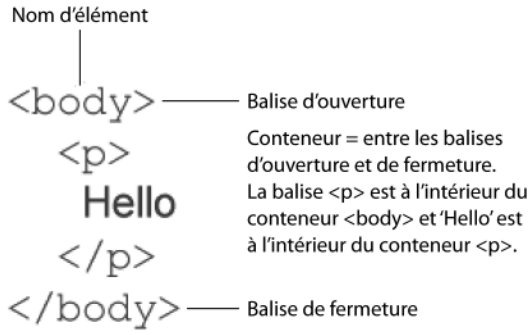


Figure 1.1 — Conteneurs en HTML5.

Par exemple, la ligne suivante :

```
<h1>Ceci est écrit en gros.</h1>
```

dit à l'interpréteur de votre navigateur de créer un texte écrit en gros qui ressemble à ceci :

Ceci est écrit en gros.

Le texte à l'intérieur des signes < > constitue le code. Dans cet exemple, h1 est le code pour écrire en gros. Les signes inférieur et supérieur disent où le conteneur commence (`<h1>`) et où il se termine (`</h1>`). Tout ce qui se trouve à l'intérieur du conteneur est configuré avec la taille et le style de la balise, qui sont intégrés dans la balise ou bien créés à l'aide de CSS3.

Pour appréhender HTML5, le plus simple et le plus amusant est de créer de petits exemples. Il vous suffit pour cela de saisir le code qui est fourni dans ce chapitre dans un éditeur de texte comme le Bloc-notes si vous êtes sous Windows ou bien TextEdit si vous êtes sur Mac. Enregistrez le fichier avec l'extension `.html`, puis ouvrez-le dans un navigateur HTML5. Pour ouvrir une page Web, démarrez votre navigateur et sélectionnez la commande Fichier → Ouvrir fichier (ou Ouvrir), et désignez le nom du fichier (vous pouvez aussi faire un double-clic sur l'icône du fichier pour ouvrir la plupart des fichiers HTML).

1.1.1 Incorporation des nouveaux éléments HTML5

Une balise est composée d'un élément et d'attributs. La balise est identifiée par son élément, par exemple `<h1>` où `h1` est l'élément. Quand on parle de balise, on fait en général référence à son élément, mais une balise est en fait constituée de son élément et d'attributs. Les attributs sont les différentes caractéristiques ou les propriétés d'un élément que l'on peut coder pour modifier le contenu du conteneur de la balise. Pour le moment, nous allons simplement parler des éléments, si bien que j'utiliserai les termes *balise* et *élément* de manière interchangeable.

Afin de vous donner un aperçu des nouveaux éléments en HTML5, le tableau 1.1 liste tous les nouveaux éléments ainsi qu'une courte description de chacun d'entre eux. Tout au long de cet ouvrage, je donnerai de nombreux exemples et expliquerai comment utiliser ces éléments.

Tableau 1.1 — Nouveaux éléments en HTML5

Élément	Description
`<article>`	Composition indépendante et autosuffisante au sein d'un document
`<aside>`	Contenu indirectement lié au contenu de l'article
`<audio>`	Conteneur de contenu sonore
`<canvas>`	Conteneur de contenu graphique
`<command>`	Commande que l'utilisateur peut invoquer
`<datalist>`	Génère une liste quand il est utilisé avec l'élément `<input>` et son nouvel attribut de liste
`<details>`	Fournit les détails d'un élément
`<embed>`	Plug-in ou contenu interactif externe
`<figcaption>`	Balise de légende pour l'élément figure
`<figure>`	Contient un groupe de médias et leur légende
`<footer>`	Conteneur pour un pied de page d'une section ou d'une page
`<header>`	Conteneur pour un titre d'une section ou d'une page
`<hgroup>`	Titre de section avec plusieurs éléments h1 à h6 d'un document
`<keygen>`	Représentation du contrôle de générateur de paires de clés

Tableau 1.1 — (suite)

`<mark>`	Chaîne de caractères dans un document, marquée ou surlignée pour être référencée dans un autre document
`<meter>`	Conteneur pour une plage connue de valeurs (par exemple, utilisation du disque)
`<nav>`	Représentation d'une section d'un document conçue pour la navigation
`<output>`	Définit la progression d'une tâche de toute sorte
`<progress>`	Représentation de la progression d'une tâche (comme le pourcentage complet d'une opération de téléchargement)
`<rp>`	Indicateur en annotation Ruby (un langage de programmation) pour définir ce qu'il faut afficher dans les navigateurs qui ne prennent pas en charge l'élément `<ruby>`
`<rt>`	Marque le composant texte Ruby d'une annotation Ruby
`<ruby>`	Élément pour les annotations Ruby
`<section>`	Identificateur de thème pour le regroupement de contenus
`<source>`	Conteneur pour plusieurs spécifications de ressources de médias
`<summary>`	Information sur un élément `<details>`
`<time>`	Conteneur pour un horodatage
`<video>`	Élément pour chaîner à un fichier vidéo
`<wbr>`	Représentation d'une possibilité de saut de ligne pour guider la césure de mots longs ou de chaînes de caractères

Certains des nouveaux éléments, comme `<video>` et `<audio>` ajoutent des contenus multimédias au HTML et représentent une avancée importante de HTML. D'autres, comme `<ruby>`, sont assez spécialisés et à moins que vous n'ayez besoin de certains caractères asiatiques, vous n'utiliserez probablement pas cet élément.

Un grand nombre de ces nouvelles balises ont pour caractéristique de travailler en conjonction avec CSS3 ou JavaScript. Cependant, la plupart des nouveaux éléments fonctionnent toujours de manière autonome. Quand on ajoute un style ou certaines fonctionnalités intéressantes, il se peut que l'on doive utiliser des rudiments de CSS3

ou de JavaScript, mais vous n'avez pas à apprendre la totalité du langage JavaScript ou même CSS3 pour en profiter.

Par exemple, le script suivant utilise le nouvel élément `<datalist>` qui n'était pas disponible dans les versions précédentes de HTML. Saisissez le code suivant dans un éditeur de texte, enregistrez-le dans un fichier nommé `Datalist.html`, ouvrez-le dans un navigateur et vous verrez comment il facilite la saisie des données (le fichier `Datalist.html` se trouve aussi dans les fichiers de ce chapitre).

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Datalist</title>
</head>
<body>
<body>
<p>
  <label> Quel langage suivant souhaitez-vous apprendre ?<br />
    <input type="text" name="web" list="lang">
    <datalist id="lang">
      <option value="HTML5">
      <option value="JavaScript">
      <option value="jQuery">
      <option value="ActionScript 3.0">
      <option value="Java">
    </datalist>
  </label>
  <br />
</p>
</body>
</html>
```

Quand on ouvre le fichier dans le navigateur Opera, on obtient une liste déroulante comme celle qui est illustrée à la figure 1.2.

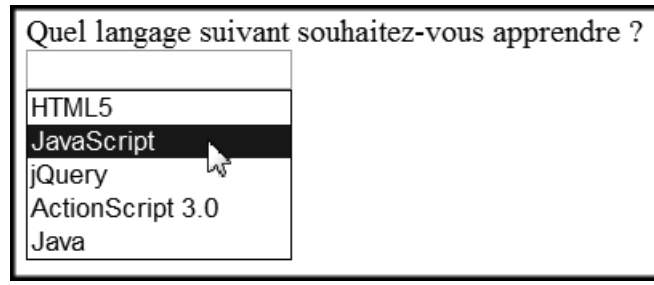


Figure 1.2 — Utilisation de la balise `<datalist>` dans un navigateur Opera.

À la différence des anciennes versions de HTML, l'utilisateur voit s'afficher une liste d'options.

1.1.2 Utilisation des balises des versions précédentes de HTML

Même si vous êtes familier avec HTML4 (ou des versions plus anciennes de HTML), vous serez surpris par le nombre d'éléments HTML que vous ne savez pas utiliser ou dont vous n'avez jamais entendu parler auparavant. Par exemple, que signifie la balise `<q>` ? Quand est-elle utilisée ? Si vous êtes novice en HTML, n'essayez pas de mémoriser tous les éléments du tableau 1.2, mais parcourez-le pour avoir une idée générale des balises disponibles.

Tableau 1.2 — Balises des versions précédentes de HTML et conservées en HTML5

Balise	Description
`<!--...-->`	Commentaire
`<!DOCTYPE>`	Type de document (un seul en HTML5)
`<a>`	Hyperlien vers une page ou une zone de page
`<abbr>`	Abréviation
`<address>`	Conteneur pour une adresse
`<area>`	Zone cliquable à l'intérieur d'une image
`<b>`	Texte gras
`<base>`	URL de base pour tous les liens d'une page
`<bdo>`	Direction d'affichage du texte
`<blockquote>`	Bloc de texte
`<body>`	Début d'un élément body
` `	Saut de ligne
`<button>`	Bouton cliquable
`<caption>`	Légende de tableau
`<cite>`	Conteneur pour une citation
`<code>`	Format de listing de code informatique
`<col>`	Définit les attributs des colonnes de tableau
`<colgroup>`	Conteneur pour les groupes de colonnes de tableau
`<dd>`	Conteneur pour une valeur de l'élément `<dt>`
`<del>`	Conteneur pour du texte supprimé
`<dfn>`	Représentation de la définition d'un terme
`<div>`	Démarcation de division d'un document

Tableau 1.2 — (suite)

<dl>	En-tête d'une liste d'associations
<dt>	Spécification d'un nom dans un groupe nom-valeur (liste de descriptions)
	Texte en emphase
<fieldset>	Conteneur pour un ensemble de contrôles de formulaire
<form>	Conteneur pour un formulaire (en général avec des éléments input)
<h1> to <h6>	Titre de niveau 1 à 6
<head>	Conteneur pour le premier code à être interprété par le navigateur
<hr>	Règle horizontale (ligne)
<html>	Conteneur pour un document HTML
<i>	Texte en italique
<iframe>	Frame avec un cadre local (sous-fenêtre)
<img>	Conteneur image
<input>	Champ de saisie dans un conteneur de formulaire
<ins>	Conteneur pour délimiter du texte inséré dans un paragraphe
<kbd>	Conteneur pour la saisie au clavier de l'utilisateur
<label>	Représentation d'une légende dans l'interface utilisateur
<legend>	Titre dans un cadre de fieldset
	Indicateur d'élément de liste
<link>	Référence de ressource (par exemple, CSS)
<map>	Conteneur d'image cliquable
<mark>	Texte dans un contexte marqué pour du texte dans un contexte différent
<menu>	Conteneur pour une liste de commandes
<meta>	Conteneur pour des informations meta
<object>	Conteneur pour un objet incorporé (par exemple, un fichier SWF)
	Liste numérotée
<optgroup>	Option de regroupement d'éléments dans une liste d'options
<option>	Conteneur pour des options individuelles d'une liste déroulante
<p>	Bloc de paragraphe
<param>	Paramètres de plug-in
<pre>	Format de texte préformaté

Tableau 1.2 — (suite)

<q>	Texte encadré par des guillemets
<samp>	Extrait de code informatique
<script>	Conteneur pour script CSS, JavaScript ou un autre type reconnu
<select>	Liste sélectionnable
<small>	Texte affiché en petit
	Section d'un document
	Texte enrichi qui ressemble au texte gras
<style>	Conteneur pour définition de style
<sub>	Texte en indice
<sup>	Texte en exposant
<table>	Définition de tableau
<tbody>	Démarcation pour un bloc de lignes d'un corps de tableau
<td>	Cellule de tableau
<textarea>	Conteneur de zone de texte
<tfoot>	Représentation d'un bloc de lignes de synthèses de colonnes d'un tableau
<th>	Format d'en-tête de tableau
<thead>	Représentation d'un bloc de ligne de synthèses de colonnes d'un en-tête de tableau
<title>	Titre du document
<tr>	Démarcation d'une ligne de tableau
	Liste non ordonnée
<var>	Style de variable dans une formule

La plupart des éléments ayant le même nom en HTML4 sont identiques en HTML5, mais certains ont une signification légèrement différente. Les règles de certaines balises ont aussi été modifiées. Par exemple, dans la création des tableaux, la balise pour spécifier une ligne, <tr>, ne nécessite plus une balise de fermeture </tr>. Certains attributs d'éléments ont également changé. Au fur et à mesure que vous apprendrez les nouvelles caractéristiques de HTML5, vous constaterez qu'un grand nombre d'éléments anciens ont de nouvelles caractéristiques. Le script HTML de

tableau suivant fournit un nouvel exemple d'anciens éléments. Saisissez ce texte dans un éditeur et enregistrez-le sous le nom `NouveauAncienTableau.html`, et ouvrez-le dans le navigateur Opera.

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Tableau</title>
</head>
<body>
<table>
  <caption>
  =Types d'éléments=
  </caption>
  <thead>
    <tr>
      <th> Type
      <th> Texte
      <th> Image
  <tbody>
    <tr>
      <th> Interface
      <td> saisie de texte
      <td> bouton
    <tr>
      <th> Liens
      <td> souligné
      <td> icone
</table>
</body>
</html>
```

La figure 1.3 illustre l'aspect du tableau.

=Types d'éléments=		
Type	**Texte**	**Image**
Interface	saisie de texte	bouton
Liens	souligné	icone

Figure 1.3 — Tableau créé en HTML5.

En général, on n'utilise pas les tableaux pour mettre en forme du texte. Au lieu de cela, les tableaux sont utilisés pour formater des données, comme celles d'une base de données ou des données créées dynamiquement par un autre programme tel que JavaScript. En HTML5, cependant, les tableaux utilisés conjointement avec CSS3 offrent de meilleures possibilités de mise en forme qu'avec les versions précédentes de HTML et CSS.

1.1.3 Renoncement ou remplacement des balises abandonnées

Le dernier ensemble de balise (voir le tableau 1.3) semblera familier à ceux qui connaissent HTML4 et les précédentes versions de HTML. Les balises suivantes ont été abandonnées parce qu'elles posaient un certain nombre de problèmes, ou elles ont été remplacées par d'autres structures qui gèrent mieux ce qu'elles faisaient.

Si vous êtes novice en HTML, vous pouvez jeter un coup d'œil à cette liste pour avoir une idée de ce qu'il faut éviter. En travaillant avec HTML, vous trouverez de nombreux exemples sur le Web, et vous les incorporerez probablement dans votre propre code. Cependant, dans la mesure où HTML5 est tellement nouveau, vous constaterez que la plupart du code HTML a été créé avec d'anciennes versions qui peuvent contenir des balises obsolètes et il faudra les remplacer par les nouvelles structures.

Tableau 1.3 — Balises abandonnées

Balise supprimée	Supprimée ou remplacée
`<acronym>`	Remplacée par `<abbr>`
`<applet>`	Remplacée par `<object>`
`<basefont>`	Mieux gérée par CSS
`<bgsound>`	Remplacée par `<audio>`
`<big>`	Mieux gérée par CSS
`<blink>`	Supprimée en HTML5
`<center>`	Mieux gérée par CSS
`<dir>`	Remplacée par `<ul>`
`<font>`	Supprimée en HTML5
`<frame>`	Supprimée en HTML5
`<frameset>`	Supprimée en HTML5
`<isindex>`	Remplacée par `<form>`
`<marquee>`	Supprimée en HTML5
`<multicol>`	Supprimée en HTML5
`<nobr>`	Supprimée en HTML5
`<noframes>`	Supprimée en HTML5
`<noscript>`	Uniquement conforme à la syntaxe HTML
`<s>`	Mieux gérée par CSS
`<spacer>`	Supprimée en HTML5

Tableau 1.3 — (suite)

<strike>	Mieux gérée par CSS
<tt>	Mieux gérée par CSS
<u>	Mieux gérée par CSS

 est l'une des balises les plus courantes qui ait été abandonnée, mais il est facile de centrer du texte avec un peu de code CSS, comme l'illustre l'exemple suivant. Saisissez ce texte dans un éditeur et enregistrez-le sous le nom CentrezMoi.html, puis ouvrez-le dans un navigateur Web.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
h1 {
     text-align:center;
 }
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Centrage avec CSS</title>
</head>
<body>
<h1>Les titres peuvent être centrés</h1>
</body>
</html>
```

Quand vous testerez le code dans votre navigateur, vous verrez :

Les titres peuvent être centrés

Cela peut sembler représenter beaucoup de travail pour obtenir un simple titre centré, mais les pages sont généralement courtes et vous pouvez centrer n'importe quel titre avec une balise <h1> car vous avez modifié le comportement de la balise. Vous pouvez modifier n'importe quelle balise avec CSS (nous en apprendrons plus sur CSS3 au chapitre 3, mais vous l'avez déjà utilisé si vous voyez le titre au milieu de la page).

1.2 CHOIX D'UN NAVIGATEUR POUR INTERPRÉTER HTML5

Si vous voulez lancer une discussion animée avec des développeurs HTML5, il suffit de poser la question suivante : « Quel est le meilleur navigateur ? ». Mais le plus important n'est pas le navigateur qu'utilisent les autres développeurs, mais bien celui des visiteurs de votre site Web. En général, les développeurs utilisent le meilleur

navigateur jusqu'à ce qu'il en apparaisse un autre encore plus performant que celui qui est employé par la moyenne des utilisateurs du Web. Si vous voulez que les gens qui visitent votre site aient la meilleure expérience possible, essayez de déterminer le navigateur qu'ils utilisent majoritairement. Il est encore préférable quand on développe des logiciels pour soi-même ou un client de tester ses pages Web sur la totalité des principaux navigateurs et au moins sur les deux plateformes principales, Macintosh et Windows. Il existe aussi des navigateurs sous Linux, mais très peu de personnes utilisent leur système Linux pour naviguer sur le Web.

Si l'on examine les principaux navigateurs qui prennent en charge HTML5, la plupart peuvent être utilisés sous Windows ou Macintosh, mais il arrive qu'un navigateur nécessite un système d'exploitation récent. Dans ces conditions, si vous avez un système d'exploitation ancien, assurez-vous que les prérequis du navigateur sont compatibles avec votre système d'exploitation.

Il y a plusieurs années, Microsoft a arrêté le développement d'Internet Explorer (IE) pour Macintosh, alors qu'Apple a développé une version Windows de son navigateur Safari. Trois éditeurs de logiciels, Google, Mozilla et Opera, ne développent pas de systèmes d'exploitation pour ordinateur, mais créent en revanche des navigateurs. Dans cette section, je vais présenter Firefox, Chrome, Opera, Safari et IE9.

Gardez à l'esprit que les fonctionnalités des navigateurs évoluent avec le temps. Ce que je décris ici représente donc l'actualité au moment où j'ai écrit ces lignes, mais cela a pu changer quand vous lirez ce livre.

1.2.1 Mozilla FireFox

Mozilla plonge ses racines dans le premier navigateur conçu par Netscape, nommé Netscape Navigator, qui est sorti au début des années 1990. Il mettait en scène une mascotte qui ressemblait à la créature du film Godzilla. Mosaic était un navigateur développé à l'Université de l'Illinois, qui devint plus tard Netscape Navigator. La combinaison de Mosaic et de Godzilla donna Mozilla, qui est actuellement une société à but non lucratif, la Fondation Mozilla. Firefox est le principal navigateur de Mozilla qui prend en charge HTML5.

Outre le support des deux systèmes d'exploitation Windows et Macintosh, Firefox prend également en charge Linux. Si Linux n'est pas considéré comme un système d'exploitation important pour les ordinateurs domestiques, il l'est en revanche pour les serveurs. Firefox est disponible gratuitement sur tous les systèmes d'exploitation. La figure 1-4 illustre une copie d'écran d'une application HTML5 dans Firefox.

Vous noterez que la barre d'adresse (la fenêtre où vous avez saisi l'URL du fichier HTML) fait référence à `file:///C:...` au lieu d'une adresse `http://`. Cela est dû au fait que la page réside sur votre ordinateur. Vous pourrez aussi constater que la page paraît différente si elle est affichée dans un environnement Windows que si elle s'affiche sur un Macintosh, quand bien même il s'agit du même navigateur (la page affichée n'est qu'un exemple et ne sélectionne pas le navigateur à votre place).

Figure 1.4 — Mozilla Firefox.

1.2.2 Google Chrome

Google, célèbre pour son moteur de recherche et ses cartes, a développé son propre navigateur, Chrome, en ayant dès le début comme préoccupation la compatibilité HTML5. Chrome est disponible gratuitement pour les systèmes tournant sous Apple, Windows et Linux. La figure 1.5 illustre exactement la même page sur le même ordinateur que celle de la figure 1.4 (essayez de voir si vous détectez les différences).

Mis à part les différences de styles des deux navigateurs, il peut être difficile de voir les différences de la page. Avec une page simple, les différences subtiles n'affecteront pas l'aspect de la page Web, mais si vos pages sont plus grandes et plus complexes, de petites différences peuvent apparaître.

Adobe propose un outil de développement, Browserlab (disponible à la page `https://browserlab.adobe.com`), qui permet de voir l'aspect d'une page Web dans différents navigateurs en même temps. Browserlab peut être exécuté directement à partir d'Adobe Dreamweaver CS5, mais vous pouvez aussi vous rendre sur la page Web de Browserlab. Afin de visualiser un peu mieux les différences, comparons l'exemple

Figure 1.5 — Google Chrome affichant la même page HTML5 que celle illustrée à la figure 1-4.

de page Web dans Firefox sur un Macintosh avec un ordinateur sous Windows 7 et exécutant Google Chrome. La figure 1.6 illustre l'aspect de la comparaison côte à côte.

La différence est due en partie à la manière dont Windows et Mac OS affichent le texte et l'interface utilisateur. Browserlab propose un autre mode d'affichage appelé Pelure d'oignon qui superpose les deux captures d'écran afin de mieux voir où le texte et l'interface utilisateur apparaissent. La figure 1.7 illustre cette différence.

Plus la pelure d'oignon apparaît floue, plus grandes sont les différences d'affichage de la page Web. À la figure 1.7, vous pouvez constater que l'affichage est très flou, ce qui signifie que des différences importantes existent entre les navigateurs et les systèmes d'exploitation.

1.2.3 Opera

Quand j'ai examiné le navigateur Opera à l'époque où je commençais à tester différents navigateurs, il semblait avoir les meilleures fonctionnalités HTML5. De plus, Opera a un navigateur spécial, Opera Mini 5, que vous pouvez télécharger gratuitement sur votre terminal mobile. HTML5 fonctionne bien sur les terminaux mobiles, comme

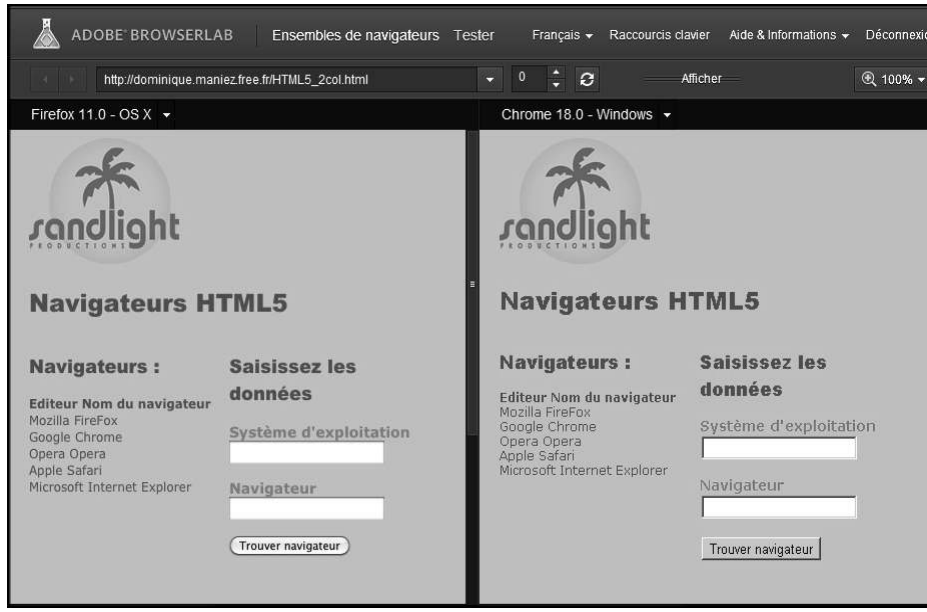


Figure 1.6 — Comparaisons de navigateurs dans Adobe Browserlab.

Figure 1.7 — Affichage en mode Pelure d'oignon de navigateurs superposés.

vous pouvez le constater à la figure 1.8, qui affiche l'exemple de page Web sur un iPhone exécutant le navigateur mobile d'Opera.

Des versions complètes du navigateur Opera sont également disponibles sous Windows, Macintosh et Linux. Quand on crée des pages Web, on doit les planifier pour différentes tailles de terminaux. Comme vous pouvez le voir, l'exemple d'application

Figure 1.8 — Navigateur Opera Mini 5.

que nous avons utilisé fonctionne aussi bien sur un terminal mobile que sur un écran d'ordinateur.

1.2.4 Apple Safari

Apple a développé le navigateur Safari pour Macintosh et Windows ainsi que sur les terminaux mobiles. À des fins de comparaison, la figure 1.9 illustre l'aspect de l'application exemple sur Mobile Safari, le navigateur développé par Apple pour l'iPhone. Comparez cette copie d'écran avec celle d'Opera Mini 5 de la figure 1.8.

Figure 1.9 — Navigateur Mobile Safari.

Tout comme il y a peu de différences entre l'apparence des pages Web visualisées sur un ordinateur de bureau ou un portable, vous ne devriez pas voir beaucoup de différences entre les affichages des divers navigateurs pour terminaux mobiles. C'est

une excellente chose ! Les développeurs Web perdent beaucoup de temps à tenter de s'assurer que leurs pages s'affichent de manière identique sur les différents navigateurs et les différentes plateformes. Avec une implémentation commune de HTML5, cela ne devrait plus être un problème. Des fonctionnalités, comme la possibilité d'ouvrir des onglets, ou d'autres caractéristiques qui facilitent la navigation sur le Web, sont intéressantes pour autant que l'implémentation de HTML5 par le navigateur soit réalisée selon les spécifications définies par le World Wide Web Consortium (W3C).

1.2.5 Microsoft Internet Explorer 9

Selon Microsoft, le navigateur IE9 est totalement compatible avec la norme HTML5. La figure 1.10 illustre la page de test affichée dans IE9.

Figure 1.10 — Internet Explorer 9.

1.2.6 Prévisualisation d'affichages différents

Comme vous l'avez constaté, les pages Web peuvent être affichées sur un certain nombre de navigateurs et de systèmes d'exploitation différents. Les développeurs Web ont donc besoin de prendre en compte les caractéristiques des terminaux sur lesquels leurs pages seront visualisées, comme un ordinateur de bureau ou bien un téléphone mobile. Supposons que vous développiez pour iPhone et iPad (ou n'importe quel autre terminal mobile ou tablette) ; si vous pouvez prévisualiser votre travail côte à côte, vous serez en mesure de faire des comparaisons. Adobe Dreamweaver, un outil de développement de page Web, permet d'afficher plusieurs terminaux de dimensions différentes en même temps (voir la figure 1.11).

Vous pouvez modifier les dimensions du terminal. Par exemple, un Motorola Droid affiche un écran de 854 x 480 alors qu'un Sony VAIO UX affiche un écran

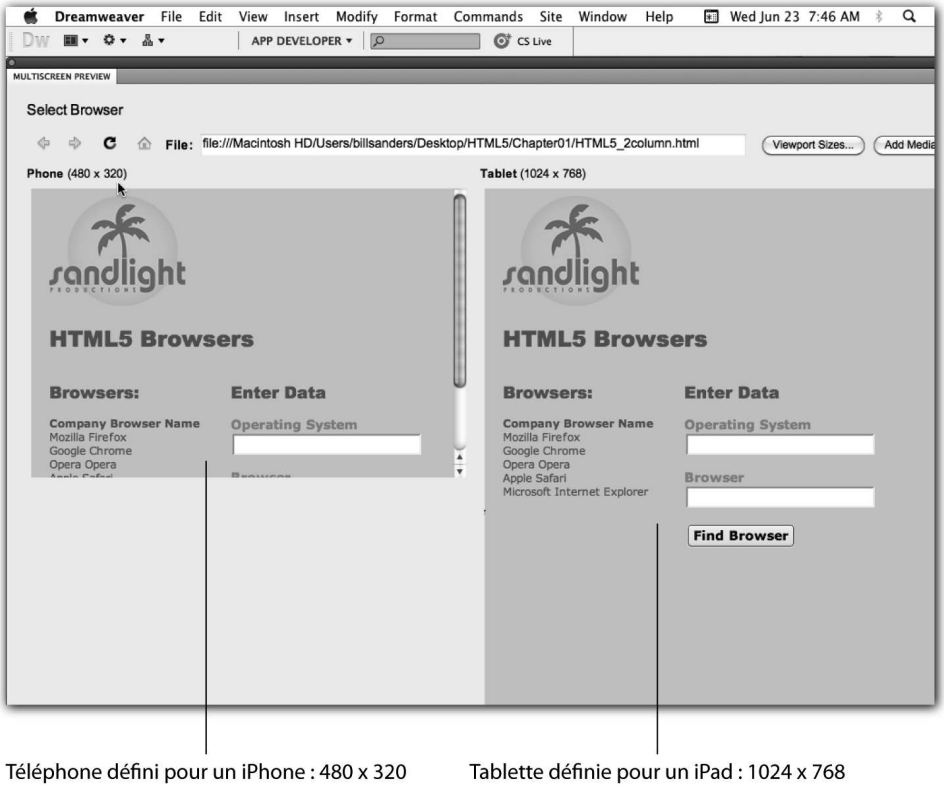


Figure 1.11 — Affichage d'écrans multiples dans Adobe Dreamweaver.

de 1024 x 600. L'affichage d'écrans multiples vous aide à concevoir votre page et à l'optimiser en fonction de vos différents utilisateurs. Trouver le meilleur compromis est un art, mais la tâche sera moins pénible si vous connaissez autant que possible votre public et le matériel qu'il utilise pour visualiser les pages.

1.3 À VOUS DE JOUER !

Pour commencer en douceur, ce premier exemple vous permet d'ajouter des informations sur vous-même. Ne vous inquiétez pas si vous ne comprenez pas tout (et même si vous ne comprenez rien). Vous devez remplacer les valeurs textuelles dans les zones encadrées par deux signes égal (=) qui se suivent et inscrire des informations personnelles. Enregistrez cette page sur votre ordinateur sous le nom `EnRoute1.html` (vous trouverez également ce programme dans les fichiers qui accompagnent cet ouvrage).

```
<!DOCTYPE HTML>
<html>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<head>
<style type="text/css">
body {
     background-color:blanchedAlmond;
     color:saddleBrown;
     font-family:Verdana, Geneva, sans-serif;
     font-size:12px;
     margin-left:20px;
}
h1, h2 {
     font-family:"Arial Black", Gadget, sans-serif;
     color:midnightBlue;
}
h1 {
     text-align:center;
}
h3 {
     background-color:goldenrod;
     color:ghostwhite;
     font-size:11px;
     font-family:"Arial";
}
</style>
<title>La roue</title>
</head>
<body>
<h1> ==Votre nom== : Le pro du développement Web HTML5</h1>
<h2> ==Nom de votre société== fournit des services Web complets</h2>
<ul>
  <li>==Service 1==</li>
  <li>==Service 2== </li>
  <li>==Service 3== </li>
  <li>==Service 4== </li>
  <li>==Service 5== </li>
</ul>
<h3>  Tous les services sont garantis. Notre service contentieux est
joignable à : ==URL où la plainte doit être envoyée==  . </h3>
</body>
</html>
```

Testez cette page dans un navigateur et vérifiez qu'elle s'affiche comme prévu. Vous pouvez aussi regarder l'affichage sur différents navigateurs (n'oubliez pas que les navigateurs sont gratuits) et sur votre terminal mobile. Si vous voulez faire plus de modifications, rendez-vous à la page suivante :

```
www.w3.org/TR/css3-color/#svg-color
```

Vous y trouverez une liste de noms de couleurs que vous pourrez utiliser avec HTML5. Essayez de changer dans le code le nom des couleurs en choisissant des couleurs qui vous plaisent.

2

Balises HTML5

Objectif

Les programmeurs classifient les langages informatiques en langages de bas niveau qui sont très proches du langage binaire de l'ordinateur et en langages de haut niveau qui sont proches du langage naturel. La version 5 du HyperText Markup Language (HTML5) est un langage de très haut niveau. Cependant, la première version de HTML avait très peu de « mots » pour décrire ce que les développeurs et les concepteurs désiraient. Au fur et à mesure de la croissance du Web, les exigences sur HTML ont progressé. Grâce aux CSS (Cascading Style Sheet) et à JavaScript, les concepteurs ont pu améliorer la qualité des pages Web, mais il manquait encore beaucoup de choses.

La création des pages Web a été facilitée par la sortie de plug-ins qui étaient capables d'exécuter des langages comme Java et des applications réalisées en Flash. En fait, la plupart des navigateurs embarquaient la dernière version du plug-in pour Flash de telle sorte que les utilisateurs pouvaient visualiser les pages créées avec Flash et Flash Builder (Flex).

Les développeurs Web en voulaient cependant toujours plus et souhaitaient exécuter HTML et CSS en mode natif à partir du navigateur. Les éditeurs de navigateurs ajoutaient tranquillement à JavaScript des fonctionnalités qui étaient nécessaires pour travailler avec les nouveaux éléments de HTML5. Avec les nouvelles versions de chaque navigateur, non seulement HTML5 était implémenté totalement, mais c'était aussi le cas de JavaScript et CSS3. Ce chapitre explique comment les différents éléments de HTML5 fonctionnent et comment ils se combinent avec CSS3 et JavaScript.

2.1 ANALYSE SYNTAXIQUE DU CODE

Tôt ou tard, vous entendrez l'expression *analyse syntaxique du code* (Ndt : en anglais, on emploie le terme *parsing* pour désigner l'analyse syntaxique ; le programme qui réalise cette opération est appelé *parser ;* comme le terme anglais est beaucoup plus court, il est parfois utilisé tel quel en français et on parlera volontiers d'un programme qui parse un fichier XML) à propos des navigateurs et de HTML5, CSS3 et JavaScript. Tout cela signifie que le navigateur lit le code et l'interprète pour réaliser ce qu'on lui dit de faire. C'est comme un interprète qui parle français et russe : l'interprète analyse le russe de telle manière que le locuteur français le comprenne et il analyse le français de telle sorte que le locuteur russe le comprenne. À proprement parler, l'analyseur est une partie de l'interpréteur du navigateur, mais pour des raisons pratiques, on se représente l'analyseur comme étant impliqué dans la traduction des balises utilisées dans le fichier Web.

Pour analyser correctement HTML5, deux choses sont nécessaires : vous devez écrire le code proprement et votre navigateur doit l'interpréter correctement. C'est la raison pour laquelle les normes sont importantes. Fondamentalement, les normes garantissent que lorsque vous écrivez du code HTML5 selon les règles établies, votre code réalise ce qui est prévu sur tous les navigateurs et tous les ordinateurs. En utilisant HTML5, CSS3 et JavaScript avec les navigateurs que nous avons étudiés au chapitre 1, vous ne devriez pas avoir de mauvaises surprises s'ils sont totalement compatibles HTML5.

Paradoxalement, les normes autorisent une grande part de créativité chez les développeurs et les concepteurs. Si vous voulez que votre page agisse d'une certaine manière en suivant les normes utilisées par les navigateurs qui interprètent vos créations, elle aura l'aspect attendu et se comportera comme prévu. Si vous-même ou le navigateur ne respectez pas les normes, votre créativité tombera à l'eau (ce qui n'est bien entendu pas notre souhait).

2.1.1 HTML5 et les fichiers connexes

Pour créer un fichier HTML5, comme cela a été vu au chapitre 1, il suffit d'enregistrer le code à l'aide d'un éditeur de texte tel que le Bloc-notes (Windows) ou TextEdit (Mac) et d'utiliser l'extension `.html` à la fin du nom du fichier (`MaPage.html`, par exemple). L'extension `.html` est importante car elle est reconnue en tant que page Web et seul un navigateur peut l'analyser. Vous constaterez aussi que seul un petit nombre d'autres types de fichiers sont reconnus par l'interpréteur du navigateur. Voici les extensions les plus courantes que vous rencontrerez :

- `.jpg` (fichier graphique JPEG)
- `.gif` (fichier graphique GIF)
- `.png` (fichier graphique PNG)
- `.svg` (fichier graphique SVG)
- `.css` (Cascading Style Sheet)
- `.js` (fichier JavaScript)

Parmi ces fichiers, les plus importants sont les fichiers graphiques car les outils que vous utilisez pour vos images peuvent les enregistrer automatiquement sous des formats différents que ceux employés pour le Web. Par exemple, Adobe Photoshop enregistre automatiquement les fichiers au format `.psd`, et Adobe Illustrator au format `.ai`. Aucun de ces deux formats de fichier graphique ne peut être utilisé avec ces pages Web. Cependant, la plupart des outils de création graphique peuvent enregistrer les fichiers dans les formats `.jpg`, `.gif`, ou `.png` si vous utilisez la commande Enregistrer sous à la place de la commande Enregistrer. Quand on utilise la commande Enregistrer sous, on peut sélectionner un format particulier parmi une liste de formats disponibles.

Réglage des paramètres d'extension de fichier par défaut sous Windows

Par défaut, Windows 7 (et les versions précédentes) masque les extensions de fichier. Cela procure une apparence plus propre des noms de fichiers, mais si vous devez choisir entre un fichier graphique ayant une extension `.psd` et un fichier ayant une extension `.png`, vous devez pouvoir visualiser son extension. Voici comment procéder :

- Ouvrez le panneau de configuration.
- Choisissez Apparence et personnalisation → Options des dossiers → Cliquez sur l'onglet Affichage.
- Désactivez la case à cocher Masquer les extensions des fichiers dont le type est connu (voir la figure).

Figure 2.1 — Désactivation sous Windows de l'option Masquer les extensions des fichiers dont le type est connu.

Vous serez à présent capable de voir toutes les extensions de fichier. Ainsi, quand vous voulez charger un fichier graphique, vous saurez s'il s'agit d'un fichier `.png`, `.jpg` ou `.gif` en regardant simplement le nom du fichier affiché à l'écran.

Réglages des paramètres de TextEdit sur votre Mac

Si TextEdit sur votre Mac a ses paramètres par défaut, vous risquez d'avoir des problèmes quand vous enregistrerez des fichiers au format HTML. Cela est dû au fait que le type de fichier par défaut dans lequel TextEdit enregistre ses fichiers est le format Rich Text Format (`.rtf`) et non pas le format texte (`.txt`). Au format `.rtf`, votre texte est enregistré avec un autre code qui n'est pas compatible avec les pages Web. Voici ce que vous devez faire pour régler ce problème :

- Ouvrez TextEdit.
- Dans le menu TextEdit, au sommet de l'écran, choisissez Préférences. La boîte de dialogue Préférences apparaît.
- Sélectionnez le bouton radio Format Texte (voir la figure).

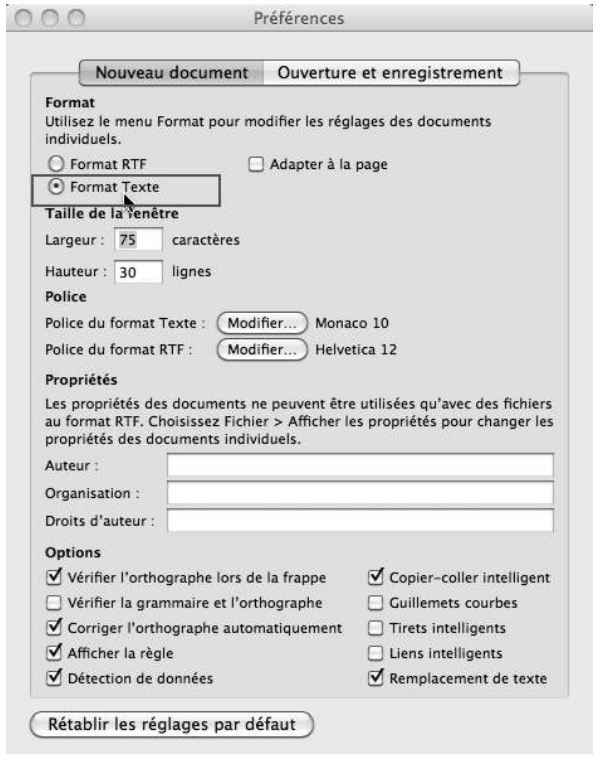


Figure 2.2 — Modification de l'option Format RTF en Format Texte dans TextEdit.

À présent, quand vous créez une page HTML5 dans TextEdit, votre fichier est enregistré par défaut au format `.txt` et vous pouvez le modifier en format `.html` en utilisant la commande Enregistrer sous.

2.1.2 Fichiers fonctionnant avec le Web

Si vous êtes néophyte dans l'écriture des pages Web, vous devez d'abord apprendre à reconnaître les fichiers qui fonctionnent avec les pages Web. Par défaut, HTML5 reconnaît l'extension `.html` et les extensions des trois types de fichiers graphiques que nous avons vus plus haut (`.jpg`, `.png`, et `.gif`). Vous verrez cependant parfois des références à des fichiers `.css`. Il s'agit de fichiers CSS externes, au format CSS3 ou des versions plus anciennes. Il existe aussi des fichiers JavaScript dotés de l'extension `.js` qui peuvent être référencés par une page Web.

Les navigateurs qui analysent les fichiers HTML analysent également les fichiers CSS et JavaScript. En fait, certains fichiers HTML contiennent du code CSS et JavaScript écrit directement au milieu de balises HTML. Quand vous voyez la balise `<script>`, vous constaterez la présence d'un script JavaScript ou CSS dans le conteneur de script (entre la balise d'ouverture `<script>` et la balise de fermeture `</script>`). Le développeur peut choisir de placer le code CSS et JavaScript dans des fichiers externes et de le charger avec la balise `<link>` pour le code CSS et la balise `<script>` pour le code JavaScript.

Par exemple, le code suivant charge le fichier externe `.css` `joliStyle.css` :

```
<link rel="stylesheet" type="text/css" href="joliStyle.css" />
```

Avec JavaScript, le fichier externe `.js` est appelé à partir du conteneur `<script>` plutôt qu'à l'intérieur d'une balise `<link>`. Le code suivant charge un fichier JavaScript nommé `faitMiracle.js` :

```
<script language="JavaScript" src="faitMiracle.js" />
```

Cet ouvrage se concentre sur HTML5, mais vous aurez absolument besoin de CSS3 pour mettre en forme vos pages si bien que nous l'étudierons également. La plupart du temps, vous verrez du code CSS incorporé dans du code HTML. Au chapitre 3, vous en apprendrez plus sur l'utilisation de CSS3 avec HTML5. Le chapitre 12 aborde l'utilisation de JavaScript avec les balises HTML5 et vous apprendrez exactement à créer et à utiliser JavaScript avec HTML5.

2.2 FONCTIONNEMENT DES BALISES

Quand on écrit du code en HTML5, on a besoin de savoir quels éléments utiliser pour obtenir ce que l'on souhaite. Comme nous l'avons vu au chapitre 1, on peut modifier la taille et l'apparence d'une police en utilisant la balise `<h1>`. Pour commencer, vous n'allez pas modifier les balises avec une CSS. Quand on utilise `<h1>`, on peut s'attendre à obtenir le même texte écrit en gros caractères toutes les fois (avec une CSS vous pouvez modifier la balise pour qu'elle affiche du texte en petits caractères de couleur verte si vous le souhaitez, mais il faudra attendre le chapitre 3 pour apprendre à réaliser cela).

En bref, vos balises fonctionnent en divisant la page en sections qui commencent par une balise d'ouverture `<élément>` et se terminent par une balise de fermeture `</élément>`. Vous pouvez écrire toutes les pages HTML5 que vous voulez avec cette méthode sans ajouter grand-chose d'autre et vos pages fonctionneront parfaitement. Naturellement, vous allez vouloir créer des pages un peu plus sophistiquées et montrer au navigateur de quoi vous êtes capable, mais la plupart du temps vous allez vous contenter d'écrire des balises. Nous allons donc commencer par nous focaliser sur le conteneur de base HTML5.

2.2.1 Balise HTML de base

Si vous êtes familier de HTML4 et de la description du type de document, vous savez déjà que vous pouvez ajouter des informations détaillées qui indiquent au navigateur à quoi il doit s'attendre. Ainsi, la première balise que vous avez besoin de prendre en compte n'est pas véritablement une balise HTML, mais une balise qui dit au navigateur que vous écrivez en HTML5 et que vous n'utilisez pas une des nombreuses versions de HTML4 ou de XHTML. Voici cette balise :

```
<!DOCTYPE HTML>
```

C'est tout ! Il n'y a rien de compliqué, mais cela annonce simplement au navigateur qu'il doit s'attendre à un document de type HTML5. Chaque page Web que vous créez doit commencer par cette balise, et vous n'avez pas besoin d'une balise de fermeture. Le point d'exclamation (`!`) indique qu'il ne s'agit pas d'une balise HTML, mais de quelque chose d'un peu différent.

2.2.2 Description de la page avec des balises

Juste après la première balise qui dit au navigateur ce à quoi il peut s'attendre, on commence avec un conteneur HTML (tout ce qui est compris entre la balise d'ouverture et la balise de fermeture). Cette balise annonce le début du code HTML qui se termine quand le navigateur rencontre la balise de fermeture. La balise de fermeture HTML se trouve à la fin de chaque page HTML.

Après l'élément HTML on trouve le conteneur `<head>`. Vous pouvez vous représenter cette portion de la page comme une zone de maintenance. Tout ce qui se trouve dans cette partie sera chargé en premier, que ce soit ou non utilisé dans le reste de la page HTML. Pour commencer, tout ce qui va dans la zone head constitue le titre de la page. Le titre apparaît au sommet de la page Web quand on l'exécute. Voici un exemple de titre :

```
<title>Seriously Sweet Page</title>
```

Ce titre apparaît dans les onglets et la fenêtre de la page. Si vous n'en mettez pas, vous obtiendrez un titre vide ou bien un titre par défaut. La figure 2-1 illustre l'apparition du titre dans différents navigateurs.

Comme vous pouvez le voir, le titre Seriously Sweet Page apparaît en différents endroits sur les quatre principaux navigateurs. Sur certains, il apparaît au sommet de la fenêtre et sur l'onglet, sur d'autres uniquement au sommet de la fenêtre, ou bien encore uniquement sur l'onglet. Cela aide l'utilisateur à trouver sa page quand plusieurs pages sont ouvertes simultanément (ou bien tout simplement à rappeler à l'utilisateur quelle page il consulte). On trouve beaucoup d'autres contenus dans le conteneur `<head>`, comme du code CSS et JavaScript, mais pour le moment, retenez seulement qu'il faut inclure un titre.

Ensuite, la balise `<body>` délimite le début du contenu de la page. Comme son nom l'indique (*body* signifie corps en anglais), il s'agit de la partie principale de toute page Web, et seul le contenu à l'intérieur du conteneur `<body>` est visible sur la page. Entre les éléments d'ouverture et de fermeture, on place tout ce que l'on veut afficher sur la page. L'ensemble de balises suivant doit se trouver sur chaque page que vous créez (vous pouvez en fait aussi l'utiliser comme modèle et l'enregistrer quelque part de manière à le réutiliser plus tard pour ne pas avoir à commencer avec une page de code vide).

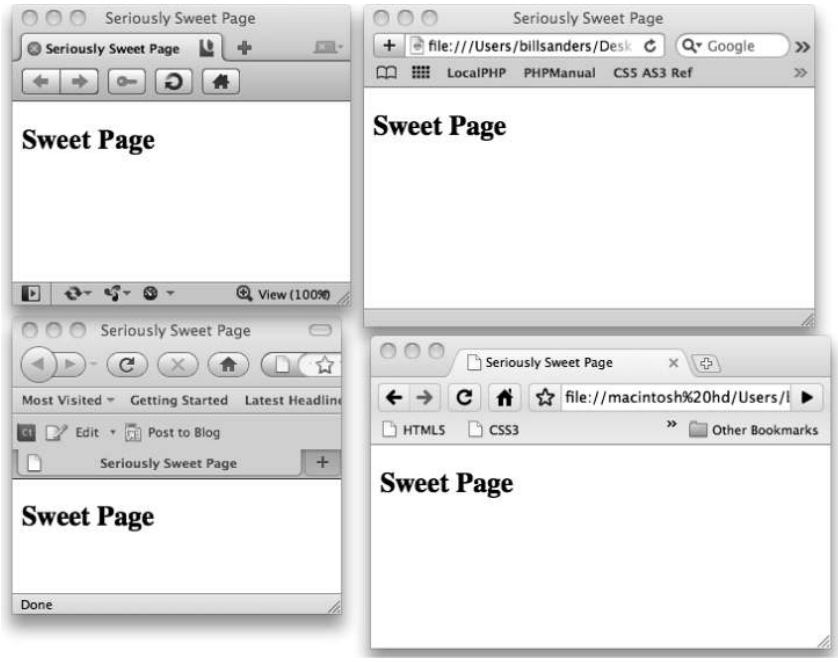


Figure 2.3 — Le titre apparaît sur les pages Web et les onglets.

```
<!DOCTYPE HTML>
<html>
<head>
<title>Le titre se place ici</title>
</head>
<body>
```

```
Le contenu se place ici : Ma page vraiment chouette
</body>
</html>
```

Au fur et à mesure que vous progresserez dans la lecture de cet ouvrage, vous rencontrerez de plus en plus d'éléments de structure à inclure dans vos pages. Les quelques lignes précédentes vous permettent cependant de commencer à créer vos pages Web.

2.2.3 Identification des parties d'une balise

Jusqu'à présent, j'ai utilisé les termes balise et élément plus ou moins de manière interchangeable. Vous devez cependant noter qu'un élément n'est qu'une partie de la balise. Chaque balise comporte des attributs et les attributs ont des valeurs. Il est donc préférable de définir une balise dans les termes suivants :

- **Élément** : le nom
- **Attribut** : certaines caractéristiques de l'élément
- **Valeurs** : état ou condition de l'attribut

La figure 2.2 illustre les trois parties d'une balise.

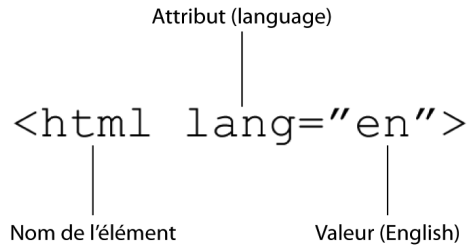


Figure 2.4 — Parties d'une balise.

Le nombre d'attributs diffère en fonction des éléments.

Selon l'élément, différentes sortes d'attributs sont disponibles et en fonction de l'attribut, on peut appliquer plusieurs types de valeur. En règle générale, on emploie des guillemets pour encadrer les valeurs, y compris les nombres. Voici quelques exemples :

```
<form action="http://localhost/php/phpversion.php" method="post">
<input type="text" width="120" hidden="false">
<input type="submit" value="A l'attaque">
```

Vous devez être très prudent dans le choix de ce que vous placez entre guillemets. Par exemple, `value="A l'attaque"` est autorisé car `'attaque` est un guillemet simple (apostrophe). Cependant la valeur `""A l'attaque," dit-il"` ne fonctionnerait pas car il y a deux paires de guillemets doubles.

Attribut language

L'attribut language (`lang`) n'est pas utilisé à moins de créer des pages dans une autre langue que l'anglais. La liste suivante fournit les valeurs d'attribut d'autres langues dans lesquelles vous pouvez écrire des pages Web :

- Allemand : `"de"`
- Arabe : `"ar"`
- Chinois (mandarin) : `"cmn"`
- Espagnol : `"es"`
- Français : `"fr"`
- Hébreu : `"he"`
- Hindi : `"hi"`
- Japonais : `"ja"`
- Portugais : `"pt"`
- Russe : `"ru"`

À la différence de certains attributs, l'attribut `lang` peut prendre un très grand nombre de valeurs. Pour obtenir une liste complète de ces valeurs, consultez la page `www.iana.org/assignments/language-subtag-registry`.

Un problème classique se produit quand vous avez une page qui contient une référence entre guillemets sur deux parties différentes de la page. Au sein d'un paragraphe, on peut placer autant de guillemets que l'on souhaite et ils s'afficheront sur la page. Cependant, il n'est possible d'assigner qu'une seule paire de guillemets à la valeur d'un attribut. Examinez le script suivant :

```
<!DOCTYPE HTML>
<html>
<head>
<title>Soyez attentif avec les guillemets</title>
</head>
<body>
<p>We read Emily Dickinson's "Wild nights! Wild nights!"<p/>
<input type="text" size="50" value="Emily Dickinson's 'Wild nights! Wild
nights!'">
</body>
</html>
```

Dans le conteneur `<p>`, les guillemets identifient le nom d'un poème. Si le même texte sert de valeur à un attribut, vous ne pouvez utiliser que des apostrophes (guillemets simples) pour délimiter le nom du poème. La figure 2.3 illustre l'affichage de la page dans un navigateur.

Quand vous assignez des valeurs à des attributs, n'oubliez pas d'employer des guillemets pour la totalité de la valeur et d'utiliser des apostrophes pour mettre en relief des sections à l'intérieur de la valeur. D'une manière générale, vous vous faciliterez la tâche si vous évitez d'utiliser des apostrophes quand vous assignez des valeurs à des attributs.

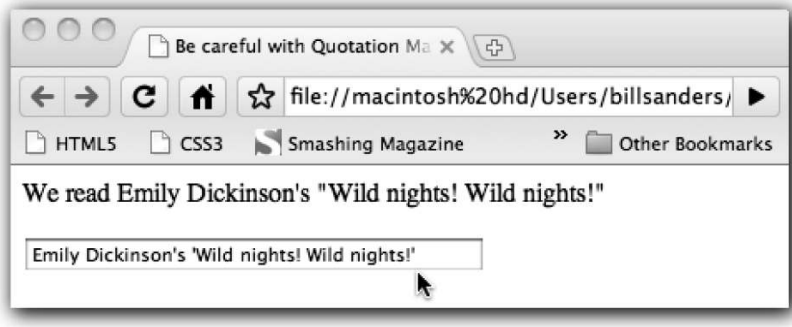


Figure 2.5 — Utilisation des guillemets dans les pages et les attributs HTML5.

2.2.4 Rôle de la balise de commentaire

Le rôle de la balise de commentaire est d'aider le développeur à communiquer avec les autres développeurs, mais aussi d'indiquer l'objectif de la page. Une page bien conçue contient des informations sur ce que fait la page, sur ce qui pourrait être ajouté ou modifié, ainsi que toute sorte d'information qui aide les développeurs examinant le script d'une page Web à voir de quoi il retourne rapidement.

La balise de commentaire se compose en réalité de deux balises : une balise de début et une balise de fin. À la différence des autres balises, la balise de commentaire ne comporte pas de texte qui permette de l'identifier. Le script suivant montre l'emplacement des commentaires et explique ce qu'ils font.

```
<!DOCTYPE HTML>
<html>
<head>
<title>Utilisation des commentaires dans votre code</title>
</head>
<body>
<h2>Les commentaires sont important</h2>
<!-- L'en-tête utilise un élément h2 au lieu d'un élément h1. -->
Les commentaires aident à vous souvenir et montrent aux autres votre projet de
conception de page.<br/>
Voici différentes utilisations :
<h5>1. Expliquer aux autres ce que vous faites.</h5>
<!--Cette page explique le rôle des commentaires.-->
<h5>2. Fournir des indications spécifiques sur les balises à utiliser.</h5>
<!--Ne pas utiliser de puces (<ul>). Nous n'avons pas appris
comment faire pour l'instant.-->
<h5>3. Lister les valeurs hexadécimales de votre palette de couleurs.</h5>
<!-- N'utiliser que les valeurs de couleurs suivantes sur
 cette page : 69675C,69623D,ECE8CF,E8D986,B5AA69 -->
<h5>4. Ne pas oublier de recharger l'ordinateur portable.</h5>
<!--Après avoir travaillé pendant deux heures, n'oubliez pas
de recharger votre batterie ! Sinon, vous risquez de tout perdre. -->
<h5>5. Vous faire garder à l'esprit qu'il y a une vie après l'ordinateur.</h5>
```

```
<!--N'oubliez pas votre rendez-vous avec Lola vendredi soir ! -->
</body>
</html>
```

Comme vous pouvez le voir quand vous chargez la page, aucun des commentaires n'est visible dans le navigateur, mais dès que vous retournez dans votre éditeur de texte, ils sont bien présents. Vous pouvez mettre n'importe quel texte dans un conteneur de commentaire et il n'apparaîtra pas.

On utilise souvent la balise de commentaire pour supprimer temporairement des balises que l'on utilisera plus tard. Ainsi, au lieu de supprimer les balises, il vous suffit de les encadrer par des balises de commentaire et de tester votre page pour voir si vous la préférez sans les balises en question. Si vous pensez que le résultat est meilleur dans sa forme originale, il n'y a qu'à supprimer les balises de commentaire. Si la page paraît mieux sans les balises mises en commentaire, vous pouvez supprimer définitivement les balises devenues inutiles.

Supposons, par exemple que vous vous posiez la question de savoir si une page que vous concevez pour un client est plus alléchante avec un sous-titre et une note de bas de page. Voici le code original code avec le sous-titre :

```
<!DOCTYPE HTML>
<html>
<head>
<title>Mise en commentaire</title>
</head>
<body>
<header>
  <h1>Restaurant Chez Joe</h1>
  <h2>*A survécu a toutes les inspections sanitaires depuis 2005</h2>
</header>
<section>
Joe cuisine les meilleurs repas du quartier ! La nourriture est délicieuse et
bon marché !
</section>
<footer>
<h6>*La table du bobo, édition 2010</h6>
</footer>
</body>
</html>
```

La figure 2.4 illustre le résultat dans un navigateur.

Après avoir réfléchi à la conception, vous suggérez au propriétaire du restaurant, qui est assez fier de sa critique gastronomique, que le message commercial serait peut-être mieux perçu si le sous-titre et la note étaient supprimés. Mais au lieu de supprimer complètement les balises, vous les mettez simplement en commentaire de la manière suivante :

```
<!DOCTYPE HTML>
<html>
<head>
```

Figure 2.6 — Conception originale.

```
<title>Mise en commentaire</title>
</head>
<body>
<header>
  <h1>Restaurant Chez Joe</h1>
  <!-- <h2>*A survécu a toutes les inspections sanitaires depuis 2005</h2> -->
</header>
<section>
Joe cuisine les meilleurs repas du quartier ! La nourriture est délicieuse et
bon marché !
</section>
<footer>
<!-- <h6>*La table du bobo, édition 2010</h6> -->
</footer>
</body>
</html>
```

Quand utiliser (et ne pas employer) des balises de commentaire

En général, on n'utilise pas suffisamment les commentaires dans une page Web. Parfois, quelques commentaires suffisent, et si une page n'en a besoin que de quelques-uns, il est inutile d'en rajouter. D'autres fois, la page a besoin de beaucoup plus de commentaires qu'elle n'en contient. Le nombre de commentaires nécessaires dépend totalement de la taille et de l'ampleur du projet Web et du nombre de développeurs qui y sont impliqués.

Il arrive cependant que certains développeurs se laissent emporter et produisent tellement de commentaires que le code HTML devient illisible. Une page comportant un long commentaire après chaque balise est contre-productive et nuit à la lecture du code. Si une page nécessite un grand nombre de commentaires, placez-les tous dans un seul conteneur, ce qui permettra ensuite aux autres développeurs de voir le code HTML et de comprendre la manière dont il est utilisé.

Une fois que vous avez effectué les changements en mettant en commentaire les balises non souhaitées, vous pouvez apprécier le résultat (voir la figure 2.5).

Figure 2.7 — Page avec le code mis en commentaire.

Si le client préfère quand même la version originale, il vous suffit de supprimer les balises de commentaires pour retrouver la page dans son état antérieur. Si vous voulez tester plusieurs apparences différentes, vous pouvez aussi utiliser la balise de commentaire.

2.3 BALISES IMBRIQUÉES

Quand on crée une page HTML, on peut imbriquer les balises, c'est-à-dire que l'on peut placer un conteneur HTML5 à l'intérieur d'un autre conteneur. En fait, j'ai déjà fait cela dans ce livre. Voici la règle à respecter : ajoutez une balise de fin à l'intérieur d'un conteneur avant la balise de fin du conteneur. Si vous écrivez une balise à l'intérieur d'un autre conteneur de balise, assurez-vous de fermer le conteneur interne avant de fermer le conteneur externe. Les exemples suivants vous permettront de mieux comprendre :

Dans l'exemple suivant, la balise `<h1>` ferme le conteneur externe `<section>` :

```
<section>
<h1>Génial !
</section>
</h1>
```

Mais elle devrait être comme ceci :

```
<section>
<h1>Génial !</h1>
</section>
```

Ici, la balise `<body>` ferme le conteneur externe `<html>`. Le conteneur `<h3>` est correct.

```
<html>
<body>Vraiment intéressant
<h3>N'oubliez pas de voter !</h3>
</html>
</body>
```

Voici la bonne version :

```
<html>
     <body> Vraiment intéressant
          <h3> N'oubliez pas de voter!</h3>
     </body>
</html>
```

Ici, la balise `<header>` est fermée avant que la balise `<nav>` ne soit fermée :

```
<header>
<nav>
<a href="html5.org">HTML5</a>  |  
<a href="css3.org">CSS3</a>>  |  
<a href="php.net">PHP</a>
</header>
<footer>
<a href="html5.org">HTML5</a>  |  
<a href="css3.org">CSS3</a>>  |  
<a href="php.net">PHP</a>
</nav>
</footer>
```

À la place, il faut utiliser deux ensembles de conteneurs `<nav>` : un pour le header et un autre pour le footer :

```
<header>
<nav>
<a href="html5.org">HTML5</a>  |  
<a href="css3.org">CSS3</a>  |  
<a href="php.net">PHP</a>
</nav>
</header>
<footer>
<nav>
<a href="html5.org">HTML5</a>  |  
<a href="css3.org">CSS3</a>  |  
<a href="php.net">PHP</a>
</nav>
</footer>
```

Parfois, quand on teste une page HTML5, on a un résultat imprévu, voire rien du tout. En ce cas, la première chose à vérifier est l'imbrication des balises.

Si vous vous interrogez sur le sens du code ` `, il s'agit d'un espace insécable (le point-virgule fait partie de la balise). Vous pouvez vous représenter simplement ce

caractère comme un espace qui encadre la barre verticale (|) utilisée pour séparer les liens. Dans votre navigateur, vous verrez :

HTML5 | CSS3 | PHP

Quand on place du code à l'intérieur de balises `<nav>`, on peut facilement l'identifier comme du code de navigation. Cependant, comme toutes les autres balises, vous devez faire attention aux règles d'imbrication utilisées en HTML5.

2.4 À VOUS DE JOUER !

La page HTML suivante comporte des erreurs qui ont besoin d'être corrigées. Elle commence par plusieurs balises qui sont vides ou partiellement complètes. Là où cela est nécessaire, vous devez ajouter les bonnes balises ou compléter le texte. Parfois, vous devrez fermer un conteneur qui a été ouvert (`<balise>`) ou bien en ouvrir un qui a été fermé (`</balise>`). Et assurez-vous que les balises sont correctement imbriquées (**Indice** : la première balise n'est pas une balise HTML, mais une balise spéciale qui commence par un point d'exclamation !).

```
<!     >
<html lang=  >
<head>
<!---Palette de couleurs
0B0B0D,29272A,A99A93,E27107,F8AC00 -->
<style type="text/css">
body {
background-color:#F8AC00;
color:#29272a;
font-family:Verdana, Geneva, sans-serif;
font-size:12px;
margin-left:20px;
}
h1 {
color:#29272A;
font-family:"Arial Black", Gadget, sans-serif;
}
h2 {
text-indent:10px;
color:#0B0B0D;
background-color:#E27107;
font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;
}
header {
text-align:center;
}
</style>
<title>==???===</title>
<    >
<body>
<header>
  <  >Mes préférences</h1>
</header>
```

```
<section>
  <h2>Ma musique préférée</h2>
  ==????==<br/>
  ==????==<br/>
  ==????==<br/>
  < >Mes films préférés</h2>
  ==????==<br/>
  ==????==<br/>
  ==????==<br/>
  <h2>Mes ordinateurs préférés</h2>
  ==????==<br/>
  ==????==<br/>
  ==????==<br/>
  <h2>Mes émissions préférées</h2>
  ==????==<br/>
  ==????==<br/>
  ==????== <br/>
  <    >
<   >
  <h5>Je ne suis pas responsable de mes goûts. C'est à prendre ou à laisser. <
>
</footer>
</body>
</html>
```

Cet exercice devrait vous aider à faire attention aux détails dans lesquels, comme chacun le sait, le diable se niche...

3

Balises de texte et un peu de CSS3

Objectif

Une page Web est différente des pages que vous saisissez dans votre traitement de texte. Les pages Web sont conçues pour des écrans d'ordinateurs, qu'il s'agisse d'un ordinateur de bureau, d'un portable ou même d'un terminal mobile. Une page Web ne s'affiche pas sur une feuille de papier au format A4, mais sur un dispositif d'affichage bien plus dynamique. La première chose à laquelle il faut donc penser est l'aspect de la page sur un écran numérique.

3.1 PRINCIPES DE BASE

Avant de pouvoir gérer du texte sur une page Web, il faut prendre en considération ses éléments fondamentaux qui incluent trois types d'actions :

- Affichage de texte,
- Chargement et affichage d'images,
- Liens vers d'autres pages.

Pour afficher du texte, il suffit de le saisir sur la page dans le conteneur. Vous pouvez le styler avec la balise `<h>` comme nous l'avons vu dans les chapitres précédents, mais le texte de base a seulement besoin d'être présent au sein d'une balise `<body>`.

Le chargement et l'affichage des images utilisent la balise `<img>` avec le format suivant :

```
<img src=''nomImage.png''>
```

Vous pouvez uniquement utiliser les fichiers `.jpg`, `.png` ou `.gif` avec l'élément `img`. L'attribut `src` fait référence à la source de l'image. L'élément `img` a d'autres attributs, mais le seul attribut indispensable pour afficher une image sur la page est `src` pour que le fichier puisse être localisé.

Dans cet ouvrage, le terme URL est souvent employé pour faire référence à l'emplacement d'un fichier, quel que soit le type de fichier impliqué. URL, qui est l'acronyme d'*Uniform Resource Locator* (localisateur de ressource uniforme), est un protocole standard pour trouver et utiliser différents types de fichiers.

Enfin, un lien vers une autre page utilise le format suivant :

```
<a href=''autrePage.html''>Lien vers une page</a>
```

La balise `href` renvoie à la référence hypertexte de la page chaînée, ou exprimé en termes plus simples, à son adresse. Comme pour l'emplacement de la source d'une image, vous verrez que le terme URL est aussi utilisé pour l'adresse du lien.

Avant de continuer, vous devez encore savoir une autre chose : la déclaration du type de document (`<!DOCTYPE HTML>`) de la toute première ligne est importante et il ne faut pas l'oublier. Il ne faut cependant pas non plus oublier la déclaration du type d'encodage des caractères. Il est utilisé pour indiquer au navigateur Web le jeu de caractères qu'il doit employer, comme l'alphabet de A à Z, les caractères hébreux, les caractères japonais, les caractères cyrilliques ou tout autre jeu de caractères. Il existe plusieurs manières de définir le jeu de caractères, mais ce livre utilise le format suivant :

```
<meta http-equiv=''Content-Type'' content=''text/html; charset=UTF-8''>
```

Vous devez toujours spécifier l'encodage des caractères. Bien que l'utilisation de la balise `<meta>` soit un peu contraignante, vous pouvez copier cette ligne et la coller dans toutes vos pages Web. Si vous ne le faites pas, vous vous exposez à des problèmes de sécurité, ce qui n'est bien entendu pas souhaitable.

Il est toujours possible de bricoler à la hâte une page Web, mais le résultat est souvent décevant et les visiteurs de votre site risquent de ne jamais y revenir. Examinons une page Web sans structure mais qui en contient cependant les éléments fondamentaux :

```
<!DOCTYPE HTML>
<html>
     <head>
<meta http-equiv=''Content-Type'' content=''text/html; charset=UTF-8''>
          <title>Principes de base</title>
     </head>
     <body>
     Ceci est du texte. Il n'y a pas besoin de balise pour écrire du texte.
```

```
    <a href=''autrePage.html''> Cliquez ici pour une autre page </a>
    <img src=''logo.png''>
    </body>
</html>
```

Comme vous pouvez le voir à la figure 3.1, tout est en désordre. L'image apparaît au milieu de la page, au-dessus du lien (texte souligné en bleu), ce qui n'est guère conforme aux usages.

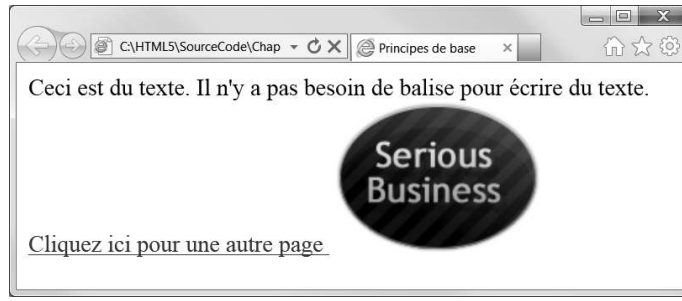


Figure 3.1 — Éléments de base d'une page Web.

Quand on crée une page Web, les liens doivent être organisés selon un système de navigation qui est censé faciliter la consultation du site. Dans la page illustrée à la figure 3.1, le lien qui est perturbé par l'image semble faire partie du texte plutôt que d'un système de navigation.

3.1.1 Un peu plus d'organisation

Une des conventions de base de la conception Web est de placer le logo dans le coin supérieur gauche de la page. De plus, les pages Web placent les liens de manière organisée selon un système cohérent de navigation. En ajoutant deux balises de plus, vous pouvez améliorer l'organisation de vos pages :

- `<br>` : Génère un saut de ligne
- `<wbr>` : Génère un saut de ligne conditionnel

Un saut de ligne (`<br>`) force un retour à la ligne du texte. Vous pouvez vous le représenter comme un espace unique entre les lignes, ou si vous êtes de la vieille école, comme un retour de chariot. HTML5 a ajouté quelque chose de nouveau qui s'appelle opportunité de saut de ligne. Il arrive que l'on ait de très longs mots, notamment dans les URL ou les adresses électroniques. L'élément `wbr` ne force pas de saut de ligne, mais vous pouvez placer la balise `<wbr>` à l'endroit où vous souhaiteriez que le mot soit coupé au cas où il devrait l'être si la page est redimensionnée. Cette prise en compte est particulièrement importante pour les terminaux mobiles car ils ont des écrans de petite taille. Par exemple, supposons que vous ayez une très longue URL qui est affichée sous cette forme :

```
www.chezjoerestaurantgastronomiquepanoramique.com
```

Si le lien n'est pas découpé, et si la page est redimensionnée, vous aurez un grand trou dans le texte ou bien le mot sera découpé à un endroit qui n'est pas souhaitable. La balise `<wbr>` vous aide à découper le texte là où vous le souhaitez. Examinez le script suivant (`SautsDeBase.html` dans les fichiers de ce chapitre), qui utilise les deux balises de saut de ligne :

```
<!DOCTYPE HTML>
<html>
     <head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
     <title>Ajout de sauts de ligne</title>
     </head>
     <body>
<img src="logo.png"><br>
     Ceci est du texte. Il n'y a pas besoin de balise pour écrire du texte.<br>
     <br>
Il dit, "Il y a parfois de très longs mots dont on souhaite s'assurer qu'ils
sont coupés au bon endroit. Par exemple, si vous avez une très longue URL
comme
www.chez<wbr>joe<wbr>restaurant<wbr>gastronomique<wbr>panoramique<wbr>.com, et
que vous êtes obligé de découper l'URL, il est souhaitable que la coupure ne se
fasse pas n'importe comment."<br><br>
Il dit, "Il y a parfois de très longs mots dont on souhaite s'assurer qu'ils
sont coupés au bon endroit. Par exemple, si vous avez une très longue URL
comme www.chezjoerestaurantgastronomiquepanoramique.com, et que vous êtes
obligé de découper l'URL, il est souhaitable que la coupure ne se fasse pas
n'importe comment."<br><br>
     <a href="autrePage.html"> Cliquez ici pour une autre page </a>
     </body>
</html>
```

En ajoutant les deux balises de saut de ligne, la page a un bien meilleur aspect. Le paragraphe qui n'utilise pas la balise `<wbr>` a un trou à l'emplacement où l'URL longue n'a pas été découpée. La figure 3.2 illustre l'aspect de la page.

Bien que cela ne soit toujours pas parfait, c'est déjà bien mieux que la page originale, même si deux nouveaux paragraphes ont été ajoutés. L'image se situe dans le coin supérieur gauche (comme c'est le cas pour la plupart des logos), les paragraphes sont séparés par des sauts de ligne et dans le premier paragraphe qui emploie une URL longue, les sauts s'effectuent aux endroits spécifiés par la balise `<wbr>`.

3.1.2 Réfléchir à la structure

Il est désormais temps d'entamer une réflexion sur la structure de vos pages Web. En ajoutant du texte, des images et des liens, la page gagne en fonctionnalités et en contenu. Vous devez donc commencer à réfléchir à des choses comme les titres, la navigation et le positionnement (qui va bien au-delà du logo dans le coin gauche). Commencez par établir une maquette simple. Utilisez une feuille pour coucher sur le papier vos idées et réaliser un croquis de votre page Web (prenez pour l'instant du papier et non pas un logiciel graphique). La figure 3.3 illustre un exemple.

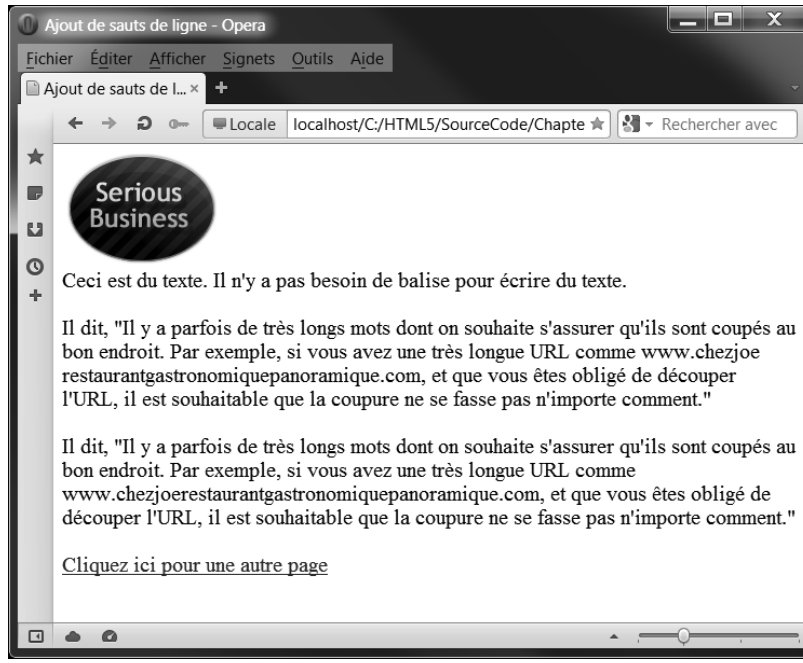


Figure 3.2 — Ajout de sauts de ligne.

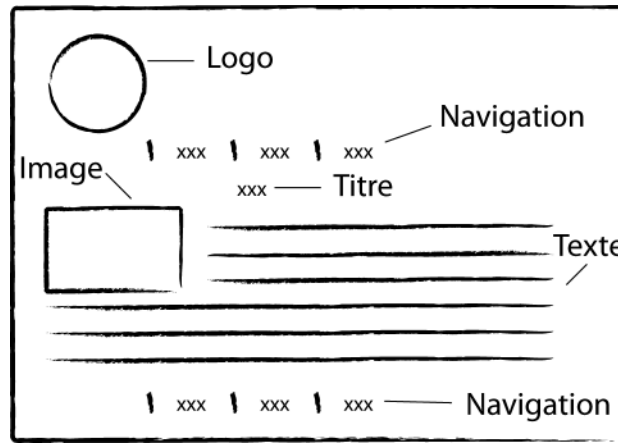


Figure 3.3 — Croquis de la structure de votre site.

En vous basant sur les balises que nous avons étudiées jusqu'à présent, serez-vous capable de créer une page à partir de votre croquis ? Le seul attribut qui manque est celui qui permet d'habiller le texte autour de l'image. L'attribut `align` de l'élément `img` réalisera cela. Dans cet exemple, l'image sera à gauche et le texte sera à droite, si bien que la ligne suivante fera l'affaire :

```
<img src=''kid.png'' alt=''kid'' align=''left''>
```

Vous avez peut-être remarqué que l'attribut `alt` a également été inclus. Cet attribut permet aux utilisateurs de savoir à quoi s'attendre si l'image met du temps à se charger.

Ainsi, avec quelques balises et l'ajout d'un attribut, le prochain script crée parfaitement une page qui correspond à la structure du croquis réalisé à la figure 3.3.

Comme vous le constaterez dans le code suivant (`Maquette2Web.html` disponible dans les fichiers de ce chapitre), j'ai utilisé un signe dièse (#) à la place d'une véritable URL dans les liens de navigation. Le signe dièse joue le rôle d'un emplacement réservé quand vous travaillez sur la structure ; il fonctionne exactement comme une véritable URL sauf qu'il ne va nulle part et ne provoque pas de message d'erreur.

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>De la maquette à la page Web</title>
</head>
<body>
<img src="logo.png" alt="logo"><br>
<a href="#">Jouets</a>  | <a href="#">Vêtements</a> | <a
href="#">Sports</a> <br>
<br>
Le paradis des enfants<br>
<br>
<img src="kid.png" alt="kid" align="left"> Les enfants, c'est une affaire
sérieuse. Ils ont besoin de jouets qui soient à la fois sûrs et éducatifs. Les
jouets doivent être amusants tout en développant la créativité des enfants. Il
n'y a pas de raison de sacrifier la sécurité pour privilégier l'amusement. Les
enfants ont besoin de beaucoup d'habits car ils grandissent si rapidement. Et
ils ont besoin de pratiquer une activité physique pour combattre l'obésité et
toutes les maladies qui y sont liées. <br>
<br>
<a href="#">Jouets</a>  | <a href="#">Vêtements</a> | <a
href="#">Sports</a>
</body>
</html>
```

Notez que nous n'avons utilisé aucun des éléments `H` évoqués dans les deux précédents chapitres. Cela est dû au fait que je les traite de manière détaillée dans la prochaine section. La figure 3.4 illustre bien le fait que la page se rapproche étroitement du croquis de la figure 3.3.

À présent, la page Web illustrée à la figure 3.4 a plus de structure que n'importe lequel des exemples précédents. Les barres de navigation au sommet et en bas de la page sont pratiques pour l'utilisateur, mais elles mériteraient peut-être d'être centrées sur la page. La barre de navigation supérieure devrait être centrée au sommet de la page, juste à côté du logo. Le texte qui est coincé contre l'image pourrait être un peu décalé et, bien entendu, le titre devrait être dans un style différent (taille et police de caractères). Le site est également assez ennuyeux, ce qui est paradoxal étant donné

Figure 3.4 — Structure de base d'une page. © David Sanders

qu'il est conçu pour des enfants. Mais comme la structure est en constante progression, vous pourrez régler tous ces détails au fur et à mesure que vous apprendrez à utiliser d'autres techniques pour améliorer le style.

3.2 RENFORCEMENT DE LA STRUCTURE HTML5

Dans la section précédente, nous avons étudié l'élément `wbr` qui est nouveau en HTML5. Cette section va examiner en détail l'utilisation des balises familières `<h..>` et de certaines balises associées qui servent à structurer du texte. Nous avons vu comment démarrer en créant à la main une ébauche de la page souhaitée pour ensuite l'implémenter dans un script HTML5. Le fait de partir d'une ébauche concrète pour passer à un plan d'ensemble des blocs facilite la compréhension de la manière dont HTML5 est organisé en blocs. Nous commencerons par examiner le bloc de texte, sujet que nous avons déjà abordé dans les deux premiers chapitres avec les balises `<h1>`, `<h2>`, et les autres éléments `h`. La figure 3.5 illustre l'organisation en blocs.

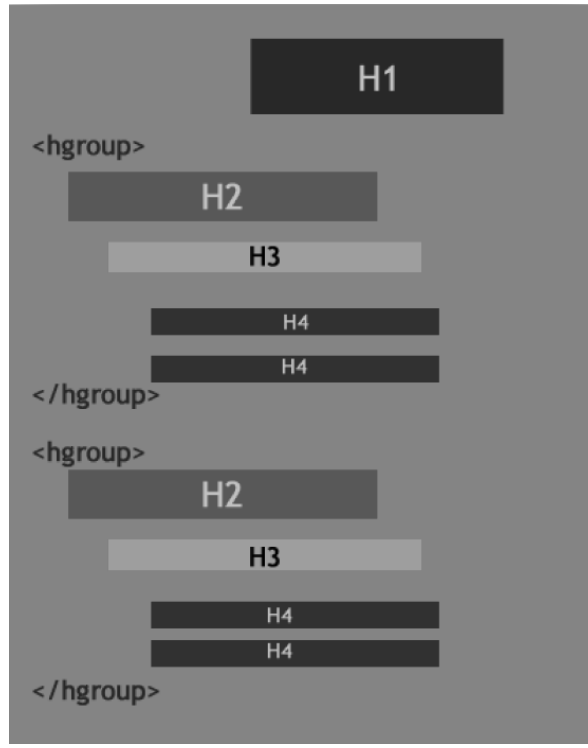


Figure 3.5 — Organisation des blocs de texte.

En termes d'organisation de la page, la mise en page des différents niveaux d'éléments `h` est réalisée par la balise HTML5 `<hgroup>`. Examinez par exemple la page Web suivante (`HelementOrg.html` disponible dans les fichiers du chapitre) qui reprend un texte du philosophe Wittgenstein (qui semblait écrire avec des balises `h` en 1918 quand il a rédigé le *Tractatus Logico-Philosophicus*) :

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv=''Content-Type'' content=''text/html; charset=UTF-8''>
<title>Tractatus logico-Philosophicus</title>
</head>
<body>
<h1>Tractatus logico-Philosophicus</h1>
<h1>by Ludwig Wittgenstein</h1>
<hgroup>
<h2>1 The world is all that is the case.</h2>
<h3>1.1 The world is the totality of facts, not of things.</h3>
<h4>1.11 The world is determined by the facts, and by their being all the
facts.</h4>
<h4>1.12 For the totality of facts determines what is the case, and also
whatever is not the case.</h4>
<h4>1.13 The facts in logical space are the world.</h4>
```

```
<h3>1.2 The world divides into facts.</h3>
<h4>1.21 Each item can be the case or not the case while everything else
remains the same.</h4>
</hgroup>
</body>
</html>
```

Si l'on regarde la page Web, on peut voir où les différents éléments `h` attribuent des tailles différentes à chaque partie, mais on ne voit pas les indentations que Wittgenstein a utilisées dans son manuscrit original. La figure 3.6 illustre la page Web sur un téléphone mobile (quoi que vous pensiez de Wittgenstein, son style fonctionne à merveille sur les écrans des terminaux mobiles).

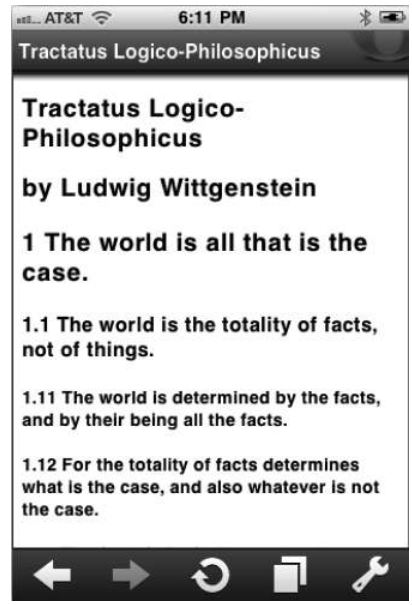


Figure 3.6 — Mise en page d'un plan à l'aide de balises `<h>` sur iPhone.

Si vous examinez le manuscrit original de Wittgenstein, vous constaterez que son style d'écriture utilisait un plan indenté qui apparaissait sous la forme suivante :

1 The world is all that is the case.

1.1 The world is the totality of facts, not of things.

1.11 The world is determined by the facts, and by their being all the facts.

1.12 For the totality of facts determines what is the case, and also whatever is not the case.

On peut régler cela si l'on veut en ajoutant des indentations aux balises `<h..>`. On pourrait faire ça en ajoutant des marges à l'aide de CSS3 comme vous le verrez dans la section suivante. Le but des éléments `h` et `<hgroup>` n'est cependant pas de définir des indentations, mais de faciliter la structuration en niveaux hiérarchiques. La balise

`<hgroup>` définit la balise `<h..>` de niveau le plus élevé dans le conteneur `hgroup` en tant qu'élément de plan. Par exemple, puisque Wittgenstein a entièrement hiérarchisé le *Tractatus Logico-Philosophicus*, l'utilisation de l'élément `hgroup` sur l'œuvre complète produirait exactement le plan qui figure dans le résumé de l'œuvre.

1 The world is all that is the case.

2 What is the case — a fact — is the existence of states of affairs.

3 A logical picture of facts is a thought.

4 A thought is a proposition with a sense.

5 A proposition is a truth-function of elementary propositions. (An elementary proposition is a truth-function of itself.)

6 The general form of a truth-function is [p, E, N(E)]. This is the general form of a proposition.

7 What we cannot speak about we must pass over in silence.

L'élément `hgroup` est lié à l'algorithme de hiérarchisation de HTML5, et bien qu'il soit peu probable que vous l'utilisiez pour des écrivains tels que Wittgenstein, il est utile pour vous aider à réfléchir à votre page en termes de structure au sein d'une page HTML5. Vous pouvez vous représenter la balise `<hgroup>` comme un masque qui couvrirait d'autres éléments `h` qui seraient dessous l'élément de niveau le plus élevé dans le conteneur `hgroup`. Dans notre exemple, les éléments `h3` et `h4` ne sont masqués que si l'élément `h2` est reconnu comme faisant partie du plan.

3.3 AJOUT DE STYLE À UN TEXTE À L'AIDE DE CSS3

Dans cet ouvrage, quand nous ferons référence à la norme Cascading Style Sheet, il s'agira de CSS3. Cela est dû au fait que HTML5 et CSS3 ont des relations étroites, mais comme pour les autres éléments étudiés dans ce livre, il y a un héritage qui a été intégré dans les versions les plus récentes de HTML et CSS. Tout comme en HTML5, il y a un mélange d'ancien et de moderne dans CSS3. Dans ces conditions, si vous êtes familier des anciennes versions de CSS et que vous voyez les mêmes propriétés en CSS3, considérez qu'il s'agit d'une fonctionnalité qui a été maintenue.

3.3.1 Stylage des éléments HTML5 avec des propriétés CSS3

Dans les chapitres 1 et 2, vous avez vu des exemples de CSS3, mais je ne vous ai fourni aucune explication sur la manière d'ajouter un nouveau style à un élément existant. Nous allons ici mettre l'accent sur l'ajout de style aux éléments `h`. Dans les trois chapitres suivants, nous irons bien plus loin dans l'utilisation de CSS3, mais pour l'instant nous allons étudier les bases de l'incorporation de CSS3 dans HTML5.

Toute feuille de styles peut être ajoutée de trois manières différentes :

- Vous pouvez utiliser la balise `<style>` pour définir les propriétés des éléments de la page HTML5.

- Vous pouvez utiliser des feuilles de styles externes qui sont des fichiers texte où l'on stocke des styles que l'on veut réutiliser.

La plupart des développeurs et des concepteurs professionnels préfèrent les feuilles de styles CSS3 externes car la mise au point des styles prend beaucoup de temps. Quand on veut faire une modification de la conception d'un site Web, on peut réaliser la modification de plusieurs pages qui utilisent une feuille de styles externe en changeant uniquement la feuille de styles. C'est bien plus efficace que d'avoir à modifier les attributs `<style>` de chaque page Web individuelle.

Vous pouvez aussi ajouter des styles sans recourir à une feuille de styles en utilisant des styles en ligne. Un style en ligne est similaire à la technique « Brisez la glace en cas d'urgence ! ». Une belle page a forcément une feuille de styles intégrée, mais il arrive parfois que des changements doivent s'opérer en urgence et au lieu de modifier la feuille de styles, on crée le style avec une balise.

Feuilles de styles incorporées

Une feuille de styles incorporée n'est que l'ajout d'une feuille de styles directement dans le script HTML5. Dans la balise `<head>` de la page, ajoutez la feuille de styles avec un conteneur `<style>`. Placez l'élément que vous voulez styler dans le conteneur de style, puis ajoutez les valeurs à la propriété à styler. La figure 3.7 indique le format général.

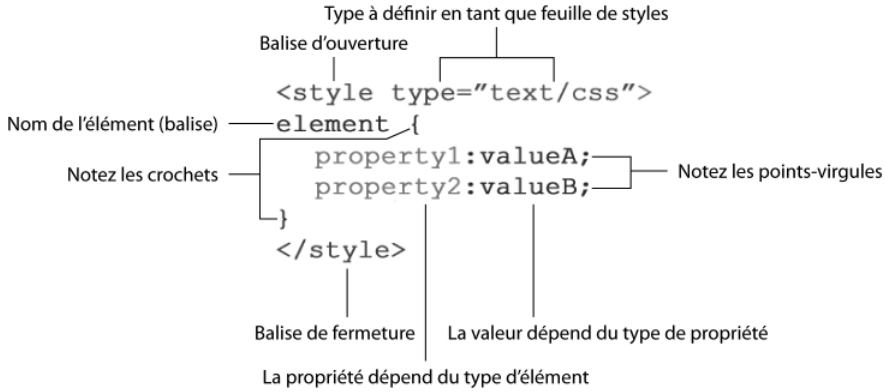


Figure 3.7 — Feuille de styles incorporée.

Chaque élément a un ensemble unique de propriétés et chaque propriété a des valeurs qui peuvent être assignées. Quand on change la valeur de la propriété, cette valeur apparaît dans le texte à l'intérieur du conteneur de l'élément. Si vous modifiez la couleur du texte en rouge, alors tout le texte à l'intérieur du conteneur de l'élément sera affiché en rouge. Le script suivant (`CSS3polices.html` disponible dans les fichiers du chapitre) propose un exemple.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
body {
background-color:#fbf7e4;
font-family:Verdana, Geneva, sans-serif;
margin-left:20px;
color:#8e001c;
}
h1 {
background-color:#8E001C;
color:#e7e8d1;
font-family:"Arial Black", Gadget, sans-serif;
text-align:center;
}
h2 {
background-color:#424242;
color:#d3ceaa;
font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;
margin-left:5px;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>CSS3-Feuille de styles incorporée</title>
</head>
<body>
<h1>Ceci est un grand titre</h1>
<h2> Voici le deuxième titre</h2>
Le corps du texte est stylé avec une marge gauche et une police particulière
qui est en couleur. Vous noterez que la couleur de fond du titre s'étend sur
toute la largeur de la page. Vous remarquerez aussi qu'un espace insécable (&
nbsp;) donne au deuxième titre, h2, une légère indentation de telle sorte qu'il
reste "à l'intérieur" de la couleur de fond. On n'a pas ce problème avec le
titre h1 car il est centré.
</body>
</html>
```

La figure 3.8 illustre le résultat de la page stylée.

Vous devez être conscient que quand on utilise des feuilles de styles, on doit faire attention aux petits détails, comme ajouter deux accolades, séparer la propriété de ses valeurs avec le caractère deux-points, et terminer chaque valeur de propriété par un point-virgule. Si votre feuille de styles CSS3 ne fonctionne pas comme elle le devrait, vérifiez tous ces petits détails de syntaxe.

Quand on utilise des couleurs de fond, elles se propagent souvent sur la totalité de la page. Certains éléments en ligne comme `<span>` peuvent être utilisés pour contenir la couleur de fond dans le texte qui est stylé. Avec les couleurs de fond qui sont justifiées à gauche ou à droite, il faudra ajouter un espace insécable (` `) pour que la couleur de fond ne déteigne pas sur toute la page.

Figure 3.8 — Texte stylé avec CSS3.

Feuilles de styles externes

Avec toutes les combinaisons de styles différents (auxquelles il faut ajouter les variations des formats d'écran des ordinateurs, des portables et des terminaux mobiles), vous devez prendre en compte le fait que la création d'une bonne feuille de styles ou d'un ensemble de feuilles de styles peut représenter un travail considérable. En enregistrant votre création CSS3 dans un fichier texte, vous pouvez réutiliser votre feuille de styles aussi souvent que vous le souhaitez. De plus, vous pouvez copier vos CSS incorporées et les coller facilement dans un fichier texte pour l'enregistrer dans un fichier `.css`.

Prenons par exemple une palette de couleurs avec un jeu de couleurs qui fait partie de la charte graphique d'un de vos clients, Mighty Smite Web Development (cela a pour conséquence que vous ne pouvez utiliser que le jeu de couleurs qui vous a été fourni). Vous commencez avec les couleurs suivantes qui font partie de la charte graphique de l'entreprise :

```
#3C371E, #8C5F26, #BCA55F, #F2CC6E, #F26205
```

La couleur de fond doit être `#F2CC6E`. Vous n'avez pas à savoir ce qu'est cette couleur et vous devez vous contenter de savoir que l'entreprise a décidé que ce serait la couleur de fond, les concepteurs s'occupant du reste.

On vous a de plus signifié que l'entreprise souhaitait une version qui s'affiche à la fois sur un téléphone et sur un ordinateur de bureau. Cela signifie que vous allez avoir besoin de deux feuilles de styles différentes. Vous vous préoccuperez plus tard de savoir comment le navigateur sait que l'utilisateur affiche la page sur un ordinateur ou sur un téléphone.

Seule la balise suivante est nécessaire :

```
<link rel=''stylesheet'' type=''text/css'' href=''mightySmiteSmall.css'' />
```

Cette balise va dans le conteneur `<head>` où la balise `<style>` allait avec le code CSS3. Le code CSS3 se trouve à présent dans un fichier séparé. Vous noterez que la balise `<link>` contient un attribut `href` avec la valeur `mightySmiteSmall.css`. Il s'agit du nom du fichier CSS3 (que vous trouverez dans les fichiers de ce chapitre). La mention `Small` indique qu'elle est conçue pour les terminaux mobiles. Il faudra créer un autre fichier CSS3, `mightySmiteLarge.css`, pour les ordinateurs qui ne sont pas des terminaux mobiles.

Pour créer un fichier CSS3, il suffit de saisir le code CSS3 dans un éditeur de texte ou dans une application de développement Web et de supprimer les balises `<style>`. Le code suivant illustre l'exemple que nous allons utiliser :

```
@charset ''UTF-8'';
/* CSS Document */
/*3C371E,8C5F26,BCA55F,F2CC6E,F26205 */
body
     {
          background-color:#F2CC6E;
         font-family:''Lucida Sans Unicode'', ''Lucida Grande'', sans-serif;
          color:#8C5F26;
         font-size:11px;
          max-width:480px;
     }
h1
     {
          color:#BCA55F;
         background-color:#3C371E;
         font-family:''Arial Black'', Gadget, sans-serif;
          text-align:center;
     }
h2
     {
          color:#F26205;
         font-family:''Lucida Sans Unicode'', ''Lucida Grande'', sans-serif
     }
h3
     {
          color:#3C371E;
         font-family:Tahoma, Geneva, sans-serif;
     }
```

La première ligne indique au navigateur qu'il s'agit d'un jeu de caractères UTF-8 et les deux lignes suivantes sont des lignes de commentaires. Elles diffèrent des lignes de commentaires en HTML5, mais elles fonctionnent de manière identique. La deuxième ligne de commentaire est un moyen pratique de mémoriser la palette de couleurs et peut gagner du temps lors de la définition de la feuille de styles.

Pour tester cette version mobile du code CSS3, nous utilisons le fichier HTML5 `ExternalSmall.html` qui est disponible avec les fichiers de ce chapitre :

```
<!DOCTYPE HTML>
<html>
<head>
```

```
<link rel=''stylesheet'' type=''text/css'' href=''mightySmiteSmall.css'' />
<meta http-equiv=''Content-Type'' content=''text/html; charset=UTF-8''>
<title>Mighty Smite Sofware Test Sheet</title>
</head>
<body>
<h1>Mighty Smite Software Conglomorate</h1>
<h2>This is an h2 head</h2>
<h3>Here's an h3 head</h3>
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut abore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco aboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est aborum.
</body>
</html>
```

Tous les styles du fichier CSS3 sont utilisés afin de tester leur apparence et le corps du texte commençant par Lorem ipsum est un texte de remplissage qui est utilisé pour avoir une idée de l'aspect des blocs de texte (ça doit être un bon texte puisqu'on l'utilise depuis le seizième siècle).

Dans la définition du fichier CSS3, le seul paramètre qui cible spécifiquement les terminaux mobiles est l'attribut `width` de l'élément `body`. Il est paramétré à 480 px car c'est la largeur de l'écran de l'iPhone qui est utilisé pour les tests. Cependant, en fonction de la manière dont les utilisateurs tiennent en main leur téléphone, ils peuvent obtenir des résultats différents. Les figures 3.9 et 3.10 illustrent l'apparence de la page quand le téléphone est tenu dans des positions différentes.

Différences des densités de pixels

Quand on crée des pages Web pour des écrans dont la taille varie de 30 pouces à 3 pouces, on ne peut pas se contenter de prendre seulement en compte le nombre de pixels sur les plans vertical et horizontal. Dans l'exemple de feuille de styles externe CSS3, la largeur est définie à 480 par le code `max-width:480px;` pour un iPhone dont la résolution horizontale est de 480 pixels. Cependant, quand on exécute l'application sur un téléphone mobile, le texte peut apparaître soit trop gros soit trop petit. Que se passe-t-il ?

On a tendance à penser les résolutions d'écran en termes de nombre de pixels (plus il y a de pixels, meilleure est la résolution). Si vous définissez votre écran en 1680 x 1050, il a par conséquent une résolution plus élevée que si vous le paramétrez à une résolution de 1024 x 768. Cependant, la résolution dépend en fait du nombre de pixels par rapport à la taille de la zone d'affichage. La densité des pixels, qui est le nombre de pixels de la zone d'affichage et qui se mesure en pixels par pouce (PPI) est plus importante que le nombre total de pixels. Si vous développez votre page Web sur un écran classique d'ordinateur, la densité de pixels se situe aux alentours de 100, mais votre terminal mobile a probablement une beaucoup plus grande densité de pixels. Par exemple, mon iPhone 3GS a une densité de pixels de 132 et une résolution de 480 x 320. Si je change pour un iPhone 4, ma densité de pixels sera de 326 et la résolution sera de 960 x 640. Les téléphones ont cependant tous les deux un écran de 3 pouces et demi. La résolution de l'iPhone 4S est double de celle de l'iPhone 3GS

et sa densité mesurée en PPI est pratiquement 2 fois et demi supérieure. Pour ma page Web, cela signifie qu'avec un paramètre de largeur de 480 elle ne s'affichera que sur la moitié de l'écran d'un iPhone 4 même si elle remplit la largeur de l'écran d'un iPhone 3GS.

Cependant, comme je réalise mes développements sur un ordinateur ayant un écran de 20 pouces avec une densité de 99 PPI, le mieux que je puisse faire est une estimation de l'apparence de mes pages sur les différents terminaux mobiles. Je peux estimer l'aspect d'une page Web sur un téléphone mobile en changeant la taille de la fenêtre du navigateur, mais au bout du compte, pour voir réellement ce que donnera la page Web HTML5, il faut la visualiser sur le terminal mobile cible.

Figure 3.9 — Style pour un téléphone tenu verticalement.

De nombreux terminaux mobiles autorisent l'affichage des pages Web verticalement ou horizontalement. Dans ces conditions, quand j'élabore un fichier CSS3 pour un périphérique mobile, j'ai tendance à définir la largeur avec la valeur de la largeur de l'écran tenu horizontalement. Vous vous apercevrez cependant rapidement que les différents navigateurs fonctionnement différemment. Ainsi le navigateur Safari sur l'iPhone affiche la page dans une page minuscule et illisible qui doit être redimensionnée, le navigateur Opera Mini (voir les figures 3.9 et 3.10) sur le même iPhone utilisant la même taille d'écran affiche la page immédiatement dans une taille optimale, que le téléphone soit tenu horizontalement ou verticalement.

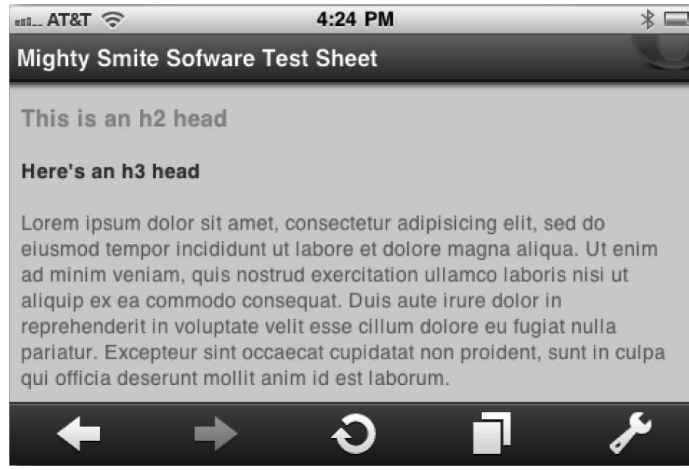


Figure 3.10 — Style défini pour un téléphone tenu horizontalement.

Style en ligne

La troisième méthode pour ajouter du code CSS3 à un document consiste simplement à créer un attribut `style` à un élément qui redéfinit le contenu du conteneur de l'élément. Par exemple, le code suivant (`CSS3EnLigne.html` disponible dans les fichiers du chapitre) comporte des modifications de styles dans le conteneur `<div>` et le deuxième conteneur `<p>` :

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>CSS3 en ligne</title>
</head>
<body>
<div style="font-family:Verdana, Geneva,
sans-serif;font-size:24px;background-color:yellow;color:navy;">Ceci est
important !</div>
<p>Mais ceci... pas tant que cela</p>
<p style="font-size:10px;font-family:sans-serif;">Et vous pouvez ignorer
cela....
</body>
</html>
```

La figure 3.11 illustre le résultat du test de la page Web dans un navigateur. Gardez à l'esprit qu'il n'y a pas d'ajout de style à la deuxième ligne.

L'utilisation d'un style CSS3 en ligne peut se révéler très précieuse quand votre fichier externe CSS3 ne possède pas le style adéquat pour réaliser quelque chose d'indispensable sur une page Web. En général, il est préférable d'utiliser les styles en ligne à la fois avec parcimonie et discernement. Cela est particulièrement vrai quand on collabore avec d'autres développeurs et concepteurs qui travaillent sur une feuille de styles commune.

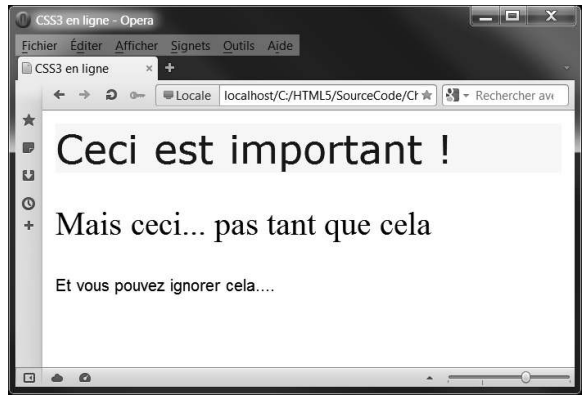


Figure 3.11 — CSS3 en ligne.

3.3.2 Création de classes et d'ID CSS3

Les classes et les ID CSS3 permettent d'étendre un style à n'importe quel élément. Par exemple, supposons que vous ayez une fonctionnalité que vous ne souhaitiez ajouter qu'à quelques éléments, comme une surbrillance jaune. Si vous définissez une couleur de fond d'une `div` ou d'un élément `p` en jaune, tout le texte de ces deux conteneurs sera en jaune, ce qui n'est pas ce que vous désirez. En revanche, si vous avez une classe qui définit une couleur de fond jaune, vous n'avez plus qu'à assigner cette classe à un élément pour le mettre en surbrillance.

Classes CSS3

On crée des classes de styles presque de la même manière que les styles d'éléments. Les définitions avec le "point" (`.`) utilisées pour créer une classe en CSS3 sont des étiquettes que l'on définit au lieu d'utiliser des noms d'éléments. La figure 3.12 montre comment créer une définition de classe CSS3.

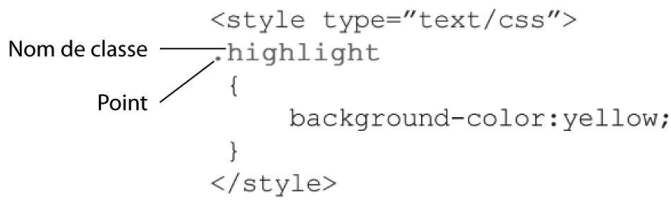


Figure 3.12 — Création d'une définition de classe.

Comme vous pouvez le constater, la définition du point se place là où va le nom de l'élément. Le reste est identique aux définitions CSS3 des éléments. L'implémentation d'un style de classe est cependant un peu différente car il peut être utilisé dans presque n'importe quelle balise d'élément.

Pour voir comment on peut utiliser un texte en surbrillance, le plus pratique est d'employer l'élément en ligne `span`. La balise `<span>` peut être ajoutée au milieu d'un élément de bloc et ne modifie que cette partie du contenu du conteneur `span` sans changer le reste du bloc. Pour ajouter une classe à un élément, on utilise le format suivant :

```
<element class=''myClass''>
```

Vous noterez que le nom de la classe n'inclut pas le point de la définition. Le point n'est utilisé que dans la définition du style pour faire comprendre à l'analyseur que le mot est une classe et non pas un élément. Le programme suivant (`SpanClass.html` disponible dans les fichiers du chapitre) fournit un exemple de la manière dont on peut utiliser la classe avec la balise `<span>`.

```
<!DOCTYPE HTML>
<html>
<head>
<style type=''text/css''>
body {
     background-color:#F93;
}
.highlight {
     background-color:yellow;
}
div {
     font-family:''Comic Sans MS'', cursive;
     font-size:18px;
}
h1 {
     font-family:''Arial Black'', Gadget, sans-serif;
     color:#930;
    text-align:center;
     font-size:20px;
}
</style>
<meta http-equiv=''Content-Type'' content=''text/html; charset=UTF-8''>
<title>Halloween Highlight</title>
</head>
<body>
<h1>Halloween Party!</h1>
<div>You are invited to a Halloween Party on <span class=''highlight''>Saturday,
October 29</span>. Costumes are <span class=''highlight''><i>de
rigueur</i></span>.</div>
</body>
</html>
```

Quand on teste le programme, on s'aperçoit que les deux parties du texte à l'intérieur des conteneurs `<span>` sont affectées. La figure 3.13 illustre le résultat dans un navigateur Opera Mini sur un iPhone (en haut) et dans un navigateur Chrome sur un Mac (en bas).

Sur les deux écrans, on voit clairement que la classe CSS3 nommée `highlight` fonctionne parfaitement. Cependant, dans le navigateur Opera Mini, la police définie

Figure 3.13 — Style défini dans une classe d'un conteneur `<span>` sur un périphérique mobile (en haut) et sur un ordinateur (en bas).

ne s'affiche pas et elle n'est pas en italique (dans Safari, la police est affichée en italique, mais ce n'est pas celle qui est définie).

ID CSS3

Un ID CSS3 est défini presque exactement de la même manière qu'une classe, sauf qu'il utilise un signe dièse (#) au lieu d'un point (.) dans la définition. De plus, en assignant un ID, on utilise `ID` au lieu de `class` pour spécifier quel ID utiliser avec un élément. On peut même utiliser des ID et des classes avec le même élément. La balise suivante est ainsi parfaitement correcte :

```
<p ID=''ceci'' class=''cela''>
```

Tous les deux peuvent sélectionner des styles et l'ID fournit un ID unique pour le paragraphe.

L'ID a quelques différences majeures par rapport à une classe. Une classe et un ID peuvent être utilisés comme sélecteurs de feuille de styles, mais un ID a d'autres limitations et fonctionnalités :

- Un ID ne peut être utilisé qu'une seule fois dans un document.
- Un ID peut servir d'ancre (voir le chapitre 7).
- Un ID peut agir comme référence de script, ce qui est important pour JavaScript.
- Un ID peut être utilisé comme nom dans un élément d'objet déclaré.
- Un ID peut être utilisé par des agents pour traiter des informations lors de la traduction d'un document HTML.

Tant que vous n'utiliserez pas JavaScript ou d'autres langages, vous n'exploiterez que les deux premières fonctionnalités de cette liste. Quoi qu'il en soit, si vous prenez garde à ces différences, vous minimiserez les problèmes sur vos pages Web et les autres vous considéreront comme un pro. L'exemple suivant (`IDCSS3.html` disponible dans les fichiers de ce chapitre) illustre l'utilisation d'un ID avec CSS3 :

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
#littleHead {
font-family:Verdana, Geneva, sans-serif;
background-color:#9FC;
font-size:16px;
}
#javascript {
/* rouge */
color:#cc0000;
}
#php {
/* bleu */
color:#009;
}
#actionscript {
/* vert */
color:#063;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Utilisation des ID</title>
</head>
<body>
<div id="littleHead">Tout ce que vous avez toujours voulu<br>
  savoir sur les variables :</div>
<p id="javascript"> Les variables JavaScript n'ont pas à être assignées à un
type de données.</p>
<p id="php"> Les variables PHP peuvent avoir un type de données grâce au
typage implicite.</p>
<p id="actionscript"> Les variables ActionScript doivent être assignées à un
type de données.</p>
</body>
</html>
```

À l'examen de ce code, on peut se demander ce que signifient les marques slash-astérisque (/* ... */). Il s'agit en fait de commentaires pour CSS3. À l'intérieur d'un conteneur `<style>` et dans une feuille de styles externe, ils fonctionnent exactement comme les marques de commentaires `<!-- -->` en HTML5. La figure 3.14 montre le résultat du programme.

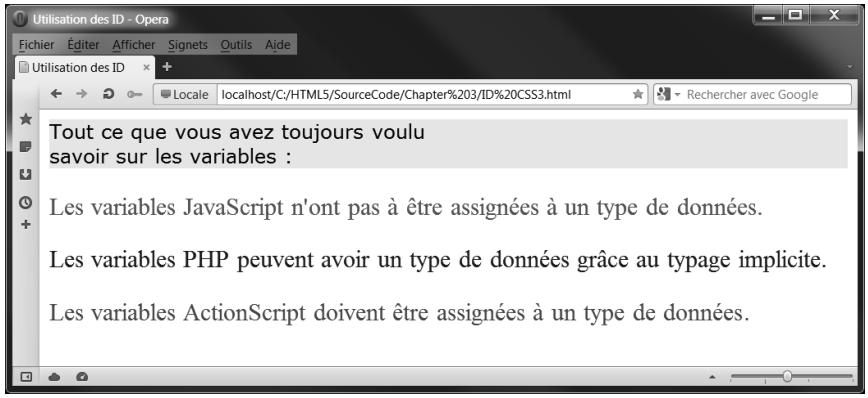


Figure 3.14 — ID sur une page Web.

Si vous avez une longue page Web avec des discussions sur JavaScript, PHP et ActionScript, l'utilisateur va devoir faire défiler la fenêtre pour trouver le sujet qui l'intéresse. En utilisant les ID, on peut écrire l'URL pour qu'elle intègre le paragraphe exact que l'utilisateur tente de localiser. Par exemple, l'URL suivante va aller directement sur le paragraphe qui traite de PHP : `www.smashingHTML5.com/myIDs#php`. La mention `#php` qui a été ajoutée appelle le paragraphe spécifique ayant l'ID php.

3.4 À VOUS DE JOUER !

Ce chapitre a traité de nombreuses notions et vous allez pouvoir réinvestir votre apprentissage. Voici deux exercices :

1. **Vous pouvez améliorer votre conception !** Après avoir démarré la création d'une page Web en utilisant différents éléments `h`, la page que vous avez obtenue et qui est illustrée à la figure 3.4 a besoin d'être améliorée. Pour une page qui est destinée à des enfants, elle manque de couleurs et les polices sont ennuyeuses. De plus, le texte est collé à l'image. En utilisant CSS3, pouvez-vous améliorer la conception de cette page ?
2. **Aidez ce pauvre Wittgenstein !** La page Web illustrée à la figure 3.6 affiche l'œuvre de Wittgenstein sans les indentations. En utilisant CSS3 et la propriété `margin-left`, voyez si vous pouvez régler ces éléments `h` de telle sorte que tous les éléments soient présents.

Amusez-vous bien et appréciez la souplesse que procure CSS3.

4

Valeurs des couleurs

Objectif

Jusqu'à présent, nous avons vu plusieurs exemples d'utilisation des codes de couleurs, mais à moins que vous ne compreniez ce que vous lisez, vous avez dû avoir l'impression d'avoir affaire à un langage codé. Dans certains exemples, on a utilisé des noms de couleurs, mais il ne s'agissait pas des couleurs de base, et il devient nécessaire de comprendre la manière dont les couleurs sont gérées en CSS3. Cela vous permettra d'avoir accès à des millions de couleurs plutôt qu'à quelques-unes.

4.1 COULEUR RGB

Si vous avez déjà mélangé des couleurs, ne serait-ce qu'avec les doigts sur une aquarelle, vous savez ce qui se produit. Sur un écran d'ordinateur, les lumières de couleur rouge, vert et bleu (RVB en français) se mélangent pour générer des couleurs différentes. Par exemple, si vous mélangez en quantité égale du rouge et du vert, vous obtenez du jaune.

Pour obtenir des couleurs sur les pages Web, on mélange différentes valeurs en utilisant des nombres entiers, hexadécimaux ou bien des pourcentages. CSS3 possède aussi un nombre limité de noms de couleurs que vous pouvez employer tout en vous servant des autres méthodes de définition des couleurs. HTML5 et CSS3 ont des éléments très sophistiqués, comme l'élément `canvas`, qui améliore considérablement la gestion des couleurs et des images par rapport aux versions précédentes de HTML. Ces techniques avancées nécessitent un peu de JavaScript et nous les détaillerons au chapitre 13. Pour l'instant, nous allons commencer par les bases.

4.1.1 Utilisation des noms

Une des expériences les plus étranges avec HTML5 et CSS3 est l'utilisation d'un jeu de noms pour les couleurs. Il existe à la base un ensemble de 16 couleurs standard qui sont listées dans le tableau 4.1.

Tableau 4.1 — Noms des couleurs standard

Noms des couleurs standard			
Aqua	Black	Blue	Fuchsia
Gray	Green	Lime	Maroon
Navy	Olive	Purple	Red
Silver	Teal	White	Yellow

En utilisant les rudiments de HTML5 que vous avez appris jusqu'à présent, vous pouvez facilement créer un graphique qui affiche toutes les couleurs (dans la section « À vous de jouer ! » à la fin de ce chapitre, vous allez devoir recréer ce tableau). La figure 4.1 illustre l'aspect de cette palette de couleurs sur une page Web.

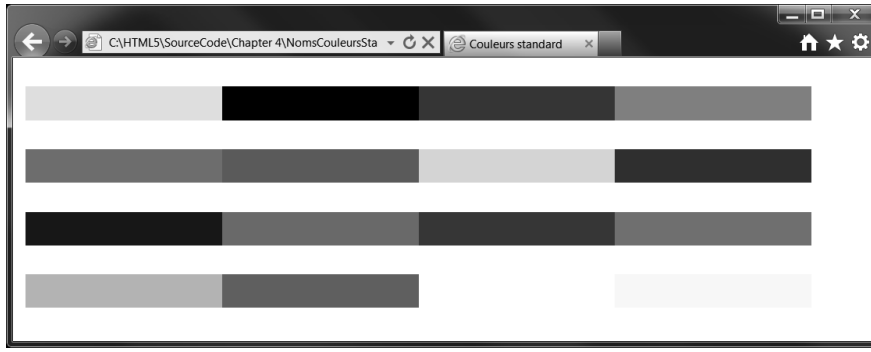


Figure 4.1 — Couleurs standard CSS3 sur une page Web.

À partir de cette base-là, vous pouvez inclure 131 autres noms qui semblent avoir été choisis au hasard. Ils font tous partie d'un ensemble créé dans les années 1980 et qui se nommait X11. Ils ont été adoptés par les premiers navigateurs et ont été conservés par la suite. Dans la documentation officielle du W3C, ils sont listés dans la rubrique SVG (Scalable Vector Graphics) et ils proviennent tous du jeu original X11. La liste complète peut être consultée à l'adresse :

```
www.w3.org/TR/SVG/types.html#ColorKeywords
```

Si tous ces noms n'ont pas été listés ici, c'est parce qu'en général les concepteurs et les développeurs ne les utilisent pas. Les concepteurs considèrent que 131 noms limitent trop leur palette et qu'en plus, le choix a été fait en dépit du bon sens. Des noms de couleurs comme papayawhip et mistyrose ne parlent pas vraiment à des

artistes. De la même manière, les développeurs pensent que les valeurs utilisées ne correspondent pas à un ensemble mathématique comme les bonnes vieilles couleurs du Web qui suivent une logique numérique standard (bien entendu, si vous voulez vous amuser, vous pouvez inclure les couleurs darkkhaki et ghostwhite dans la palette de couleurs de vos pages Web). Dans les sections suivantes, vous verrez comment créer la couleur exacte que vous voulez en choisissant parmi un million de combinaisons possibles.

4.1.2 Pourcentages RGB et HSL

Quand on mélange des couleurs, on utilise parfois des quantités de peinture exprimées en pourcentage. Un certain pour cent de rouge, de vert et de bleu va donner une couleur précise. Lors de la définition des couleurs en CSS3, on peut utiliser des pourcentages de deux manières différentes. On peut d'abord assigner une valeur de couleur en utilisant le format suivant :

```
rgb(r%,g%,b%);
```

La première valeur est le pourcentage de rouge, la seconde le pourcentage de vert et la troisième le pourcentage de bleu. Par exemple, le paramètre `rgb(43.9%,50.2%,56.5%)` génère la couleur utilisée par l'équipe de baseball Los Angeles Dodgers. L'addition des trois pourcentages produisant un total supérieur à 100 pour cent, vous comprenez donc que le pourcentage est un pourcentage de la couleur elle-même et non pas sa proportion dans le mélange. Comme vous pouvez le constater, on peut être très précis car les valeurs peuvent inclure des fractions de pourcentages. Le script suivant (`RGBpourcent.html` disponible dans les fichiers du chapitre) montre comment utiliser cette assignation de couleur dans une page HTML5 :

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
body {
background-color:rgb(43.9%,50.2%,56.5%);
}
h1 {
background-color:rgb(11.8%,56.5%,100%);
color:rgb(100%,100%,100%);
font-family:"Arial Black", Gadget, sans-serif;
font-style:italic;
text-align:center;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Bleu Dodger</title>
</head>
<body>
<h1>Los Angeles Dodgers<br>
```

```
  (Anciennement de Brooklyn)</h1>
</body>
</html>
```

Quand on exécute la page, les couleurs s'affichent précisément comme cela a été demandé (le résultat est illustré à la figure 4.2).

Figure 4.2 — Définition de couleurs avec des pourcentages RGB.

La deuxième méthode pour assigner des couleurs à l'aide de pourcentages est d'utiliser le modèle HSL (*hue saturation lightness*, en français TSL pour teinte saturation luminance). Le gros avantage de HSL est que la clarté est symétrique. Cela facilite la mise au point des couleurs.

Si l'on se représente un cercle de couleurs de 360 degrés comme une boussole, on sélectionne une teinte. Au sommet (à 0 pour cent), on trouve les rouges. Si l'on se déplace dans le sens des aiguilles d'une montre, à 30 pour cent, les teintes virent au rouge orangé. À 60 pour cent, elles deviennent jaunes. Et ainsi de suite tout autour du spectre de couleurs jusqu'à ce que l'on atteigne 360 pour cent (ou 0 pour cent) où l'on retrouve les teintes rouges. Pour les concepteurs qui comprennent le spectre de couleurs, cela facilite le choix des couleurs. Pour créer une couleur plus claire, on augmente la valeur de la lumière et on diminue la valeur de la lumière pour faire une teinte plus foncée. Supposons par exemple que l'on tente d'obtenir la bonne nuance de rouge. On commence par l'assignation de couleur suivante :

```
hsl(0,100%,50%);
```

Vous noterez que la première valeur n'est pas un pourcentage. Cela est dû au fait que les valeurs sont comprises entre 0 et 359 (les 360 degrés d'un cercle ; n'oubliez pas que 0 et 360 représentent le même point du cercle). En augmentant ou en diminuant la lumière (le troisième paramètre), on peut rendre la couleur plus claire ou plus foncée, ce qui est bien plus intuitif que de modifier les pourcentages RGB. Le script HTML5/CSS3 suivant (`CouleurHSL.html` disponible dans les fichiers du chapitre) montre comme il est facile de diminuer et d'augmenter la valeur de la lumière pour obtenir exactement la bonne teinte de rouge.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
.redBase {
color:hsl(0, 100%, 50%);
}
.redDarker {
color:hsl(0, 100%, 25%);
}
.redLighter {
color:hsl(0, 100%, 75%);
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Assignation de couleur HSL</title>
</head>
<body>
<h1 class="redBase">Rouge de base</h1>
<h1 class="redDarker">Rouge foncé</h1>
<h1 class="redLighter">Rouge léger</h1>
</body>
</html>
```

Quand on utilise HSL pour la première fois, il faut se rappeler le rôle que jouent l'ombre et la lumière du soleil sur les couleurs. Ce processus de mise au point est plus facile pour les concepteurs qui sont habitués à ces notions. La figure 4.3 montre les différentes teintes de rouge obtenues.

Figure 4.3 — HSL facilite la mise au point des nuances.

Les concepts de nuance et de lumière sont faciles à comprendre, mais la notion de saturation peut se révéler plus délicate. Fondamentalement, la saturation est la quantité de couleur d'une couleur donnée. Une saturation de 100 pour cent est la couleur d'une intensité totale d'une teinte sous une lumière donnée, alors qu'avec un pourcentage inférieur la couleur paraît plus fade. Pour toutes les couleurs, une lumière médiane sera grise quand la saturation est de 0 pour cent. Il arrive que des couleurs décolorées soient préférables comme celle d'un jean délavé.

4.1.3 Paramétrage RGB avec des entiers décimaux

Il existe un deuxième moyen de mélanger des couleurs à l'aide de valeurs RGB : il suffit d'insérer des valeurs comprises entre 0 et 255 (soit un total de 256 valeurs puisque l'on compte le 0) à la place des pourcentages employés dans l'exemple précédent. La valeur 256 représente le nombre de combinaisons possibles avec un nombre stocké sur deux octets (16 bits). En d'autres termes, le système est basé sur la manière dont un ordinateur stocke et traite l'information. Avec un ensemble de trois valeurs comprises entre 0 et 255, on peut générer 16 777 216 combinaisons. La technologie des couleurs est cependant bien plus complexe que ce qu'il nous est possible d'en dire ici et les traitements de couleurs modernes permettent de générer des couleurs encore meilleures. Nous nous contenterons d'affirmer ici que l'on peut générer un très grand nombre de couleurs avec ces combinaisons de rouge, vert et bleu. Voici le format pour assigner une valeur de couleur :

```
rgb(integerR, integerG, integerB);
```

Par exemple, voici le jaune qui est un mélange de rouge et de vert :

```
rgb(255,255,0);
```

Cela n'est pas aussi intuitif que HSL, mais au bout d'un moment, on commence à avoir une bonne sensation des mélanges basés sur 256 valeurs plutôt que sur des pourcentages. L'exemple suivant (`CouleurDec.html` disponible dans les fichiers du chapitre) illustre une implémentation simple.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
body {
/* Couleur de fond rouge */
background-color:rgb(255,0,0);
}
h1 {
/* Grand texte en jaune */
color:rgb(255,255,0);
font-family:Verdana, Geneva, sans-serif;
font-size:36px;
font-weight:bold;
}
h2 {
/*Texte en bleu + fond gris */
color:rgb(0,0,255);
background-color:rgb(150,150,150);
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Couleurs décimales</title>
</head>
```

```
<body>
<h1>  Grand titre en jaune</h1>
<h2>  Titre en bleu avec un fond gris</h2>
</body>
</html>
```

La seule différence entre l'utilisation de RGB avec des valeurs de 0 à 255 et des valeurs exprimées en pourcentage relève de la perception. Vous pouvez penser être plus précis avec des couleurs exprimées en 256 valeurs plutôt qu'avec une plage d'une centaine de valeurs, mais cela n'est pas le cas parce que l'on peut utiliser des fractions avec les assignations en pourcentage. L'emploi de la notation en pourcentage ou la notation sur 256 valeurs n'est qu'une question de préférences personnelles. La figure 4.4 illustre le résultat à l'aide du navigateur Opera Mini sur un iPhone.

Figure 4.4 — Couleurs mélangées à l'aide de valeurs exprimées avec des entiers décimaux.

Comme vous pouvez le voir à la figure 4.4, le terminal mobile n'affiche pas la police Arial Black, mais il n'a pas de problème avec les couleurs. Assurez-vous de contrôler l'affichage des polices et des autres effets sur un terminal mobile si cela est essentiel pour votre site. **Rappel** : la plupart des ordinateurs ont un ensemble de polices et de styles bien plus complet que les terminaux mobiles, même si cette différence risque de s'estomper avec le temps.

4.1.4 Paramétrage avec des valeurs héxadécimales

Dans les chapitres précédents, vous avez vu l'assignation de couleurs à l'aide de valeurs contenant des caractères alphanumériques (un caractère alphanumérique est un caractère qui peut être un chiffre ou une lettre). Par exemple, la valeur `6F001C` génère un rouge très foncé. Si on décompose cette valeur, on peut voir qu'il s'agit simplement d'un mélange de rouge, de vert et de bleu. Mais pour comprendre ce qui se passe, on doit en savoir un peu plus sur les systèmes de numération informatiques.

Nous sommes habitués à compter à l'aide d'un système décimal. On utilise les valeurs de 0 à 9 (dix chiffres), et une fois que tous les chiffres sont utilisés, on

recommence avec un nombre à deux chiffres (1 et 0) que l'on appelle « dix ». Comme vous le savez peut-être, les ordinateurs sont basés sur un système de commutateurs à deux états (*On* et *Off*). En remplaçant *On* par 1 et *Off* par 0, on peut écrire un code basé sur un système binaire ; au lieu d'avoir dix chiffres, on en a seulement deux, 0 et 1. Le tableau 4.2 liste les valeurs de 0 à 15 écrites en binaire, en décimal et en base 16, que l'on appelle également système hexadécimal.

Tableau 4.2 — Système de numération

Binaire	Décimal	Hexadécimal
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

Chacune des valeurs binaires s'appelle un bit (0 ou 1), et un groupe de huit bits s'appelle un octet. Dans le tableau 4.2, la plus grande valeur binaire est exprimée dans un nombre de 4 bits. La plus grande valeur d'un octet est donc exprimée sous la forme d'un nombre de 8 bits, soit 11111111. Quand on compare les valeurs décimales et hexadécimales de ce nombre, on voit quelque chose d'intéressant, comme cela est illustré dans le tableau 4.3.

Comme vous pouvez le voir dans le tableau 4.3, la valeur hexadécimale FF est la valeur la plus élevée pour un nombre de deux chiffres ; de la même manière, la valeur binaire 11111111 est la valeur la plus élevée d'un octet, alors que le nombre décimal

Tableau 4.3 — Valeurs d'un octet

Binaire	Décimal	Hexadécimal
11111111	255	FF

est sur trois chiffres et ne représente aucune limite. En d'autres termes, le système décimal n'est pas très symétrique avec le système de numération binaire, ce qui n'est pas le cas du système hexadécimal.

Comme vous le savez, le système RGB d'assignation des entiers aux valeurs de couleurs utilise des valeurs de 0 à 255. En utilisant des valeurs hexadécimales, on n'a besoin que de nombres à deux chiffres (en fait des entiers hexadécimaux) pour représenter la totalité des 256 valeurs, ce qui est plus propre et favorise l'utilisation de l'hexadécimal dans le système d'assignation des valeurs de couleurs.

En utilisant six valeurs hexadécimales (deux pour le rouge, deux pour le vert et deux pour le bleu), on peut assigner n'importe quelle valeur de couleur. Ainsi la valeur `6F001C` doit être décomposée de la manière suivante :

- Rouge : `6F`
- Bleu : `00`
- Vert : `1C`

Le fait de s'habituer à l'hexadécimal peut prendre un peu de temps, mais une fois que cela est fait, il est facile d'ajouter des valeurs de couleurs grâce à ce système. Vous pouvez vous le représenter de la même manière que les entiers décimaux RGB, mais au lieu d'utiliser les valeurs de 0 à 255, on emploie les nombres de 00 à FF. L'exemple suivant (`PaletteHexa.html` disponible dans les fichiers de ce chapitre) montre l'utilisation de couleurs codées en hexadécimal.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
/* Palette -- On n'utilise que ces couleurs !
69675C, 69623D,ECE8CF,  E8D986,    B5AA69
  gray,  olive, cream, dark cream, khaki */
body {
font-family:"Comic Sans MS", cursive;
background-color:#ECE8CF;
color:#69675C;
}
h1 {
font-family:"Arial Black", Gadget, sans-serif;
color:#B5aa60;
background-color:#E8D986;
text-align:center;
}
h2 {
font-family:"Lucida Sans Unicode", "Lucida Grande", sans-serif;
```

```
color:#b5aa69;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Palette avec des valeurs hexadécimales</title>
</head>
<body>
<h1> Style avec une palette de couleurs</h1>
<h2> Désert à l'automne</h2>
À l'automne, quand l'air se rafraîchit, le désert commence à se parer d'un
ensemble de nuances plus chaudes.
</body>
</html>
```

Cet exemple utilise une palette de couleurs et place simplement les valeurs de couleurs dans un commentaire au sein du conteneur `<style>` de telle sorte qu'il puisse être visualisé quand on assemble la page Web. La figure 4.5 illustre le résultat.

Figure 4.5 — Palette de couleurs codées en hexadécimal.

Les couleurs appartiennent à un ensemble de couleurs qui crée une certaine atmosphère. Celle-ci, appelée Désert à l'automne, est censée représenter une palette de couleurs qui exprime l'ambiance qui règne à cette saison dans le désert.

4.2 AJOUT DE TRANSPARENCE À LA COULEUR

La transparence, ou opacité variable, est l'une des nouvelles fonctionnalités que l'on rencontre sur les navigateurs qui sont compatibles HTML5. Un objet totalement opaque à l'écran empêche de voir tout ce qui se trouve derrière lui, alors qu'un objet totalement transparent permet de voir tout ce qui se trouve dessus, comme s'il était en verre. La valeur utilisée pour décrire le niveau d'opacité est exprimée dans une propriété dont la plage varie entre 0 et 1. Que l'on utilise le système de couleur RGB ou HSL, cette valeur est le quatrième paramètre qui ne peut malheureusement pas être en hexadécimal. Par exemple, `rgba(255,0,0, 0.5)` génère du rouge avec 50 pour cent d'opacité. De la même manière, `hsla(120, 100%, 50%, 0.3)` crée du vert avec 30 pour cent d'opacité (ou 70 pour cent de transparence).

Dans la quatrième partie de cet ouvrage, je traite des différents moyens d'ajouter de la profondeur à vos pages avec la balise `<canvas>` de telle sorte que lorsque l'on empile des objets les uns sur les autres, on peut mieux voir pourquoi l'ajout de la transparence est important. Mais pour l'instant, vous avez besoin de quelque chose que vous pouvez placer sous les blocs de texte afin qu'ils puissent être visualisés au travers d'un bloc de texte transparent. La méthode la plus facile est de placer un objet d'arrière-plan en utilisant la propriété `background-image`. L'extrait de code suivant illustre cette technique :

```
body { background-image:url(fichierImage.png); }
```

Vous pouvez utiliser n'importe quel fichier `.jpg`, `.gif` ou `.png` comme image de fond. Dans notre exemple, trois cercles de couleur rouge, verte et bleue sont utilisés comme arrière-plan et on trouve au sommet de la page du texte `<h1>` avec 50 pour cent d'opacité pour montrer l'effet que les différentes couleurs ont quand elles sont affichées au travers d'un objet transparent. Le code suivant (`Transparence.html` disponible dans les fichiers de ce chapitre) utilise à la fois les formats `rgba()` et `hsla()`.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
body {
background-image:url(rgbBalls.png);
}
.transRed {
color:rgba(255, 0, 0, .5);
}
.transGreen {
color:rgba(0, 255, 0, .5);
}
.transBlue {
color:hsla(240, 100%, 50%, .5);
}
.transBackground
{
background-color:hsla(120, 100%, 50%, .5);
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Transparence/Opacité</title>
</head>
<body>
<h1 class="transRed">Testing 123, Testing 123, Testing 123</h1>
<h1 class="transGreen">Testing 123, Testing 123, Testing 123</h1>
<h1 class="transBlue">Testing 123, Testing 123, Testing 123</h1>
<h1 class="transBackground">Testing 123, Testing 123, Testing 123</h1>
</body>
</html>
```

Le résultat est illustré à la figure 4.6 ; comme vous pouvez le constater, le texte et l'arrière-plan transparents permettent à l'image de fond d'être vue au travers.

Quand une couleur est transparente, elle prend la couleur sous-jacente. En pareil cas, il faut donc faire attention à harmoniser la couleur sous-jacente et la couleur de recouvrement (en ce sens, la figure 4.6 illustre bien les raisons pour lesquelles on utilise rarement des images d'arrière-plan car elles mettent en désordre l'écran et détruisent toute sensibilité du texte).

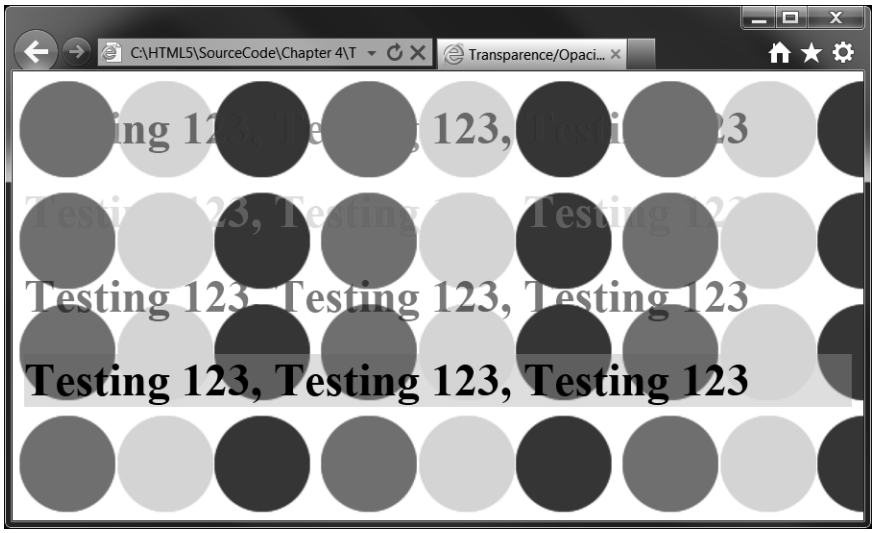


Figure 4.6 — Texte transparent par-dessus des images.

4.3 CRÉATION D'UN MODÈLE DE COULEURS

Si vous êtes un concepteur, vous pensez sans doute qu'il est difficile de choisir sa palette de couleurs parmi tous ces nombres. Si vous êtes un développeur, vous vous posez peut-être la question de savoir si les couleurs que vous allez utiliser vont bien ensemble. Ces deux questions ont la même réponse : Kuler. Kuler est un site où l'on peut saisir une couleur clé (couleur de base) et à l'aide de différents algorithmes, Kuler détermine les couleurs qui sont compatibles et affiche les informations des valeurs de couleurs en décimal et hexadécimal. Les créateurs peuvent saisir n'importe quelle couleur qu'ils veulent employer et Kuler génère tous les calculs ; les développeurs peuvent saisir n'importe quelle valeur et Kuler génère le modèle de couleurs.

Kuler est disponible à l'adresse `http://kuler.adobe.com`. Il nécessite un plug-in Flash (qui est déjà intégré à la plupart des navigateurs) et si votre navigateur n'en dispose pas, vous pouvez le télécharger gratuitement à `www.adobe.com/products/flashplayer`. Vous pouvez aussi télécharger un widget Kuler qui fonctionne sur votre ordinateur.

4.3.1 À partir d'une couleur de base

Pour créer un modèle de couleurs avec Kuler, on commence avec une couleur de base puis on teste différents algorithmes pour générer des modèles de couleurs. Ensuite, on sélectionne un algorithme pour afficher différentes manières d'associer les couleurs. Les règles qui sont basées sur la théorie des couleurs permettent de choisir entre les paramètres suivants : analogue, monochromatique, triade, complémentaire, composé, teintes, ou personnalisé. La dernière catégorie est pour les concepteurs qui utilisent leurs talents artistiques pour générer une palette (les développeurs ne sont pas non plus oubliés car ils ont des algorithmes automatiques). La figure 4.7 illustre un exemple typique de modèle de couleurs généré à partir d'une couleur de base à l'aide de l'algorithme analogue.

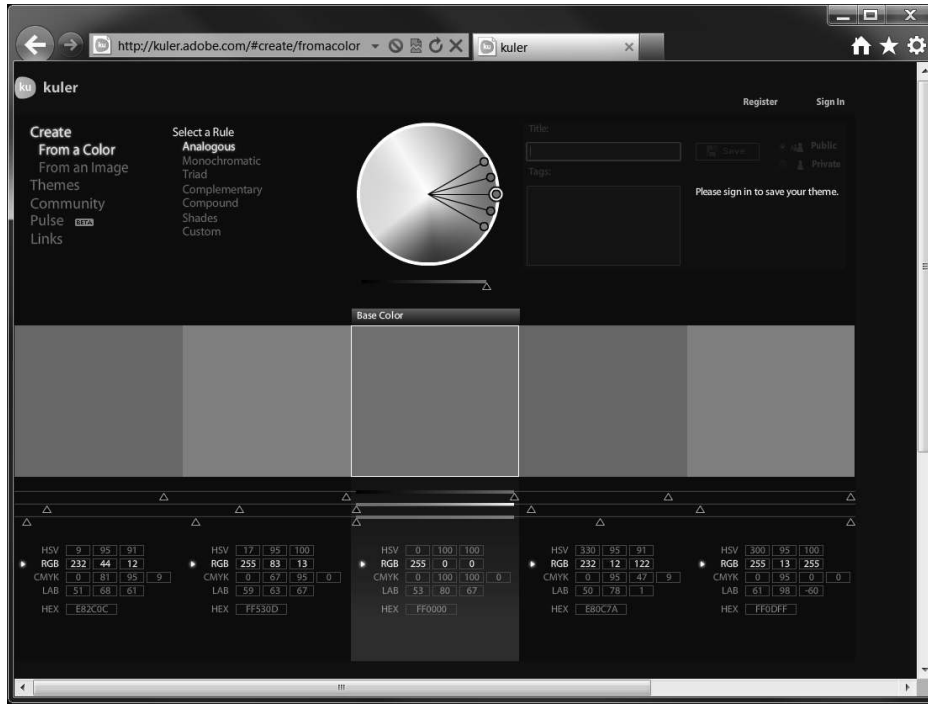


Figure 4.7 — Modèle de couleurs généré à partir d'une couleur de base.

4.3.2 À partir d'une image

Outre la création d'une palette de couleurs à partir d'une couleur de base, vous pouvez aussi charger une image et Kuler génère automatiquement un modèle de couleurs basé sur la couleur de l'image importée.

Mauvaises combinaisons de couleurs

Afin de voir les différences entre un bon modèle de couleurs et une mauvaise combinaison, nous allons observer un exemple. L'ouvrage de Leslie Cabarga, *The Designer's Guide to Color Combinations,* contient un chapitre sur les mauvaises couleurs. La figure suivante montre à quoi ressemblent deux pages Web identiques avec un modèle de couleurs basé sur une photo et un modèle basé sur un exemple de mauvaise couleur issu du livre de Cabarga.

L'image de gauche utilise les couleurs générées à partir de la photo, ce qui n'est pas le cas de l'image de droite, d'où une mauvaise combinaison.

Figure 4.8 — Bonne et mauvaise couleur.

Quand on utilise une image, on peut modifier le modèle de couleurs en choisissant plusieurs ambiances : coloré, brillant, atténué, profond et sombre. Tous les modèles de couleurs peuvent être enregistrés et quand ils sont chargés, ils conservent toutes les informations dont vous avez besoin pour saisir les données relatives aux couleurs dans une page Web HTML5.

4.4 INTÉGRATION DE LA PALETTE DE COULEURS À LA PAGE WEB

Le fait d'avoir une palette de couleurs ne signifie pas que votre page sera jolie, ni même harmonieuse du point de vue des couleurs. Avec la même palette, certaines couleurs vont mieux ensemble que d'autres. Par exemple, un arrière-plan en ton moyen peut ne pas fournir le contraste nécessaire pour les autres tons moyens, si bien qu'une couleur sombre ou claire peut constituer un meilleur choix. La figure 4.8 montre la palette de couleurs développée à partir d'un logo qui sera utilisée comme palette de la page.

Les valeurs hexadécimales des quatre couleurs sont rappelées pour mémoire en commentaire dans le code CSS3 au sommet de la page HTML5. Le script suivant

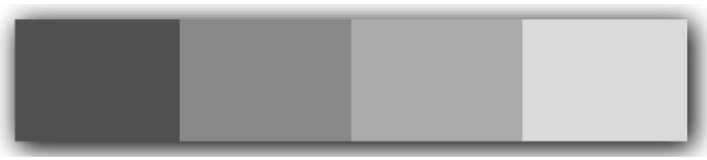


Figure 4.9 — Palette de couleurs basée sur un logo.

(`ColeursPalette.html` disponible dans les fichiers du chapitre) emploie les couleurs de telle sorte qu'elles fonctionnent avec le logo et le reste de la page.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
/* 027333,7FA646,D9B448,F2DFA7 */
body {
margin-left:1em;
background-color:#F2DFA7;
color:#027333;
font-family:Verdana, Geneva, sans-serif;
font-size:11px;
}
h1 {
font-family:Tahoma, Geneva, sans-serif;
color:#7FA646;
}
h2 {
font-family:"Lucida Sans Unicode", "Lucida Grande", sans-serif;
color:#7FA646;
background-color:#D9B448;
}
div
{
text-align:center;
}
a {
font-family:Arial, Helvetica, sans-serif;
text-align:center;
font-size:10px;
text-decoration:none;
background-color:#027333;
color:#F2DFA7;
}
a:hover {
color:#D9B448;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Couleurs compatibles</title>
</head>
<body>
<div><nav>
<a href="#">  Lien 1 | </a>
```

```
<a href="#">  Lien 2 | </a>
<a href="#">  Lien 3  </a>
</nav> </div>
<img src="Sandlightlogo.gif" alt="Logo" align="left">
<header><h1>    Bienvenue</h1></header>
<br><br>
<article>
<h2> Présentation...</h2>
Sandlight Productions est une entreprise de développement qui se spécialise
dans les technologies HTML5/CSS3, le streaming vidéo, les terminaux mobiles,
le e-learning, ActionScript 3.0, Flash et PHP.
</article>
<br>
<footer><div>
<nav>
<a href="#">  Lien 1 | </a>
<a href="#">  Lien 2 | </a>
<a href="#">  Lien 3  </a>
</nav>
</div></footer>
</body>
</html>
```

Le script CSS3 utilise la propriété `a:hover` pour modifier la propriété quand la souris passe sur un lien. Dans la définition CSS3 de la balise `<a>`, `text-decoration` est initialisé à `none`, ce qui signifie que le lien du texte ne sera pas souligné. Le soulignement étant absent, il est nécessaire d'alerter l'utilisateur de la présence d'un lien, d'où l'utilisation de la propriété `hover`. En modifiant légèrement la couleur du texte du lien, on montre effectivement à l'utilisateur que la souris se trouve sur un lien. La couleur initiale et la couleur de survol font toutes les deux parties de la palette. Dans la définition de la page, vous devez donc vous souvenir qu'il n'y a pas que les balises `<body>` et `<h>` qui utilisent la palette de couleurs.

Cette conception particulière est centrée sur les périphériques mobiles, mais elle s'affiche correctement sur les ordinateurs (voir la figure 4.9) et les tablettes.

Bien entendu, votre page sera toujours plus jolie si vous pouvez vous offrir les services d'un concepteur Web. Les développeurs peuvent toutefois améliorer l'aspect de leurs pages en accordant de l'attention aux combinaisons de couleurs.

4.5 À VOUS DE JOUER !

Les deux exercices suivants devraient vous amuser et vous apprendre plein de choses :

1. **Reproduction du tableau des couleurs standard :** La Figure 4.1 illustre une image avec des couleurs standard. Le premier défi consiste à voir si vous pouvez reproduire la page Web qui affiche ces couleurs. Voici deux indices pour vous permettre de commencer :
   - Définissez chaque couleur nommée en tant que classe dans le conteneur `<style>` avec la même couleur pour les couleurs de texte et d'arrière-plan.

Figure 4.10 — Modèle de couleurs appliqué à une page.

```
.aqua { color:aqua; background-color:aqua; }
```

– Pour ce faire, on peut utiliser la balise `<span>` pour assigner les classes au contenu du conteneur `<span>`.

```
<h3> <span class=aqua>COLORNAME</span><span class=black>COLORNAME
</span><span class=blue>COLORNAME</span><span class=fuchsia>COLORNAME
</span> <h3>
```

2. **Votre photo sur une page Web !** Il y a trois étapes à réaliser :
   – Prenez une photo numérique de vous-même.
   – Chargez cette photo dans Kuler et créez une palette de couleurs.
   – Créez une page Web avec votre photo en utilisant la palette de couleurs créée avec Kuler.

DEUXIÈME PARTIE

Conception de pages et de sites Web

5

Organisation de la page

Objectif

Un grand nombre des nouvelles balises HTML5 sont des balises organisationnelles. Dans les chapitres précédents, nous en avons utilisé quelques-unes, mais nous ne les avons pas vraiment expliquées. Ce chapitre étudie en détail l'organisation des pages HTML5 à l'aide de CSS3 et la manière dont il faut appréhender ce processus organisationnel. On ne peut comprendre certains éléments qu'une fois qu'on a commencé à utiliser JavaScript, mais si vous définissez votre page en suivant les recommandations HTML5, votre page sera parfaitement opérationnelle quand vous commencerez à y ajouter un peu de JavaScript.

5.1 SOMMET D'UN DOCUMENT HTML5

Les quatre premiers chapitres de ce livre expliquent comment utiliser les informations au-dessus de la balise `<body>`. Le code au-dessus de la balise `<body>` n'ajoute pas de contenu à la page Web, mais il influence l'aspect de la page et informe le navigateur qu'il s'agit d'une page Web et lui indique son type. La figure 5.1 illustre l'organisation générale de la première partie d'une page Web.

La balise `<html>` est l'élément racine et on peut y inclure un attribut de langue. On trouve ensuite au sein du conteneur `<head>` les métadonnées. Figurent également dans le conteneur `<head>` les éléments de script qui seront brièvement décrits dans cette section et expliqués en détail dans la quatrième partie de cet ouvrage.

Mis à part dans les scripts CSS3, nous n'avons pas placé jusqu'à présent beaucoup de balises dans la section `head` de nos exemples de documents HTML5. La balise

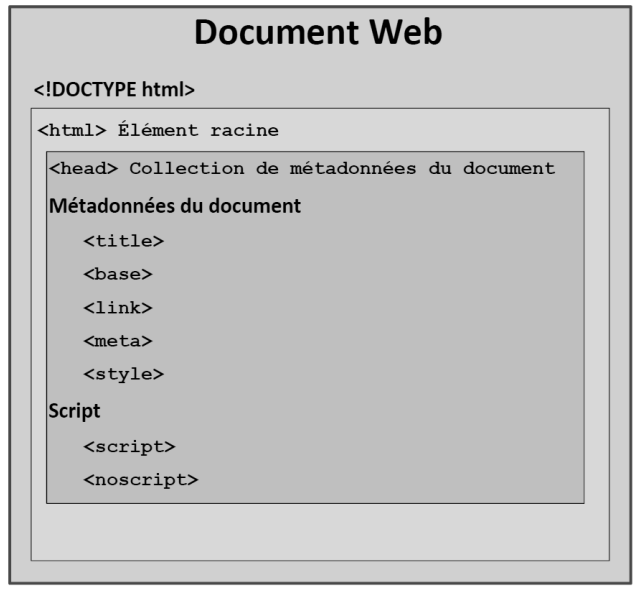


Figure 5.1 — Organisation du sommet d'une page Web.

`<meta>` a de nombreuses utilisations, mais nous ne l'avons employée pour l'instant que pour spécifier le jeu de caractères. Nous verrons d'autres utilisations de cette balise dans ce chapitre.

5.1.1 Définition de la base de référence

Dans un site Web typique, on a en général plusieurs pages différentes qui sont reliées les unes aux autres, ce qui constitue un système de navigation. Si l'on définit une balise `<base>` dans la section `head` de la page avec un lien vers une URL, on peut faire référence aux autres pages par rapport à cette page de base. Par exemple, les deux scripts suivants (`Base.html` et `PremiereBase.html` disponibles dans les fichiers du chapitre) ont des liens l'un vers l'autre, mais il s'agit de liens relatifs par rapport à la base qui est définie dans le conteneur `head`.

```
<!DOCTYPE HTML>
<html>
<head>
<base href="http://www.sandlight.com/html5/smashing/">
<style type="text/css">
body {
background-color:#FCC;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Base d'accueil</title>
</head>
```

```
<body>
<h1>Ceci est la base d'accueil</h1>
<a href="PremiereBase.html">Première base</a>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html>
<head>
<base href="http://www.sandlight.com/html5/smashing/">
<style type="text/css">
body {
background-color:#FC0;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Première base</title>
</head>
<body>
<h1>Ceci est la première base</h1>
<a href="Base.html">Base d'accueil</a>
</body>
</html>
```

Que se passe-t-il ici ? La balise `<base>` dit au navigateur comment résoudre toutes les références aux autres documents du fichier HTML, comme dans l'ancre `<a href=''Base.html''>`. Le navigateur saura chercher le document `Base.html` dans l'emplacement spécifié dans la balise `<base>`, à savoir `http://www.sandlight.com/html5/smashing/`.

5.1.2 Description du site avec des métadonnées

Jusqu'à présent, nous avons utilisé la balise `<meta>` pour établir que votre site utilise le jeu de caractères UTF-8, mais cet élément peut faire bien d'autres choses. Il faut se représenter l'élément `meta` comme une balise multitâche. L'un des attributs les plus importants de l'élément `meta` est la paire `name` et `contents`. En définissant l'attribut `name` à `keywords`, on peut spécifier le contenu de son site de telle sorte que les moteurs de recherche pourront l'indexer. Ainsi, les internautes qui cherchent des produits ou des services seront orientés vers votre site. Supposons, par exemple, que votre site comporte des liens vers des blogs et d'autres sites sur les niches pour chiens. Votre balise meta pourrait ressembler à ceci :

```
<meta name=''keywords'' content=''niches, barrières pour chien, barrière
antifugue''>
```

Chaque valeur de l'attribut `content`, qui doit être séparée par une virgule, est censée décrire le contenu de votre site. Ces balises, qui sont faciles à définir, permettent d'orienter les utilisateurs vers votre site.

La balise `<meta>` a un autre attribut très intéressant : `http-equiv`. En lui attribuant la valeur `Refresh`, on peut automatiquement actualiser une page, voire changer son contenu. Par exemple, vous pouvez afficher automatiquement les photos d'un diaporama sur votre site. Voici la syntaxe de l'utilisation de `Refresh` :

```
<meta http-equiv="Refresh" content="[secs]">
```

L'exemple suivant actualise (recharge) la page toutes les 30 secondes :

```
<meta http-equiv="Refresh" content="30">
```

Vous pouvez non seulement recharger la même page, mais vous pouvez aussi recharger des pages différentes. Si vous voulez charger une séquence de pages, vous pouvez définir la balise meta de la manière suivante afin de lui assigner une URL après une demi-seconde :

```
<meta http-equiv="Refresh" content=".5; URL=pg2.html">
```

Vous noterez que la valeur en secondes et l'URL figurent à l'intérieur de la même paire de guillemets. Le code HTML5 suivant lance une série de pages qui continuent à s'actualiser jusqu'au chargement de la page d'accueil :

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta http-equiv="Refresh" content=".5; URL=pg2.html">
<title>Image 1</title>
</head>
<body>
<img src="one.png" alt="one">
</body>
</html>
```

Après le chargement de la page initiale, on a la séquence suivante, chaque page se chargeant l'une après l'autre :

- Page 2 : `<meta http-equiv="Refresh" content=".5; URL=pg3.html">`
- Page 3 : `<meta http-equiv="Refresh" content=".5; URL=pg4.html">`
- Page 4 : `<meta http-equiv="Refresh" content=".5; URL=pg5.html">`
- Page 5 : `<meta http-equiv="Refresh" content=".5; URL=Accueil.html">`

La page d'accueil, `Accueil.html`, n'aura pas de valeur `Refresh` dans la balise `<meta>`. En fait, en dehors des éléments meta avec l'attribut `Content-Type`, il n'y aura pas d'autres balises meta (si la page d'accueil reboucle sur la première page, le rechargement des pages ne s'arrête jamais).

5.1.3 Savoir quand on a besoin d'un script

Plus on utilise HTML5, plus on a besoin de scripts pour tirer le meilleur parti de ses pages Web. JavaScript est le langage de script qui est utilisé le plus couramment avec HTML5. Votre navigateur dispose d'un interpréteur JavaScript, tout comme il en a un pour décoder HTML5. Fort heureusement, JavaScript est facile à apprendre et on peut se contenter d'en écrire que de courts extraits, si bien que même ceux qui ne sont pas développeurs y arrivent.

Pour inclure un script JavaScript, il suffit d'ajouter une balise au début de la page :

```
<script type="text/javascript">
```

Le programme JavaScript vient se placer dans le conteneur `<script>`. Le code suivant HTML5 (`BaliseScript.html` disponible dans les fichiers de ce chapitre) montre combien il est facile d'apprendre JavaScript.

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
alert("Je sais faire du JavaScript !");
</script>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Un avant-goût de JavaScript</title>
</head>
<body>
Page Web normale....
</body>
</html>
```

Quand on teste ce petit programme, on voit s'afficher une boîte de dialogue (illustrée à la figure 5.2).

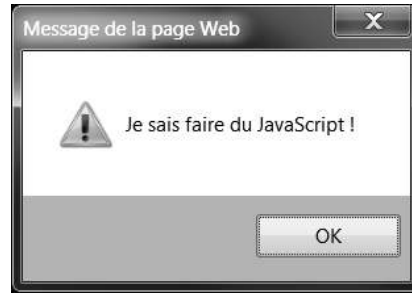


Figure 5.2 — Boîte de dialogue JavaScript.

Vous noterez que la boîte de dialogue JavaScript se charge avant le contenu de la page Web. Cela est dû au fait que le conteneur head se charge en premier. Si vous avez un programme JavaScript plus élaboré qui sera utilisé dans une page HTML5, il

est nécessaire de le tester sur différents navigateurs et également de le placer dans un fichier JavaScript externe. La figure 5.3 illustre une boîte de dialogue similaire chargée dans Safari sur un iPhone et vous pouvez clairement voir que la page Web associée au code HTML5 n'a pas été chargée.

Dès que l'utilisateur clique sur le bouton OK, la page Web se charge. Pendant ce temps, on peut voir les fichiers du répertoire en arrière-plan sur l'écran du terminal mobile. En outre, vous remarquerez que la boîte de dialogue affiche le domaine où se trouve JavaScript. Certains navigateurs, comme Google Chrome, vérifient d'abord si l'utilisateur accepte le code JavaScript issu du site en question avant d'afficher la boîte de dialogue.

Figure 5.3 — Chargement d'une boîte de dialogue avant la page Web.

Comme pour les feuilles de styles, les programmes JavaScript peuvent être chargés à partir de fichiers externes. Cependant, au lieu d'utiliser l'élément `link`, les fichiers JavaScript sont chargés à l'aide de l'élément `script`, comme dans l'exemple suivant :

```
<script type="text/javascript" src="programmeJS.js"></script>
```

Les fichiers JavaScript comportent l'extension `.js`, alors que les fichiers CSS3 ont l'extension `.css`.

Dans la quatrième partie de ce livre, vous verrez qu'il est intéressant d'utiliser JavaScript avec la balise `<canvas>` et d'autres éléments HTML5. De plus, les balises `<script>` et le code JavaScript peuvent être ajoutés en plein au milieu du script HTML5. Il y a cependant un avantage à placer le code JavaScript dans le conteneur head, car il est chargé en premier, avant la page Web.

5.2 UTILISER LES SECTIONS DANS SA CONCEPTION

Les sections sont l'une des modifications majeures de HTML5 par rapport aux anciennes versions de HTML. Avant HTML5, on pouvait facilement représenter des sections avec l'élément `body` et des balises `<h>`. En HTML5, on peut envisager une page avec des sections et des sous-sections. Si l'on agrandit le contexte, une page Web est un `article`, et comme dans un article de magazine, on peut trouver différentes sections qui constituent les blocs de l'article. La figure 5.4 fournit une vue d'ensemble des sections d'une page HTML5.

Figure 5.4 — Sections composant une page.

En examinant la figure 5.4, on peut voir différents blocs d'information, mais les balises utilisées n'ont en général pas de capacité intrinsèque à structurer visuellement l'information. Les balises `<h>`, qui sont des éléments de section, permettent assurément d'afficher le texte dans des tailles différentes, mais les autres balises de section servent tout autant à organiser la page qu'à en spécifier l'aspect visuel.

Voici une liste d'éléments de section :

- `Body`
- `Section`

- Nav
- Article
- Aside
- H1 . . . H6
- Hgroup
- Header
- Footer
- Address

L'élément `body` est la racine des sections, tout comme l'élément `html` est la racine de la page. Au cours des précédents chapitres, nous avons déjà rencontré plusieurs éléments de section qui vous sont ainsi devenus familiers, mais le script suivant va vous aider à comprendre comment les utiliser conjointement (`ArticleStructure.html` qui est disponible dans les fichiers du chapitre).

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv=''Content-Type'' content=''text/html; charset=UTF-8''>
<title>Sections</title>
</head>
<body>
<article>
  <header>
    <h1>Pilots and Planes</h1>
    <p><q>I never left one up there. </q><i>Ace Davis</i></p>
  </header>
  <nav><a href=''#''> Safety</a> | <a href=''#''>Check Lists</a> | <a
href=''#''>Landings</a></nav>
  <section>
    <h2>Flying Stories by Real Pilots</h2>
    <h3>...and other cures for insomnia.</h3>
    <section>
      <h4>Short Final</h4>
      <p>As we were on short final, control cleared the Maule for immediate
takeoff, which it did in about 15 feet of runway at an airspeed of 20 mph. It
filled my windshield as I approached stall speed. After realizing its mistake,
the tower instructed the Maule to loop, and we were able to land without
incident.</p>
    </section>
    <section>
      <h4>Thermal on Takeoff</h4>
      <p>Taking off from Gila Bend, Arizona, with the ambient temperature of
130 F, we encountered a strong thermal at the end of the runway, which took
our Cessna 177b to 15,000 feet in 12 seconds flat, at which time we leveled
off and proceeded to New Mexico via the jet stream, setting a new speed
record.</p>
    </section>
  </section>
  <aside>
    <h2>Truthful Pilot Found!</h2>
```

```
    <p>Emily Rudders, a pilot in Moose Bite, Vermont, was recently found to be
the only truthful pilot in existence. When asked to relate her most exciting
flying adventure, Emily replied, <q>I ain't never flew no airplane. I jus'
shoot at 'em when they fly over and bother the moose.</q></p>
  </aside>
  <footer>
    <address>
    Contact us at:<a href=''www.aopa.org''>AOPA</a>
    </address>
  </footer>
</article>
</body>
</html>
```

Le but des sections est de diviser la page en parties cohérentes. Il s'agit d'un ensemble organisé d'éléments qui peuvent être utilisés pour de la mise en forme, mais ce n'est pas leur objectif principal. Pour formater un paragraphe ou un groupe de paragraphes, la norme W3C encourage l'emploi de la balise `<div>`.

La figure 5.5 illustre l'affichage du script précédent dans un navigateur. Bien que la mise en page ne soit pas particulièrement attirante, elle se révèle fonctionnelle. L'article est consacré aux pilotes et à la navigation aérienne. Le titre de l'article annonce le sujet (les pilotes et les avions) et offre une citation d'un pilote en utilisant une balise `<q>`. Après le titre, la première section traite d'histoires de pilotage. Deux autres balises `<section>` sont imbriquées à l'intérieur de la première section afin de séparer deux histoires.

Une section sur la véracité des histoires de pilotes d'avion est placée dans un conteneur d'élément séparé `aside`. À la figure 5.4, vous avez peut-être remarqué que l'élément aside était placé dans une colonne séparée, mais en soi l'élément `aside` est une référence à la signification de la page, et il ne s'agit donc pas d'un élément de mise en forme en tant que tel.

Pourquoi l'organisation en sections est importante

Vous pensez peut-être que vous pouvez composer une page sans vous embêter avec les balises de section. Cela est vrai, mais en coulisse de votre belle page Web, il y a un moteur qui peut référencer différentes parties de votre page. Connus sous le nom de Document Object Model (DOM), les différents regroupements que vous avez définis avec les éléments de section peuvent être traités comme une hiérarchie d'objets en un flux de données bien organisées circulant sur Internet. En prêtant attention au modèle organisationnel de HTML5, tout le monde sera content : vos pages Web, Internet, et même l'univers !

À la fin de l'article se trouve un pied de page. Les éléments `footer` peuvent en fait aller n'importe où, y compris à l'intérieur des conteneurs d'éléments individuels `section` et `aside`. Un élément `footer` agit comme un élément organisationnel de fermeture des éléments de la section. Au sein du pied de page, il y a un élément `address` avec un lien vers une URL en rapport avec l'article.

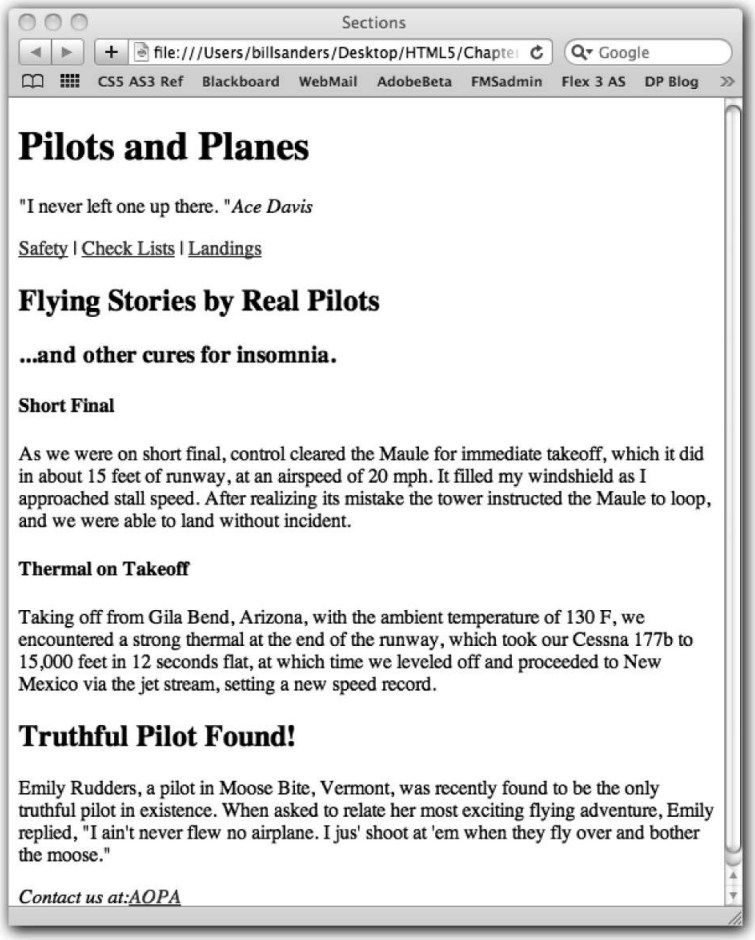


Figure 5.5 — Page organisée avec des sections.

En examinant la page illustrée à la figure 5.5 et son code, vous pouvez voir la signification de la page apparaître dans les balises de section. Comme nous l'avons déjà dit, il ne s'agit pas ici de mise en forme, mais bien d'organiser le sens de la page.

5.3 ORGANISATION DES CONTENUS

Quand vous avez un plan d'organisation général, il est souhaitable de mettre en ordre les contenus au sein des différentes sections. À la figure 5.4, vous avez vu que plusieurs des éléments de section contenaient des éléments de regroupement, comme les balises `<p>`. Les éléments de regroupement constituent l'emplacement idéal pour l'ajout des styles CSS3, ce qui n'est pas le cas des éléments de section. Nous allons ici étudier les principaux éléments qui vous aideront à organiser vos pages Web.

5.3.1 Paragraphes, divisions et listes

Les balises `<p>` et `<div>` étaient largement employées dans les pages HTML pour les regroupements et les styles. Si ces balises sont toujours importantes, vous devez cependant garder à l'esprit qu'elles ne servent plus à intégrer les contenus dans des sections sur la page, mais à regrouper les différentes parties d'une section. Par exemple, l'extrait de code suivant montre comment on utilisait ces deux balises avant HTML5 :

```
<div>
  <h1>Les choses importantes de la vie</h1>
  <p>
  <h2>Trouver le véritable amour</h2>
  </p>
  <p>
  <h2>Choisir un bon métier</h2>
  </p>
  <p>
  <h2>Trouver une place de parking</h2>
  </p>
</div>
```

Ce code fonctionne parfaitement en HTML5, mais il est préférable de l'organiser en utilisant les éléments qui conviennent le mieux pour cette tâche. L'approche suivante est par conséquent meilleure :

```
<header>
  <h1>Les choses importantes de la vie</h1>
</header>
<section>
  <h2>Trouver le véritable amour</h2>
  <h2>Choisir un bon métier</h2>
  <h2>Trouver une place de parking</h2>
</section>
```

Quand on visualise la page Web, le résultat des deux versions du code est identique, mais il est plus judicieux d'employer le code HTML5 avec les nouveaux éléments de section.

La question suivante est de savoir où l'on peut utiliser les éléments `p` et `div`. En fait, il n'est pas souhaitable d'employer abondamment ces balises, même si elles peuvent se révéler pratiques quand on veut ajouter un élément de style ou un autre attribut au milieu d'une balise `<article>` ou `<section>`. Examinez le code suivant (`UtilisationDiv.html` qui est disponible dans les fichiers du chapitre).

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
body {
font-family:"Comic Sans MS", cursive;
color:#0C6;
```

```
background-color:#FFC;
}
.girls {
background-color:pink;
}
.boys {
background-color:powderblue;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Prénoms des enfants</title>
</head>
<body>
<article>
<header>
  <h1>Prénoms des enfants</h1>
</header>
<section>
  <div class="girls">
    <h2> Filles</h2>
    <ul>
      <li>Olivia</li>
      <li>Prune</li>
      <li>Emilie</li>
    </ul>
  </div>
</section>
<section>
  <div class="boys">
    <h2> Garçons</h2>
    <ul>
      <li>Jacques</li>
      <li>Richard</li>
      <li>John</li>
    </ul>
  </div>
</section>
</body>
</html>
```

La figure 5.6 illustre le résultat, mais le point important est que la balise `<div>` n'a été employée que pour fournir les couleurs de fond des deux différents éléments `<section>`.

Comme vous pouvez le voir dans le listing, l'élément `div` a permis l'emploi de deux styles de couleur de fond différents dans les conteneurs `section` sans avoir à ajouter de classes à la balise `<section>`. Vous devez cependant garder à l'esprit que dans l'ensemble `<p>` et `<div>` sont des éléments plus génériques et que vous devez privilégier les éléments qui décrivent le mieux le contenu de votre page Web.

Outre l'utilisation de la balise `<div>` pour le regroupement et le stylage, on peut aussi employer les listes pour faire ressortir les données. HTML5 utilise toujours les balises `<ul>` pour regrouper les prénoms des filles et des garçons, mais il existe une différence subtile, mais importante entre les listes ordonnées (`<ol>`) et les listes non ordonnées (`<ul>`).

Figure 5.6 — Utilisation de la balise `<div>` pour le stylage.

L'utilisation de ces listes dépend du contexte. Par exemple, lors de la coupe du monde de football en 2010 en Afrique du sud, les équipes de l'Allemagne, des Pays-Bas, de l'Espagne et de l'Uruguay étaient qualifiées. Si l'on voulait lister ces équipes au début de la compétition, on pouvait utiliser une liste non ordonnée. À la fin de la compétition, on pouvait employer une liste ordonnée pour afficher le résultat final. La page Web suivante (`liste.html` qui est disponible dans les fichiers du chapitre) reflète ces différents regroupements en fonction du contexte et de la signification qui est attachée au contexte.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
/*20268C,0C080C,2F8C2B,F27507,F20505 */
body {
background-color:#2F8C2B;
color:#0C080C;
font-family:Verdana, Geneva, sans-serif;
}
h2 {
background-color:#F27507;
color:#20268C;
font-family:"Comic Sans MS", cursive;
}
h3 {
font-family:"Comic Sans MS", cursive;
}
ol {
background-color:#F27507;
}
```

```
ul {
background-color:#F20505;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Liste ordonnée et non ordonnée</title>
</head>
<body>
<h2> Coupe du monde 2010</h2>
<h3>Début</h3>
<ul>
  <li>Espagne</li>
  <li>Pays-Bas</li>
  <li>Allemagne</li>
  <li>Uruguay</li>
</ul>
<h3>Fin</h3>
<ol>
  <li>Espagne</li>
  <li>Pays-Bas</li>
  <li>Allemagne</li>
  <li>Uruguay</li>
</ol>
</body>
</html>
```

Comme vous pouvez le constater à la figure 5.7, la signification du groupe au début de la coupe du monde n'a pas de hiérarchie car il ne s'agit que d'une liste de quatre équipes participant à cette coupe. Cependant, à la fin de la coupe, l'ordre a de l'importance de telle sorte que la liste ordonnée est la plus appropriée.

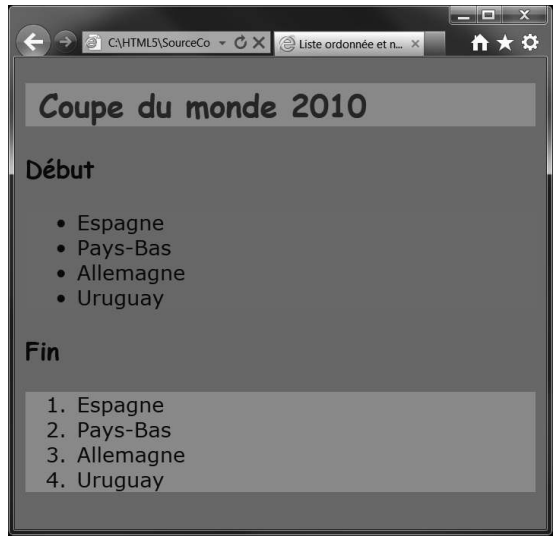


Figure 5.7 — Les listes ordonnées et non ordonnées ont des significations différentes.

Vous noterez également que les deux différentes sortes de listes ont des couleurs de fond différentes qui ont été ajoutées avec CSS3. Cela signifie que lorsque l'on emploie des éléments de regroupement, on peut également accentuer le rassemblement du contenu par l'utilisation des couleurs, comme cela a été illustré dans les figures 5.6 et 5.7.

5.3.2 Regroupement sans fracture

La balise `<hr>` est l'un des éléments de regroupement qu'il n'est pas souhaitable d'utiliser, hormis pour le regroupement de l'en-tête avec le reste de la page, quand cela est nécessaire. L'élément `hr` (qui signifie *horizontal rule*, c'est-à-dire règle horizontale) n'est qu'une ligne, mais il doit être utilisé judicieusement et avec parcimonie. Prenons, par exemple, l'extrait suivant du poème « Kubla Khan » de Samuel Taylor Coleridge :

In Xanadu did Kubla Khan
A stately pleasure-dome decree:
Where Alph, the sacred river, ran
Through caverns measureless to man
Down to a sunless sea.

So twice five miles of fertile ground
With walls and towers were girdled round;
And there were gardens bright with sinuous rills,
Where blossomed many an incense-bearing tree;
And here were forests ancient as the hills,
Enfolding sunny spots of greenery.

But oh! that deep romantic chasm which slanted
Down the green hill athwart a cedarn cover!
A savage place! as holy and enchanted
As e'er beneath a waning moon was haunted
By woman wailing for her demon-lover!

Les trois strophes sont séparées par un simple espace double, ainsi que le titre. Cependant, si des balises `<hr>` sont insérées, comme dans le listing suivant (`HR.html` qui est disponible dans les fichiers du chapitre), vous constaterez que le résultat est assez différent et dénature le sens du poème.

```
<!DOCTYPE HTML>
<html>
<head>
```

```
<style type=''text/css''>
/*A1A680,D9D7BA,D90404,8C0303,590202 */
body {
     background-color:#A1A680;
    color:#590202;
    font-family:''Palatino Linotype'', ‘‘Book Antiqua'', Palatino, serif;
     font-size:8px;
}
h4 {
     background-color:#D9D7BA;
    color:#8C0303;
    font-family:Tahoma, Geneva, sans-serif;
}
</style>
<meta http-equiv=''Content-Type'' content=''text/html; charset=UTF-8''>
<title>Trop de balises HR</title>
</head>
<body>
<header>
  <h4> Kubla Khan</h4>
</header>
<article>
  <hr>
  In Xanadu did Kubla Khan<br>
  A stately pleasure-dome decree:<br>
  Where Alph, the sacred river, ran<br>
  Through caverns measureless to man<br>
  Down to a sunless sea.<br>
  <hr>
  So twice five miles of fertile ground<br>
  With walls and towers were girdled round;<br>
  And there were gardens bright with sinuous rills,<br>
  Where blossomed many an incense-bearing tree;<br>
  And here were forests ancient as the hills,<br>
  Enfolding sunny spots of greenery.<br>
  <hr>
  But oh! that deep romantic chasm which slanted<br>
  Down the green hill athwart a cedarn cover!<br>
  A savage place! as holy and enchanted<br>
  As e'er beneath a waning moon was haunted<br>
  By woman wailing for her demon-lover! </article>
</body>
</html>
```

Comme vous pouvez le voir, les balises `<hr>` sont toutes au sein de l'élément `article`, alors que le titre fait partie de l'élément `header`. La figure 5.8 montre que les lignes horizontales ne clarifient pas le sens du poème et ont plutôt tendance à le fragmenter.

Quand votre page comporte une division importante, une ligne horizontale peut être appropriée, mais il faut alors ajouter un style CSS3 pour alléger l'élément `hr` de telle sorte qu'il apporte une note subtile (on peut notamment ajouter de la transparence). Les bons concepteurs savent utiliser les lignes horizontales avec modération, alors que les néophytes peuvent très facilement mettre la pagaille dans leurs pages Web en abusant des balises `<hr>`.

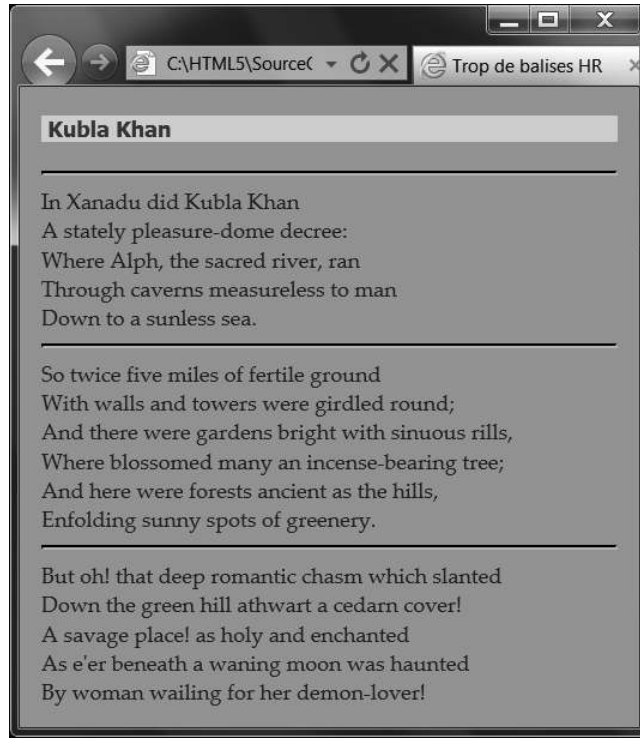


Figure 5.8 — Les lignes horizontales peuvent fragmenter la signification.

5.3.3 Figures et légendes

L'utilisation combinée en HTML5 des balises `<figure>` et `<figcaption>` est très frustrante. En plaçant un élément `figcaption` à l'intérieur d'un conteneur d'élément `figure`, on peut légitimement présupposer qu'ils forment un objet unique pour la mise en forme et la conception. L'élément `figcaption` est considéré comme un enfant de l'élément `figure` quand `figcaption` est imbriqué à l'intérieur d'un élément `figure`. Cependant, cela ne signifie pas qu'ils apparaîtront ensemble sur la page. En fait, l'alignement d'une figure avec sa légende peut se révéler délicat.

Dans les mises en forme CSS3 plus sophistiquées, la figure et sa légende peuvent être traitées comme un objet avec une relation parent-enfant. Ce n'est pas parce `figure` et `figcaption` font partie des éléments de regroupement de HTML5 que cela signifie qu'ils sont mis en forme conjointement sur la page. Au contraire, cela signifie qu'ils peuvent être référencés comme un flux unique du contenu principal de la page. Vous devez donc travailler prudemment quand vous utilisez ensemble ces deux éléments, comme cela est illustré dans le programme HTML5 suivant (`Figure_legende.html` qui est disponible dans les fichiers du chapitre) où la légende référence une image stylisée.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
/* 732D3F,A66879,D9C3B0,260101,F2F2F2 */
body {
background-color:#D9C3B0;
color:#732D3F;
font-family:Verdana, Geneva, sans-serif;
font-size:11px;
}
aside {
margin-left:260px;
}
h1 {
font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;
background-color:#F2F2F2;
color:#A66879;
text-align:center;
}
figcaption {
color:#A66879;
background-color:#F2F2F2;
}
img {
margin:5px;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Regroupement d'une figure et de sa légende</title>
</head>
<body>
<header>
  <h1>Souvenirs de Baja</h1>
</header>
<article>
  <figure> <img src="PuntaBufeo250.png" alt="Punta Bufeo"><br>
    <figcaption> Piste d'atterrissage sur la plage de Punta
Bufeo </figcaption>
  </figure>
  <section>
    <p>Les meilleures destinations de Baja ne sont accessibles qu'en 4x4 ou
bien en petit avion. Les plages y sont immaculées, désertes et spacieuses. La
<i>mer de Cortez</i> (appelée aussi <i>golfe de Baja</i> et la <i>mer
Vermillion</i>) sont d'un bleu transparent.</p>
  </section>
</article>
</body>
</html>
```

Vous pouvez commencer à réfléchir aux éléments et à leurs descendants. Dans ce cas, l'élément `figcaption` est un descendant de l'élément `figure`. La figure 5.9 illustre la légende sous l'image, les deux étant à l'intérieur d'un conteneur `<figure>`.

Comme vous pouvez le voir clairement à la figure 5.9, l'élément `<figcaption>` est stylé différemment, même s'il s'agit d'un descendant du conteneur `<figure>`. Cependant, vous ne pouvez pas présupposer qu'un élément `figcaption` sera correctement positionné, comme c'est le cas à la figure 5.9, simplement parce que c'est un enfant de l'élément `figure` qu'il légende.

Figure 5.9 — `Figure` et `figcaption` utilisés avec une image.

5.4 ORGANISATION DES FICHIERS

Avec un site Web simple, l'organisation des fichiers est facile. Au fur et à mesure que la complexité d'un site s'accroît, particulièrement si plusieurs développeurs et concepteurs sont impliqués dans le projet, il faut organiser le site en répertoires séparés et parfois même en serveurs séparés. Dans cette section, vous allez étudier plusieurs problèmes organisationnels et apprendre à gérer l'agencement des fichiers et leur accès.

5.4.1 Organisation et référencement des images

Un site Web typique comporte un ou plusieurs dossiers (répertoires) consacrés aux fichiers image ou aux types de fichiers image. Jusqu'à présent, dans la plupart des exemples de ce livre, nous n'avons pas utilisé des dossiers séparés pour les images et les pages HTML5 qui les chargent ; au lieu de cela, toutes les images étaient placées dans le même répertoire que les fichiers HTML5. Quand on a un grand nombre de pages

Web et d'images à charger, il est plus efficace d'organiser le site avec des répertoires séparés pour les différents groupes de fichiers. La manière dont on organise ses images dépend d'un certain nombre de facteurs. Voici quelques exemples de classifications en répertoires et sous-répertoires :

- Classifications formelles (Animaux > Mammifères > Rongeurs)
- Sujet (Vacances > Destinations > Logement > Affaires à emporter)
- Processus (Pâtisserie > Fabrication de la pâte > Préchauffage du four > Temps de cuisson)

Quel que soit le modèle organisationnel choisi, vous devez comprendre comment accéder à vos images. Toutes les références aux images sont absolues ou relatives.

5.4.2 Référence absolue

Toute référence à une image s'effectue par le biais d'une URL, qu'il s'agisse d'un listing complet de l'adresse ou simplement du nom du fichier. Une adresse absolue commence par `http://` et comprend le chemin complet du fichier HTML5. Voici, par exemple, une adresse absolue d'un fichier :

```
http://www.smashinghtml5.com/organization/graphics/faces.html
```

Quel que soit l'emplacement à partir duquel cette URL est appelée, le fichier nommé à la fin de l'URL est reconnu. Cela est identique avec une référence source (`src`) à une image. Si votre code comporte le lien suivant, il chargera le fichier `nose.png` quel que soit l'emplacement de la page Web qui référence l'image :

```
<img src="http://www.smashinghtml5.com/organization/graphics/nose.png">
```

Même si la page Web qui référence l'image est sur un serveur totalement différent, l'adresse absolue fonctionne.

L'avantage de l'emploi des adresses absolues est que vous n'avez pas à vous préoccuper si une page figure bien sur votre site Web. Vous n'avez pas non plus à vous préoccuper si elle figure sur le même serveur. Ce type d'organisation n'est cependant pas optimal et les longues URL ne sont pas faciles à gérer.

5.4.3 Référence relative

Une référence relative est relative par rapport à la position de la page appelante sur le site Web ou à sa base qui a été définie. Sur votre ordinateur, votre page Web a une position `file` au lieu d'une position `http`. Par exemple, l'adresse suivante est la position absolue du fichier `somePage.html` sur mon ordinateur :

```
"file:///Macintosh HD/Users/billsanders/Desktop/HTML5/somePage.html"
```

Si j'ai une image dans le dossier `HTML5/`, je peux utiliser son adresse relative pour l'appeler à partir de `somePage.html`. Par exemple, si j'ai un fichier `anyGraphic.png` dans le dossier `HTML5`, je peux utiliser simplement la référence relative suivante :

```
<img src="anyGraphic.png">
```

Mais si je veux organiser mes images dans un dossier séparé appelé `images`, à l'intérieur du dossier `HTML5`, j'utiliserai alors l'adresse relative :

```
<img src="images/anyGraphic.png">
```

Vous pouvez créer autant de niveaux relatifs que vous le souhaitez, chaque niveau étant séparé par un slash (`/`). Par exemple, on peut imaginer une arborescence plus complexe :

```
<img src="images/animaux/chiens/SaintBernard/monToutou.png">
```

Si l'on peut faire référence à des niveaux inférieurs, on peut aussi accéder aux niveaux supérieurs. Supposons par exemple que l'on dispose du chemin suivant et que la page HTML5 soit dans le répertoire `dossierBase` :

```
dossierHaut/dossierMilieu/dossierBase
```

Pour accéder à une image située dans le répertoire `dossierMilieu`, il faut utiliser la syntaxe suivante :

```
<img src=".../anyGraphic.png">
```

Si l'image est dans le répertoire `dossierHaut`, il faut utiliser le format suivant :

```
<img src="../../anyGraphic.png">
```

On n'a pas besoin de nommer le dossier cible dans lequel se situe la page Web appelante, mais on utilise les caractères `../` successifs jusqu'à ce que l'on atteigne le niveau désiré. Cela signifie que l'on peut remonter au niveau souhaité et redescendre vers une autre branche. Dans l'exemple suivant, on remonte jusqu'au répertoire `dossierHaut`, puis à l'intérieur de `dossierHaut` on redescend jusqu'au dossier qui contient les images pour atteindre l'image cible :

```
<img src="../../images/anyGraphic.png">
```

La figure 5.10 fournit une illustration générique de l'accès aux ressources situées dans des dossiers de niveaux inférieurs et supérieurs.

Comme cela a été noté dans la section « Définition de la base de référence », plus haut dans ce chapitre, votre position relative peut être définie à un emplacement qui

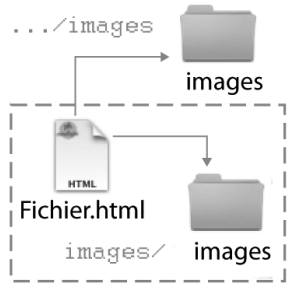


Figure 5.10 — Chemins relatifs.

est différent de celui où se situe le fichier. Examinez, par exemple, les deux pages Web suivantes (`Terre.html` et `Alien.html` qui sont disponibles dans les fichiers du chapitre). La première appelle la seconde sur un serveur différent ; cependant, comme la base de la première page est définie sur le second serveur, l'appel est relatif. Le premier fichier, qui se nomme `Terre.html`, est situé sur le domaine `smashingHTML5.com` dans le dossier `smashing`. Sa base est pourtant définie à `smashingHTML5.net` dans le dossier `smashing`. Il est donc possible d'utiliser une URL relative pour accéder au fichier `Alien.html` sur un serveur totalement différent.

Base définie sur un serveur différent

```
<!DOCTYPE HTML>
<html>
<head>
<base href="http://www.sandlight.net/html5/smashing/"><meta
http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Terre</title>
</head>
<body>
<h1>Ici la terre</h1>
<a href="Alien.html">Mise à feu !</a>
</body>
</html>
```

Page Web sur un serveur différent

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Planète Smashing</title>
</head>
<body>
<h1>Page d'un serveur alien</h1>
</body>
</html>
```

Même si le domaine de la première page (`Terre.html`) est `smashingHTML5.com`, la base est définie à `smashingHTML5.net`. Cela a pour effet qu'un lien relatif à `Alien.html`, qui réside sur `smashingHTML5.net`, est réalisé sans avoir à utiliser une adresse absolue.

5.5 À VOUS DE JOUER !

Dans la première section de ce chapitre, vous avez vu comment utiliser l'état `Refresh` pour changer automatiquement des pages. Pour vous amuser avec les animations et l'état `Refresh`, jetez un coup d'œil sur le lien relatant les travaux d'Eadweard Muybridge :

```
http://138.23.124.165/collections/permanent/object_genres/
photographers/muybridge/contents.html#
```

Muybridge est intéressant car en 1878 il a réussi à créer des films en utilisant une série de photographies. L'université Riverside de Californie a conservé les travaux de Muybridge et les a mis en ligne en utilisant des fichiers GIF animés. Pour voir comment on peut réaliser des animations en utilisant le rafraîchissement des pages, téléchargez l'un des fichiers GIF animés de la collection Muybridge à partir du lien ci-dessus et extrayez les douze photographies individuelles du fichier GIF. Vous pouvez extraire ces images avec Adobe Photoshop, Adobe Fireworks, et plusieurs autres programmes (cherchez « extraire images GIF animé » dans un moteur de recherche pour trouver de nombreuses façons de récupérer les images individuelles. Si vous avez un Mac, vous pouvez utiliser l'application Prévisualisation et simplement faire glisser les images individuelles dans un dossier séparé).

Une fois que vous avez extrait les fichiers GIF individuels, concevez votre animation en utilisant l'état Refresh avec l'élément `meta` dans la section `<head>` de votre programme. Pour commencer, utilisez le script HTML5 suivant (`an1.html` qui est disponible dans les fichiers du chapitre).

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta http-equiv="Refresh" content="0.1; URL=an2.html">
<title>Image 1</title>
</head>
<body>
<img src="an1.png" alt="one">
</body>
</html>
```

Les fichiers GIF individuels ont été sauvegardés sous la forme de fichier `.png` et renommés `an1.png` jusqu'à `an12.png` (le préfixe `an` signifie animation). De la même manière, les douze fichiers HTML ont été nommés en utilisant le préfixe `an`, de `an1.html` à `an12.html`. Une fois que vous aurez terminé, vous obtiendrez un cheval qui avance au pas. Si vous liez la douzième page à la première, le cheval n'arrête pas de se déplacer.

6

Affichage des données dans des tableaux

Objectif

Aux débuts de HTML, l'élément `table` était utilisé pour la majeure partie de la mise en forme des pages. L'avènement de CSS a introduit un nouvel ensemble de règles de mise en forme et l'élément table a été abandonné comme outil de mise en forme, ce qui était justifié. Cependant, CSS3 a réintroduit certains types spécifiques de mise en forme, si bien que les tableaux, même s'ils ne sont toujours pas des outils génériques de mise en forme, ont désormais des fonctions intéressantes pour afficher des ensembles de données et pour gérer la mise en forme générique CSS3.

Ce chapitre étudie les nouvelles propriétés CSS3 que vous pouvez utiliser pour accomplir des mises en pages générales, mais il se concentre principalement sur l'affichage des données tabulaires. *Les données tabulaires* ne sont rien d'autre que des données disposées dans un tableau pour en faciliter la lecture et pas spécialement des structures de mise en page.

6.1 PROPRIÉTÉS DE TABLEAUX CSS3 POUR HTML5

Dans une déclaration du W3C (World Wide Web Consortium), l'organisme officiel qui définit les standards du Web et notamment HTML5, il est stipulé de manière catégorique que « les tableaux ne doivent pas être utilisés pour faciliter la mise en page ». Puis, dans une note qui suit cet avertissement, le même document indique « qu'il y a de nombreuses alternatives à l'utilisation des tableaux HTML pour la mise en page, et notamment le positionnement CSS et le modèle de tableau CSS ».

Cela signifie qu'en général, les éléments table ne doivent pas être utilisés pour la mise en page, si ce n'est pour les données tabulaires. Cependant, si vous avez besoin de mettre en page des tableaux, il faut utiliser les propriétés de tableaux CSS3.

Cet avertissement est dû au fait que lorsque CSS est sorti, toutes les mises en page étaient censées être réalisées à l'aide de CSS. Afin de ne pas dissuader les concepteurs et les développeurs d'utiliser les propriétés de tableaux CSS3, le W3C a ajouté cette note. Si vous avez respecté cette mise en garde et n'avez pas utilisé les éléments table dans vos mises en page, nous vous confirmons que les propriétés de tableaux CSS3 sont parfaites pour la conception, jusqu'à un certain point.

Afin d'examiner les fonctionnalités CSS3, la première étape consiste à étudier les valeurs de la propriété CSS3 `display` : `table` et `table-cell`. Cette propriété peut être envisagée comme une instruction de mise en page dont les valeurs organisent la disposition de l'affichage. Il peut être utile de se représenter la propriété table comme un grand conteneur et la propriété table-cells comme les cellules individuelles de ce conteneur. Pour les conceptions plus sophistiquées, table-cells se rapproche de l'élément table et de tous les problèmes qui lui sont inhérents. Il convient donc de l'utiliser pour les applications simples où il suffit de quelques colonnes pour réaliser une tâche peu complexe.

La syntaxe CSS3 pour définir l'affichage utilise des classes prédéfinies, une classe utilisateur, ou un ID. On assigne à la propriété display une simple valeur table ou table-cell. Voici un exemple (avec une définition de style) :

```
.story {
     display: table;
}
.col1 {
     display: table-cell;
     width: 250px;
     padding-right: 20px;
     color:#cc0000;
}
```

La classe `story` définit simplement la propriété display en tant que table. La classe col1, que vous pouvez placer à l'intérieur du tableau, est affichée en tant que table-cell, et il est utile de se la représenter en tant que telle. Le code suivant (`AfficheTableau.html` disponible dans les fichiers du chapitre) montre comment définir une conception qui peut être utilisée pour afficher du texte et des images dans deux colonnes.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
body {
font-family:Verdana, Geneva, sans-serif;
font-size:12px;
}
h1 {
```

```
font-family:"Arial Black", Gadget, sans-serif;
width:520px;
text-align:center;
color:#005500;
}
.story {
display: table;
}
.col1 {
display: table-cell;
width: 250px;
padding-right: 20px;
color:#cc0000;
}
.col2 {
display: table-cell;
width: 250px;
color:blue;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Tableau avec la propriété Display</title>
</head>
<body>
<header>
  <h1>Coupe du monde de football 2010</h1>
  <div  class="col1"><img src="cupImages/us.gif"></div>
  <div class="col2"><img src="cupImages/england.gif"></div>
</header>
<br>
<article class="story">
  <section class="col1">Pendant la coupe du monde de football 2010 en Afrique
du sud, chaque pays était représenté par une équipe. L'équipe des États-Unis
était composée de joueurs venant d'états où l'on joue au football depuis une
quarantaine d'années car ce sport n'est pas aussi populaire aux USA que dans le
reste du monde. Quoi qu'il en soit, l'équipe des USA a bien joué et terminé
première de sa poule.</section>
  <section class="col2">Le Royaume-Uni était l'une des rares nations à être
représentée par plusieurs pays pendant la coupe du monde. L'Angleterre qui
était représentée par le drapeau de Saint Georges (et non pas l'Union Jack) est
arrivée à égalité avec les États-Unis dans la première phase de la coupe.
Comme les USA, l'Angleterre s'est qualifiée pour le second tour, mais n'est pas
arrivée à passer le tour suivant.</section>
</article>
</body>
</html>
```

La classe `story` est un conteneur pour ordonner différentes sections à qui on assigne les classes `col1` ou `col2`. Cependant, les classes `col1` et `col2` n'ont pas à être placées dans un tableau. Vous noterez que les deux images (placées dans deux classes différentes table-cell) sont définies avec des balises `<div>` au sein d'un conteneur `<header>`. Elles sont ensuite utilisées une nouvelle fois à l'intérieur du conteneur `<article>` qui a été assigné à la classe `story` (table). Les deux sections ont été définies en tant que propriétés display de `col1` et `col2`, et bien qu'elles ne soient pas visibles dans les

conteneurs des deux images, vous pouvez voir que le texte de couleur différente aide à différencier leur état. La figure 6.1 illustre le résultat attendu dans un navigateur.

Figure 6.1 — Utilisation de la propriété CSS3 `display` avec la valeur table.

Comme vous pouvez le constater à la figure 6.1, l'utilisation de table-cells est un moyen simple de définir plusieurs colonnes. Quand on développe des sites Web plus sophistiqués, il est souhaitable d'employer des définitions display CSS3 plus avancées qui vont au-delà de table et table-cells, mais la propriété `table` de CSS3 est toujours disponible quand vous en avez besoin.

6.2 TABLEAUX ET DONNÉES TABULAIRES

Tout en gardant à l'esprit qu'il ne faut pas utiliser les balises standard table pour la conception des sites Web, nous allons étudier dans cette section la manière d'employer des tableaux pour afficher des données tabulaires. Les données tabulaires peuvent représenter n'importe quoi, des ensembles de nombres jusqu'aux images, en passant par du texte descriptif. Si vous avez déjà commandé quelque chose en ligne, il y a de grandes chances pour que les articles aient été listés dans un format tabulaire. Habituellement, on trouve une description de l'article, le numéro de l'article et son prix.

La clé des données tabulaires est qu'elles sont représentées sous la forme de lignes et de colonnes afin d'afficher les informations selon des catégories courantes. De plus, l'objectif d'un tableau est de clarifier l'information, de telle sorte que l'utilisateur puisse trouver ce dont il a besoin.

6.2.1 Bases d'un tableau

Les éléments de base d'un tableau sont :

- Le tableau lui-même, `<table>`
- Les lignes du tableau, `<tr>`
- Les cellules du tableau, `<td>`
- Les en-têtes du tableau, `<th>`

En général, une légende de tableau `<caption>` est utilisée au sommet du tableau. Un tableau clair se compose habituellement de titres de lignes de colonnes nettement identifiés. La cellule en haut à gauche du tableau est souvent laissée vide, de telle sorte que la première colonne n'a pas d'en-tête, mais la norme indique qu'aucune cellule ne doit être vide. C'est la raison pour laquelle la cellule du coin de `TableauDeBase.html` contient "l/c" afin de combler l'espace. L'exemple suivant illustre les éléments de base d'un tableau simple. Les lignes et les colonnes ont des titres et les cellules de données représentent les données placées dans les lignes et les colonnes étiquetées.

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Tableau de base</title>
</head>
<body>
<table>
  <caption>
  Lignes et colonnes d'un tableau
  </caption>
  <tr>
    <td>l/c
    <th>Colonne 1
    <th>Colonne 2
    <th>Colonne 3
  <tr>
    <th>Ligne 1
    <td>données a
    <td>données b
    <td>données c
  <tr>
    <th>Ligne 2
    <td>données x
    <td>données y
    <td>données z
</table>
</body>
</html>
```

Un des aspects les plus intéressants des balises de tableau est que les balises de fermeture sont optionnelles, et il n'y a aucune recommandation en la matière. Il est important de mettre en forme le code de telle sorte que les lignes ressortent clairement afin de pouvoir identifier clairement la structure du tableau. Si l'on n'inclut pas

les balises de fermeture des cellules, le code paraît bien plus lisible, ce qui est une excellente chose. Nous allons donc abandonner les balises de fermeture des cellules, à moins qu'elles n'aient un réel intérêt pour la clarté du listing. La figure 6.2 illustre le résultat du programme dans un navigateur.

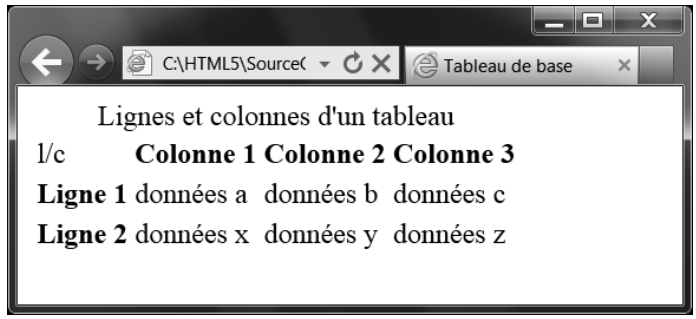


Figure 6.2 — Tableau de base.

Vous noterez qu'alors que la balise `<th>` provoque l'affichage du texte en gras, ce n'est pas le cas de la balise `<caption>`. Ce problème peut être réglé avec CSS3 qui permet aussi d'améliorer le reste du tableau. Mais pour l'instant, vous n'avez besoin de comprendre que les éléments de base des tableaux.

6.3 STYLAGE D'UN TABLEAU

On n'utilise pas les tableaux pour des tâches de stylage général, mais cela ne signifie pas que l'on doit ignorer les styles des tableaux. Il y a une bonne nouvelle en HTML5 : les bordures des tableaux ne sont pas présentes par défaut, comme c'était le cas dans les versions précédentes de HTML. En fait, l'attribut de bordure de tableau n'est plus supporté en HTML5. Si vous voulez encadrer les cellules, vous devez le faire vous-même en utilisant CSS3. Selon de nombreux concepteurs, il faut réaliser les bordures avec beaucoup de soin, sinon il vaut mieux y renoncer.

6.3.1 Ajout de bordures avec CSS3

Edward Tufte, qui est un concepteur renommé, met en garde contre l'utilisation des bordures qui peuvent perturber l'arrière-plan de telle sorte que les données deviennent illisibles. Bien que les bordures séparent clairement les données tabulaires, des bordures visibles sèment la confusion entre les données, ce qui empêche de les discerner facilement. Afin de voir ce que Tufte veut dire, saisissez le script suivant (`MauvaisesBordures.html` disponible dans les fichiers du chapitre) et examinez le résultat.

```
<!DOCTYPE HTML>
<html>
<head>
```

```
<style type="text/css">
table {
width:400px;
border-style:groove;
border-width:thick;
border-color:#FF5C19;
color:#C00;
font-family:Verdana, Geneva, sans-serif;
font-size:10px;
}
caption {
font-family:Tahoma, Geneva, sans-serif;
font-size:24px;
color:hsl(17, 60%, 40%);
padding:12px;
}
td, th {
border-style:solid;
border-width:thin;
border-color:#000;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Bordures aveuglantes</title>
</head>
<body>
<table>
  <caption>
  Les animaux de la ferme
  </caption>
  <tr>
    <td>&#167;
    <th>Poule
    <th>Chèvre
    <th>Oie
  <tr>
    <th>Cri
    <td>Caquètement
    <td>Béguètement
    <td>Criaillement
  <tr>
    <th>Petit
    <td>Poussin
    <td>Chevreau
    <td>Oison
</table>
</body>
</html>
```

La figure 6.3 illustre le résultat, mais avant de la regarder, examinez le code CSS3 soigneusement. La valeur `&#167` est le code de caractère d'un symbole saisi en utilisant son code plutôt que le clavier. Tous les caractères UTF-8 peuvent être saisis de cette manière et certains symboles, comme les signes supérieur à (`>`) et inférieur à (`<`), doivent être saisis avec cette méthode ; dans le cas contraire, l'analyseur les interprète

comme faisant partie d'une balise. Jetez à présent un coup d'œil à la figure 6.3 pour voir la page avec ses bordures.

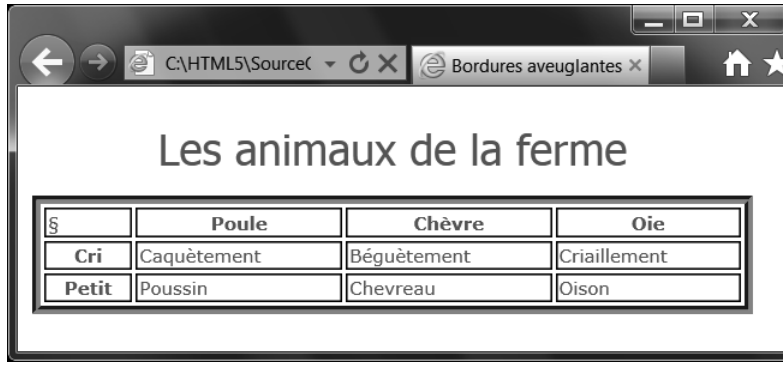


Figure 6.3 — Les bordures peuvent nuire à la clarté des données.

Quand on tente de lire les différents éléments de données, les bordures gênent. Pour régler ce problème, il suffit d'ajouter de l'espace autour du texte des cellules. Pour ce faire, on modifie le style des éléments `td` et `th` de la manière suivante :

```
td, th {
     height:50px;
     border-style:solid;
     border-width:thin;
     border-color:#000;
    padding:20px;
}
```

Nous avons simplement modifié la hauteur de la cellule et l'espace entre la bordure et le texte (padding). Comme vous pouvez le constater à la figure 6.4, ces différences sont significatives.

En ajoutant cet espace autour des données, les valeurs des cellules apparaissent bien plus clairement. Les cellules ne sont pas très jolies, mais il est facile de régler le problème : il suffit de les supprimer !

6.3.2 Mise en évidence des données avec des couleurs de fond

Au début de l'histoire de l'informatique, les ordinateurs imprimaient sur du papier dont la couleur de fond alternait afin de discriminer facilement les enregistrements individuels (dans ce contexte, les lignes et les enregistrements sont des synonymes). Comme vous avez pu le constater, des bordures épaisses autour de chaque cellule nuisent à la clarté des données. C'est la raison pour laquelle les anciennes imprimantes utilisaient des couleurs de fond différentes. Au lieu de séparer les enregistrements par des bordures, il faut donc envisager d'utiliser des couleurs de fond différentes (voir le programme `LignesEnCouleur.html` disponible dans les fichiers du chapitre).

Figure 6.4 — Ajout d'espace dans les cellules d'un tableau.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
td {
width:70px;
}
body {
font-family:Verdana, Geneva, sans-serif;
font-size:10px;
}
caption {
font-family:"Arial Black", Gadget, sans-serif;
font-size:12px;
font-weight:500;
color:#360;
background-color:hsla(113, 46%, 91%, 1);
}
.money {
text-align:right;
}
table {
background-color:#FFC;
}
.alt1 {
background-color:hsla(113, 46%, 91%, .8);
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Séparation des couleurs</title>
</head>
```

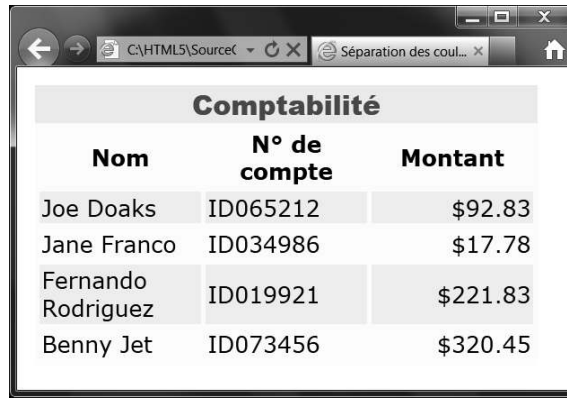
```
<body>
<table>
  <caption>
  Comptabilité
  </caption>
  <tr>
    <th>Nom
    <th>N° de compte
    <th>Montant
  <tr class="alt1">
    <td>Joe Doaks
    <td>ID065212
    <td class="money">$92.83
  <tr>
    <td>Jane Franco
    <td>ID034986
    <td class="money">$17.78
  <tr class="alt1">
    <td>Fernando Rodriguez
    <td>ID019921
    <td class="money">$221.83
  <tr>
    <td>Benny Jet
    <td>ID073456
    <td class="money">$320.45
</table>
</body>
</html>
```

Par défaut, l'élément `td` justifie le texte à gauche, ce qui est souhaitable dans la plupart des cas. Cependant, avec des nombres décimaux, il est préférable d'utiliser une justification à droite. C'est la raison pour laquelle une des classes de la feuille de styles comprend une classe `money` qui justifie à droite les données financières.

La totalité du tableau a une teinte jaune clair comme couleur de fond, mais cette couleur n'affecte pas les données du conteneur `<caption>` ; c'est la raison pour laquelle l'élément `caption` possède une couleur de fond compatible avec celle du tableau. Bien que le tableau soit relativement petit, il a été optimisé pour un affichage sur des terminaux mobiles, si bien que le texte est défini à 10 px (10 px est assez proche d'un texte affiché dans une police en corps 10). La figure 6.5 illustre le résultat.

En utilisant une couleur avec une opacité inférieure à 100 pour cent (c'est-à-dire avec un peu de transparence), le vert qui alterne est légèrement mélangé avec le jaune clair qui sert de couleur de fond. La couleur de fond du titre est identique au vert qui alterne dans les lignes, mais il a une opacité à 100 pour cent et vous pouvez constater qu'il a une teinte légèrement différente. Les éléments `th` ont hérité de la couleur de fond du tableau, mais servent d'étiquettes de colonnes sans aucune autre forme d'ajustement.

Les largeurs des cellules sont définies de manière absolue (70 px) car la largeur reflète le fait que le tableau est optimisé pour la lecture sur un terminal mobile. Cela a pour effet que les noms peuvent occuper deux lignes sans nuire à la lisibilité du tableau.



Comptabilité		
Nom	**N° de compte**	**Montant**
Joe Doaks	ID065212	$92.83
Jane Franco	ID034986	$17.78
Fernando Rodriguez	ID019921	$221.83
Benny Jet	ID073456	$320.45

Figure 6.5 — Alternance des couleurs dans les lignes d'un tableau.

6.4 TABLEAUX COMPLEXES

Quand on parle de tableaux complexes, cela signifie que les tableaux sont difficiles à comprendre. En fait, les tableaux complexes sont des solutions à des problèmes épineux. Si vous utilisez des tableaux pour organiser des données en provenance d'une base de données, il y a de grandes chances pour que vous puissiez employer un tableau standard avec des lignes et des colonnes de taille identique.

Quand on commence à utiliser des tableaux pour afficher des données qui viennent de toutes parts, on rencontre des situations qui vont perturber le bel ensemble de lignes et de colonnes et il faudra alors réaliser des ajustements.

Afin de comprendre les tableaux complexes, vous devez appréhender le concept de cellule. Un tableau n'est rien de plus qu'une collection de cellules ordonnées en lignes et en colonnes. L'intersection d'une colonne et d'une ligne représente la cellule. En HTML5, on crée des cellules en utilisant les balises `<td>` et `<th>`. La figure 6.2, plus haut dans ce chapitre, illustre des cellules de base organisées en lignes et colonnes.

6.4.1 Utilisation des attributs rowspan et colspan

Pour modifier les caractéristiques par défaut d'une cellule d'une intersection entre une ligne et une colonne, vous devez utiliser les attributs d'un élément `td`, `rowspan` et/ou `colspan`. On assigne à chaque attribut un entier positif qui étend une cellule afin qu'elle couvre plusieurs lignes ou plusieurs colonnes. La figure 6.6 illustre un tableau standard composé de cellules de taille identique et un autre tableau dont certaines cellules ont été étendues.

La figure 6.6 montre que la première cellule de la deuxième ligne (Row 2) du tableau du bas occupe l'espace de trois cellules de la deuxième ligne du tableau du haut. Dans la cinquième colonne du tableau du bas, les lignes 1 et 2 ont été réunies en une seule cellule. Vous constaterez que le tableau du haut compte dix cellules alors que le tableau du bas n'en comporte que sept. Quand on code des tableaux avec `rowspan`

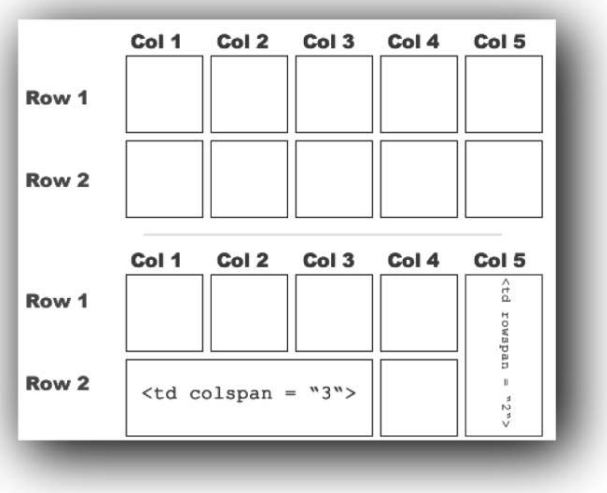


Figure 6.6 — Tableau composé de cellules de taille identique et tableau utilisant les attributs `rowspan` et `colspan`.

et `colspan`, on utilise moins de balises `<td>` (voir le programme `RowSpanColSpan.html` disponible dans les fichiers du chapitre).

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
caption {
font-family:"Arial Black", Gadget, sans-serif;
color:#C60;
}
table {
font-family:Verdana, Geneva, sans-serif;
}
td, tr {
border-style:solid;
border-width:thin;
border-color:#ccc;
width:120px;
padding:5px;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Colspan et Rowspan</title>
</head>
<body>
<table>
  <caption>
  Rowspan et Colspan
  </caption>
  <tr>
```

```
    <td rowspan="2">Ligne A et Ligne B
    <td>Colonne 2a
    <td>Colonne 3a
    <td>Colonne 4a
  <tr>
    <td>Colonne 2b
    <td>Colonne 3b
    <td>Colonne 4b
  <tr>
    <td>Ligne C
    <td>Colonne 2c
    <td colspan="2">Colonne 3c et Colonne 4c
</table>
</body>
</html>
```

Cet exemple utilise une bordure légèrement grisée de manière à mieux visualiser l'emploi des attributs `rowspan` et `colspan`, même si vous n'avez pas besoin d'une bordure pour vous en servir. En fait, s'il n'y a pas de bordures, il peut être difficile de distinguer les regroupements de cellules, si bien qu'elles sont plutôt utiles dans ce cas de figure. La figure 6.7 illustre le tableau avec deux regroupements de cellules.

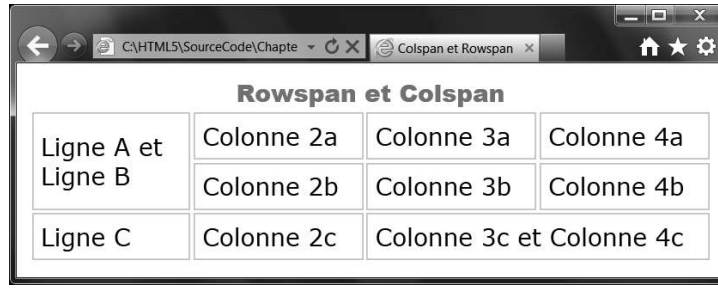


Figure 6.7 — Ajout de regroupements de cellules vertical et horizontal.

Vous pouvez constater que le tableau de la figure 6.7 comporte 10 cellules, alors qu'un tableau de 4 par 3 en comporterait 12. De la même manière, dans le listing, vous pouvez voir dix balises `<td>`. Les attributs `colspan` et `rowspan` peuvent être un peu délicats à comprendre, mais si vous vous les représentez en termes de fusion de cellules, ils sont plus faciles à appréhender.

6.4.2 Mise en pratique dans des tableaux

Quand on crée des tableaux complexes avec `colspan` et `rowspan`, l'exercice peut sembler futile car une application pratique ne paraît pas si évidente. Vous pouvez également envisager l'ensemble du problème avec CSS3 sans utiliser les éléments ou les attributs de la balise `table`. L'exercice suivant repose sur un scénario simple, mais réaliste, où la fusion des cellules est une solution pratique.

Supposons qu'une société de développement Web souhaite gérer ses projets en utilisant des tableaux afin d'en mesurer l'état d'avancement. L'équipe de production est divisée en différents groupes qui représentent chacun une ligne du tableau :

- Coordination (1)
- Conception (4)
- Interaction (2)
- Développement du frontal (3)
- Test (2)

Chaque projet comprend quatre colonnes :

- Tâche à accomplir par le groupe
- Nom du projet
- Constitution des membres de l'équipe
- Échéance de la tâche

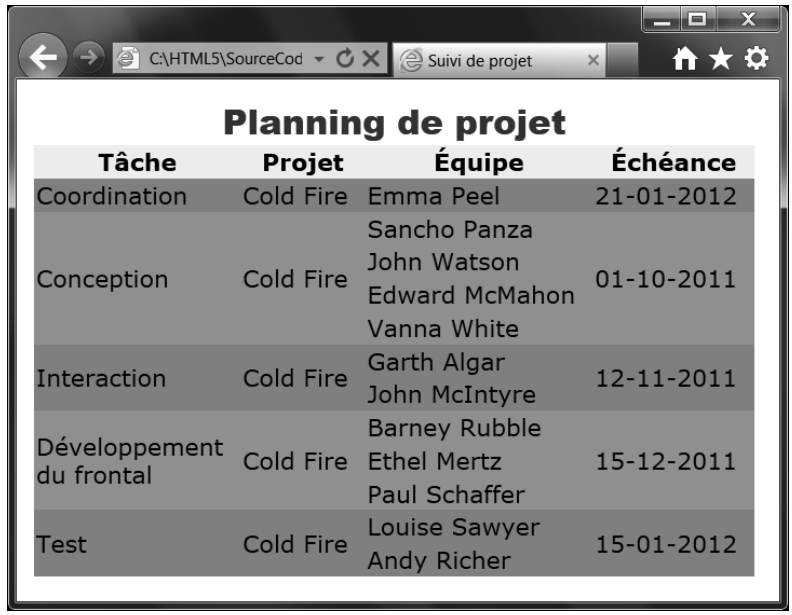
Cela devrait être suffisamment simple pour être compréhensible et suffisamment complexe pour être utile. L'ironie de la réalisation de ce tableau réside dans l'ajout des fusions à l'endroit où il n'y a qu'un seul élément dans la cellule. Cela semble heurter le bon sens car la colonne des membres de l'équipe aura plusieurs lignes à l'intérieur des autres cellules qui ont un attribut `rowspan` de la taille du nombre de membres dans l'équipe. Le programme suivant (`PlanningProjet.html` disponible dans les fichiers du chapitre) illustre la manière dont cela est réalisé.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
/* F2F0E6,595443,A6A08D,3A3F59,8D91A6 */
caption {
font-family:"Arial Black", Gadget, sans-serif;
color:#3A3F59;
}
table {
font-family:Verdana, Geneva, sans-serif;
background-color:#F2F0E6;
padding:5px;
border-collapse:collapse;
}
td, tr {
padding-right:8px;
font-size:11px;
border-collapse:collapse;
}
.bluish {
background-color:#8D91A6;
}
.brownish {
background-color:#A6A08D;
}
```

```
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Suivi de projet</title>
</head>
<body>
<table>
  <caption>
  Planning de projet
  </caption>
  <tr>
    <th>Tâche
    <th>Projet
    <th>Équipe
    <th>Échéance
  <tr class="bluish">
    <td>Coordination
    <td>Cold Fire
    <td>Emma Peel
    <td>21-01-2012
  <tr class="brownish">
    <td rowspan="4">Conception
    <td rowspan="4">Cold Fire
    <td>Sancho Panza
    <td rowspan="4">01-10-2011
  <tr class="brownish">
    <td>John Watson
  <tr class="brownish">
    <td>Edward McMahon
  <tr class="brownish">
    <td>Vanna White
  <tr class="bluish">
    <td rowspan="2">Interaction
    <td rowspan="2">Cold Fire
    <td  rowspan>Garth Algar
    <td rowspan="2">12-11-2011
  <tr class="bluish">
    <td> John McIntyre
  <tr class="brownish">
    <td rowspan="3">Développement<br>
      du frontal
    <td rowspan="3">Cold Fire
    <td>Barney Rubble
    <td rowspan="3">15-12-2011
  <tr class="brownish">
    <td>Ethel Mertz
  <tr class="brownish">
    <td>Paul Schaffer
  <tr class="bluish">
    <td rowspan="2">Test
    <td rowspan="2">Cold Fire
    <td  rowspan>Louise Sawyer
    <td rowspan="2">15-01-2012
  <tr class="bluish">
    <td>Andy Richer
</table>
</body>
</html>
```

Fondamentalement, les balises `<td>` qui incluent un attribut `rowspan` sont celles qui doivent être suffisamment larges pour accueillir le nombre de membres de l'équipe qui seront sur la même ligne. La figure 6.8 illustre le résultat dans un navigateur.

La chose la plus importante qu'il faut se rappeler à propos des tableaux est qu'ils doivent être utilisés judicieusement. Il ne s'agit pas d'outils génériques de mise en page, mais vous pouvez utiliser CSS3 pour concevoir l'aspect des données tabulaires affichées dans des éléments `table`. En conclusion, vous devez absolument utiliser CSS3 pour les données tabulaires, les tableaux et le contenu non tabulaire.



Planning de projet

Tâche	Projet	Équipe	Échéance
Coordination	Cold Fire	Emma Peel	21-01-2012
Conception	Cold Fire	Sancho Panza John Watson Edward McMahon Vanna White	01-10-2011
Interaction	Cold Fire	Garth Algar John McIntyre	12-11-2011
Développement du frontal	Cold Fire	Barney Rubble Ethel Mertz Paul Schaffer	15-12-2011
Test	Cold Fire	Louise Sawyer Andy Richer	15-01-2012

Figure 6.8 — Utilisation de l'attribut `rowspan` dans un tableau.

6.5 À VOUS DE JOUER !

La figure 6.9 illustre le résultat final de l'exercice qui vous est proposé. Le tableau comporte des en-têtes en haut et en bas, et les couleurs de fond des lignes ont une opacité de 20 pour cent et 40 pour cent. Le but du jeu est de dupliquer ce tableau en utilisant HTML5 et CSS3.

Vous pouvez bien entendu adapter ce tableau en modifiant les régions, les villes, les équipes et les lieux à visiter, l'essentiel étant de préserver l'aspect du tableau illustré à la figure 6.9.

City	Teams	To See	City	Teams	To See
Seattle	Seahawks Mariners Sonics	Space Needle	Boston	Patriots Red Sox Celtics	Faneull Hall Back Bay
Portland	Trail Blazers	Rain Trees	New York	Giants Jets Knicks	Museums Broadway Park Ave
San Francisco	49ers Giants Warriors	Golden Gate Marina Grant St. Market St.	Philadelphia	Eagles Phillies 76ers	Lib Bell Indep. Hall Franklin Square
Los Angeles	Dodgers Lakers Clippers	Hollywood Beaches Universal Studios	Miami	Dolphins Marlins Heat	Beaches Jai Lai Scuba Manatees
West Coast			**East Coast**		

Figure 6.9 — Saurez-vous relever ce défi ?

7

Tout sur les liens

Objectif

Outre l'affichage de textes et d'images, l'intérêt majeur des pages Web réside dans la possibilité de charger d'autres pages. En utilisant des pages Web, les gens (ce qui inclut les concepteurs et les développeurs) ont tendance à penser qu'ils *vont quelque part* ou qu'ils *obtiennent quelque chose*. On fournit même aux utilisateurs des cartes des sites et des outils de navigation, ce qui peut leur faire penser qu'ils ont entrepris une sorte de voyage (les problèmes de navigation, qui sont importants, seront traités au chapitre 8).

Ce chapitre est consacré à la manière dont les liens chargent d'autres pages Web, et à la manière dont on les utilise pour accéder à d'autres feuilles de styles. Nous étudierons aussi les différents attributs qui sont liés au chargement des pages, les détails de l'accès à une page, et les propriétés CSS3 utilisées pour styler les liens et gérer des fonctionnalités interactives.

7.1 L'ÉLÉMENT LINK ET SES PRINCIPAUX ATTRIBUTS

Le principal élément de lien est l'élément `a`, raison pour laquelle la majeure partie de ce chapitre lui sera consacrée. Cependant, avant d'aborder cet élément, nous allons étudier la balise `<link>`. Elle charge également des pages, et même si les fichiers chargés avec l'élément `link` ne peuvent pas être visualisés, cette fonctionnalité de chargement de données est importante et mérite d'être comprise pour l'utiliser au mieux.

Les attributs employés à la fois par les balises `<a>` et `<link>` partagent des caractéristiques avec tous les autres éléments HTML5, si bien qu'ils peuvent être traités comme n'importe quel autre attribut HTML5. Cependant, les attributs utilisés

par les éléments de lien ont tendance à se focaliser sur le chargement des fichiers (`.html`, `.css` et `.js`) plutôt que sur l'apparence.

L'élément `link` lui-même fait partie du contenu des métadonnées et se trouve à l'intérieur du conteneur head, au sommet de la page Web. Au chapitre 3, vous avez vu comment utiliser `link` pour charger des fichiers CSS externes. Dans la première section, je vais vous montrer comment définir votre page Web pour qu'elle soit capable de charger des feuilles de styles indépendantes.

7.1.1 Feuilles de styles alternatives

Afin de rendre accessibles les pages Web au plus grand nombre, les nouveaux navigateurs HTML5 disposent de menus déroulants qui leur permettent de sélectionner plus d'une feuille de styles. Avec la balise `<link>` et l'attribut `rel` défini à `alternate stylesheet`, on peut inclure autant de feuilles de styles que l'on souhaite et l'utilisateur peut sélectionner celle qu'il préfère. Voici un aperçu de la syntaxe générale :

```
<link rel="stylesheet" type="text/css" href="defaut.css" title="Défaut">
<link rel="alternate stylesheet" type="text/css" href="autre.css"
title="Alternative">
```

La valeur `ref alternate stylesheet` est une entité qui est différente de la valeur `alternate` que je traiterai dans la section suivante. Vous pouvez charger autant de feuilles de styles que vous voulez, mais l'utilisateur ne peut sélectionner qu'une feuille de styles alternative, et pas une feuille de styles normale.

Pour voir comment les feuilles de styles alternatives fonctionnent, l'exemple suivant commence par deux feuilles différentes qui se nomment `chaud.css` et `froid.css`. Ensuite, le code de la page Web crée le code qui charge la feuille de style par défaut (`chaud.css`), et l'utilisateur peut ensuite permuter entre les deux feuilles de styles.

Thème de couleurs chaudes

```
@charset "UTF-8";
/* CSS Document */
body {
/*FFE0A3,7F7D78,FFFAF0,7F7052,CCC8C0 */
font-family:Verdana, Geneva, sans-serif;
font-size:11;
background-color:#FFFAF0;
color:#7F7052;
}
h1 {
font-family:"Arial Black", Gadget, sans-serif;
color:#CCC8C0;
text-align:center;
}
h2 {
font-family:"Lucida Sans Unicode", "Lucida Grande", sans-serif;
background-color:#7F7D78;
color:#FFE0A3;
}
```

Thème de couleurs froides

```
@charset "UTF-8";
/* CSS Document */
body {
/*056CF2,0F88F2,52B5F2,85D3F2,F2F2F2 */
font-family:Verdana, Geneva, sans-serif;
font-size:11;
background-color:#F2F2F2;
color:#056CF2;
}
h1 {
font-family:"Arial Black", Gadget, sans-serif;
color:#52B5F2;
text-align:center;
}
h2 {
font-family:"Lucida Sans Unicode", "Lucida Grande", sans-serif;
background-color:#85D3F2;
color:#0F88F2;
}
```

Les deux thèmes utilisent un code CSS3 identique, mis à part les valeurs des couleurs. La page Web suivante (`FeuillesDeStylesAlternatives.html` disponible dans les fichiers du chapitre) utilise deux fichiers CSS externes avec un fichier par défaut (`stylesheet`) et un fichier alternatif (`alternate stylesheet`) :

```
<!DOCTYPE HTML>
<html>
<head>
<link rel="stylesheet" type="text/css" href="chaud.css" title="Affichage chaud
(par défaut)">
<link rel="alternate stylesheet" type="text/css" href="froid.css"
title="Paysage froid">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Feuilles de styles externes alternatives</title>
</head>
<body>
<h1>Chaud et froid</h1>
<h2> Passez du chaud au froid </h2>
Pour permuter, sélectionner Afficher (ou Affichage) > Style (ou  Style de la
page) à partir du menu du navigateur et choisissez le style que vous désirez.
Utilisez pour commencer Opera ou FireFox, et testez ensuite d'autres
navigateurs HTML5.
</body>
</html>
```

Le reste des manipulations à accomplir s'effectue dans le navigateur. La figure 7.1 illustre le paramétrage par défaut dans le navigateur Opera avec le menu de sélection des feuilles de styles.

Comme vous pouvez le voir à la figure 7.1, le menu Afficher > Style indique le nom de la feuille de styles CSS3 par défaut — Affichage chaud (par défaut). Si l'on veut basculer sur la feuille de styles alternative, il suffit de sélectionner Paysage froid.

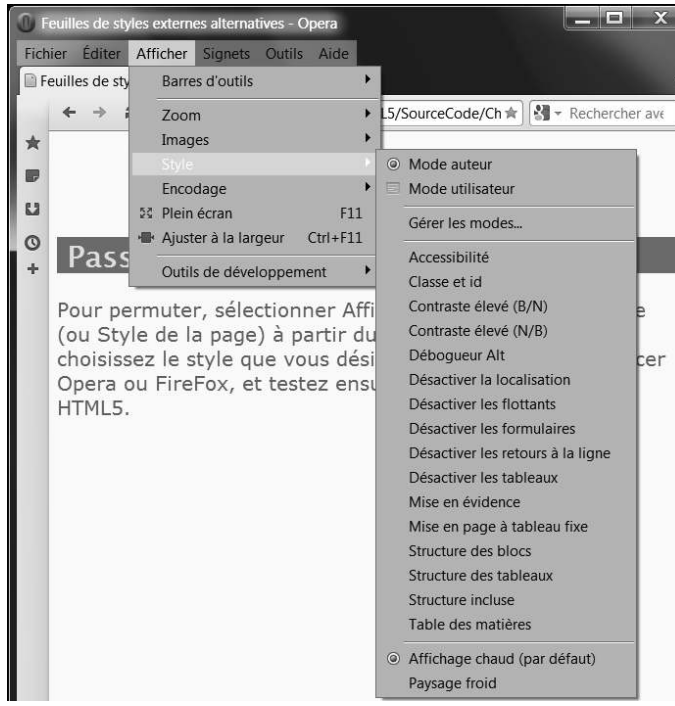


Figure 7.1 — Affichage du thème de couleurs chaudes dans le navigateur Opera.

La figure 7.2 illustre la sélection de la feuille de styles alternative dans le navigateur Firefox.

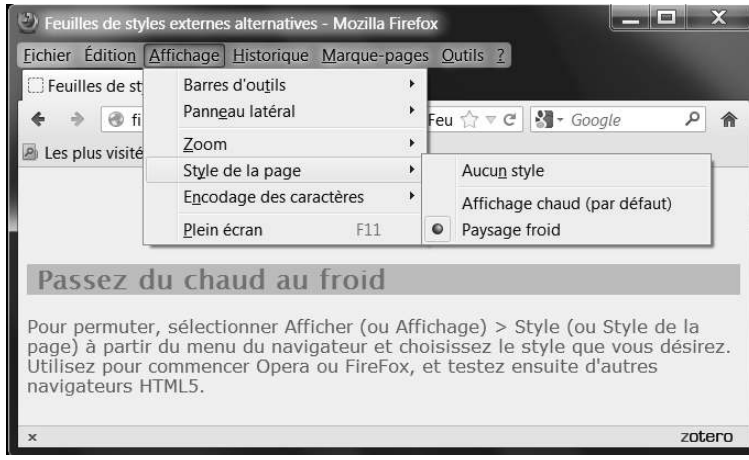


Figure 7.2 — Permutation de feuilles de styles dans Firefox.

Dans Firefox, le menu de sélection des feuilles de styles est légèrement différent, mais comme dans Opera, il offre aux utilisateurs la possibilité de changer les styles de manière dynamique s'ils le souhaitent.

7.1.2 Icônes de liens

L'attribut `rel` qui sert à assigner des feuilles de styles peut aussi être utilisé pour définir une petite icône afin de représenter la page. Les icônes peuvent être assignées à l'attribut `rel` avec la syntaxe suivante :

```
<link rel="icon" href="graphic.png" sizes="32x32"/>
```

Dans les versions précédentes de HTML, la valeur de l'attribut était `shortcut icon`, mais `icon` fonctionne également.

L'exemple suivant (`IconeLien.html` disponible dans les fichiers du chapitre) inclut également plusieurs balises `<meta>`. Ces balises contiennent des informations sur la page qui seront utilisées par les moteurs de recherche, et bien que cela soit toujours utile, ceci n'est pas nécessaire pour définir le lien vers une icône.

```
<!DOCTYPE HTML>
<html>
<head>
<meta name="application-name" content="HTML5, CSS3"/>
<meta name="description" content="HTML5 Icône lien"/>
<meta name="application-url" content="IconeLien.html"/>
<link rel="icon" href="Ancre.png" sizes="32x32"/>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Icône de page</title>
</head>
<body>
Icône de lien
</body>
</html>
```

Lors des tests d'affichage de l'icône dans cinq navigateurs différents (Safari, Chrome, Opera, Firefox et Internet Explorer), seuls Opera et Firefox étaient capables d'afficher l'icône. Vous noterez également que les navigateurs des terminaux mobiles n'affichent pas non plus l'icône. La figure 7.3 indique l'emplacement où les icônes apparaissent (une petite ancre verte) dans les navigateurs Opera et Firefox.

Pour la création de l'icône, j'ai utilisé un fichier .png défini avec la taille par défaut de 32 x 32 pixels. Vous pouvez utiliser d'autres tailles, mais les limites ne sont pas claires (il faut cependant que la hauteur et la largeur de l'image soient identiques).

7.1.3 Préchargement

HTML5 a introduit une nouvelle valeur pour l'attribut `rel` de l'élément `link` : `prefetch`. Supposons que vous ayez une page qui soit un peu lourde car elle contient

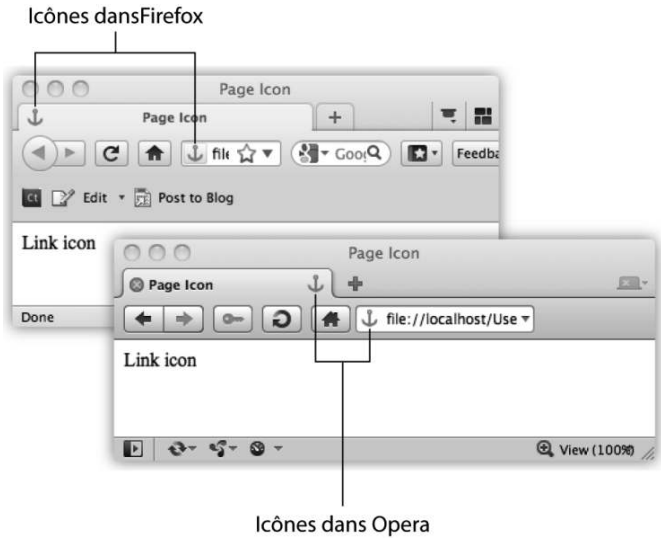


Figure 7.3 — Affichage des icônes dans Firefox et Opera.

de grandes images, des contenus vidéo ou audio. Avant que les utilisateurs n'aillent sur cette page, ne serait-il pas intéressant de précharger la page (les images et tout le reste) de telle sorte que quand ils cliquent sur le lien, tout le contenu soit déjà prêt ? C'est la raison d'être du préchargement (que l'on appelle aussi *prefetching*). Quand le navigateur est au repos, le préchargement lui donne quelque chose à faire. Par exemple, le code suivant utilise le prefetching pour charger une vidéo :

```
<link rel="prefetch" href="Test.mov">
```

Ainsi, quand l'utilisateur va sur la page qui contient la vidéo, elle a déjà commencé à se charger ou elle peut même s'être chargée complètement et être prête à être lue. Voici quelques autres exemples :

```
<link rel="prefetch" href="monkeys.html">
<link rel="prefetch" href="monsterTrucksFull.png">
<link rel="prefetch" href="http://www.sandlight.com">
<link rel="prefetch alternate stylesheet" ="http://wherever.org/fall.css">
href
<link rel="prefetch" href="Vacances.mp4" title="Vacances d'été">
```

Avant de commencer à envisager d'utiliser la valeur `prefetch` sur chaque page qui est liée à une page « lourde », vous devez réfléchir à la probabilité qu'ont les utilisateurs de visiter cette page en question. Par exemple, supposons que vous conceviez un site Web pour un grand magasin et que les utilisateurs puissent sélectionner plusieurs images de produits. Si la page Web précharge toutes les images, cela va surcharger inutilement l'ordinateur de l'utilisateur. Ainsi, au lieu d'obtenir une réponse rapide,

on risque au contraire de ralentir le processus à cause de toutes les images préchargées qui sont en mémoire.

Pour optimiser le préchargement, vous devez organiser vos pages de telle sorte que les liens vers une page lourde aient un chemin qui limite le préchargement. Les pages qui incluent des médias qui nécessitent un temps conséquent de chargement ne doivent être accessibles que par un nombre limité de chemins.

7.1.4 Autres attributs link

Parmi les autres attributs `link`, on peut lister :

- `href` : pointe sur des feuilles de styles et des icônes externes.
- `media` : spécifie le type de média pour `link` (screen, PDF, print) ; si aucune valeur n'est assignée, la valeur par défaut est « all ».
- `hreflang` : fournit la langue d'une ressource (à titre purement informatif).
- `type` : identifie le type du contenu du fichier, comme « text/css » ou les types MIME.
- `sizes` : spécifie les dimensions d'une icône, comme 32x32, 48x48, et d'autres tailles utilisées pour les images employées en tant qu'icônes.
- `title` : a une réelle valeur quand on utilise des feuilles de styles alternatives, sinon cet attribut est purement informatif.

Comme vous l'avez constaté dans les exemples utilisant l'attribut `rel`, ces autres attributs sont souvent employés conjointement avec `rel`.

7.2 LIENS DE PAGES

L'élément `a` en HTML5, ainsi que dans les versions précédentes de HTML, est l'un des éléments clés du langage. Son utilisation principale est de charger une page à l'aide de l'attribut `href`. Sans l'attribut `href`, la balise `<a>` peut servir de placeholder, mais dans la pratique, l'élément `a` est véritablement une combinaison de l'élément et de l'attribut. C'est pourquoi on a tendance à se le représenter sous la forme `a href` ou `<a href>` plutôt que sous la forme de l'élément individuel `a`. Cette section étudie les nuances de l'élément `a` en se concentrant sur l'attribut `href`, mais on va commencer par l'utilisation de l'attribut `rel` avec l'élément `a`.

7.2.1 Attribut rel en détail

L'attribut `rel` n'est pas lié qu'à l'élément `link`, et alors que la plupart des valeurs `rel` assignées à `link` s'appliquent aussi aux éléments `a` et `area`, seul un sous-ensemble sera étudié ici. Vous trouverez ci-dessous la liste complète des valeurs applicables à l'attribut `rel` de l'élément `a` :

- `alternate`

- archives
- author
- bookmark
- external
- first
- help
- index
- last
- license
- next
- nofollow
- noreferrer
- prev
- search
- sidebar
- tag
- up

Parmi celles-ci, un certain nombre sert à l'organisation de la navigation et nous les aborderons en détail au chapitre 8. Par exemple, `index`, `first`, `last`, `prev` et `next` (entre autres) font tous référence à l'ordre de navigation. J'en parle ici de telle sorte que lorsque nous étudierons la navigation dans son ensemble au chapitre 8, vous serez déjà familier de ces concepts. D'autres valeurs assignées à l'attribut `rel` de l'élément `a` concernent l'identification de certaines caractéristiques, comme l'auteur du lien ou un lien d'aide, sujet par lequel nous allons commencer.

Relations avec l'auteur

Parfois, une page Web comprend l'auteur de la page de manière à ce que l'on puisse le contacter. Pour faciliter l'identification, une valeur `author` peut être assignée au lien. Dans de telles situations, il est courant d'utiliser un lien `mailto:` dans une assignation `href`. Par exemple, le listing suivant (`LienAuteur.html` disponible dans les fichiers du chapitre) utilise la valeur `author` avec le lien `mailto:`.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
/* FFF8E3,CCCC9F,33332D,9FB4CC,DB4105  */
body {
font-family:Verdana, Geneva, sans-serif;
font-size:11px;
background-color:#CCCC9F;
color:#33332D;
}
h1 {
```

```
background-color:#33332D;
color:#9FB4CC;
font-family:"Arial Black", Gadget, sans-serif;
text-align:center;
}
h2 {
background-color:#DB4105;
color:#FFF8E3;
}
a {
text-decoration:none;
font-size:9px;
color:#DB4105;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Auteur</title>
</head>
<body>
<header>
  <h1>Tout sur HTML5</h1>
</header>
<article>
  <header>
    <h2>  En cela repose la sagesse des anciens </h2>
  </header>
  <section> Whoaaa !&#8213;<em>La sagesse des anciens ?</em>&#8213;C'est une
lourde responsabilité ! Pourquoi pas&#8213;<em>Le mieux que je puisse faire
depuis 2010 ?</em>
    <p> Au fait, qui a écrit cette chose ?</p>
    <h3>Il l'a fait !&#8595;</h3>
    <footer>
      <nav><a href="mailto:bill@billzplace.net" rel="author">Bill
Sanders</a></nav>
    </footer>
  </section>
</article>
</body>
</html>
```

En créant le lien de l'adresse électronique de l'auteur, l'élément `a` est stylé et il se débarrasse du soulignement qui est le style par défaut des liens. Cette couleur discrète se remarque bien et, dans une certaine mesure, toute la page est conçue pour attirer l'attention de l'utilisateur sur le lien. Comme vous pouvez le constater à la figure 7.4, quand le curseur de la souris survole le lien, l'adresse électronique de l'auteur apparaît dans un encadré.

L'élément `cite` peut être confondu avec la valeur `author` assignée à l'attribut `rel` d'un élément `a`. Tout d'abord, `cite` est un élément indépendant ; de plus, il met en italique le contenu d'un conteneur `cite`. Par exemple, l'extrait suivant illustre la manière dont les deux mots-clés sont utilisés dans le même paragraphe :

```
<p>La plupart des citations peuvent se trouver dans l'œuvre de <a
href="http://www.willieS.com" rel="author">William Shakespeare</a>, notamment
```

```
le célèbre livre de référence, <cite>Camford's Complete Works of the
Bard</cite>.</p>
```

Figure 7.4 — Utilisation d'un lien de l'adresse électronique de l'auteur.

Ce code génère le texte suivant :

La plupart des citations peuvent se trouver dans l'œuvre de William Shakespeare, notamment le célèbre livre de référence, *Camford's Complete Works of the Bard*.

Comme vous pouvez le constater, quand on place ce code dans une page Web, la valeur `author` est informative et l'élément `cite` modifie l'apparence du texte. À certains égards, les deux sont informatifs en ce sens où chacun attire l'attention sur le contenu, l'un dans le code et l'autre à l'écran.

Types de liens hiérarchiques et séquentiels

Vous pouvez organiser vos liens en utilisant à la fois des types de liens hiérarchiques et séquentiels. Les valeurs `rel` hiérarchiques comprennent `index` et `up`. La valeur `up` fait référence au niveau supérieur dans la hiérarchie et `index` fait référence au sommet de la hiérarchie. Par exemple, le code suivant fait référence à un répertoire qui est sommet de la hiérarchie, trois niveaux au-dessus de la page appelante :

```
<a href="/" rel="index up up up">Accueil</a>
```

Le chemin de cet exemple est exprimé en référençant à la fois le niveau d'`index` et le nombre de niveaux `up`.

Les types de liens séquentiels comprennent `first`, `last`, `next` et `prev`, chaque mot-clé étant relatif à une page au sein d'une séquence. Par exemple, le code suivant va à la page suivante par rapport à la page de la séquence :

```
<a href="page4.html" rel="next">Page 4</a>
```

L'implémentation de ces types de liens est différente selon les navigateurs et il est préférable de les utiliser avec l'élément `link` afin de planifier l'organisation d'un site par rapport à une page donnée plutôt que de relier directement une page avec un élément `a`.

7.2.2 Ancres de page et ID

Outre le lien direct sur une page, vous pouvez faire un lien sur un emplacement spécifique d'une page. Pour ce faire, on peut assigner une ancre à une balise de la page en utilisant l'attribut `name`. Par exemple, le code suivant sautera à l'emplacement de la page en cours où se trouve le nom « developpeur » :

```
<a href="#developpeur">Développeurs</a>
```

Pour définir la cible avec une ancre, il suffit d'assigner à une balise le nom de l'ancre de la manière suivante :

```
<div name="developpeur">
```

Quand on teste la technique de l'ancre sur les navigateurs HTML5, on s'aperçoit qu'il y a des erreurs. Les navigateurs HTML5 semblent avoir adopté cette technique en utilisant exclusivement CSS3 pour créer des ID. L'exemple suivant (`IDAncre.html` disponible dans les fichiers du chapitre) montre comment utiliser les ID en tant qu'ancres :

```
<!DOCTYPE HTML>
<html>
<style type="text/css">
/*D4CBA0,BD4A14,804130,4F3C33,6D7F59*/
body {
font-family:Verdana, Geneva, sans-serif;
background-color:#D4CBA0;
color:#804130;
}
h1 {
font-family:"Arial Black", Gadget, sans-serif;
color:#4F3C33;
background-color:#BD4A14;
text-align:center;
}
h2 {
color:#6D7F59;
}
h3 {
margin-left:15px;
color:#4F3C33;
}
a {
font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;
font-size:11px;
```

```
color:#BD4A14;
text-decoration:none;
}
nav {
text-align:center;
}
#fsquirell { };
#cats { };
#dogs { };
</style>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Ancres</title>
</head>
<body>
<article>
<header>
  <nav><a href="#fsquirrel">Écureuils volants</a>  | <a
href="#chats">Chats</a>  | <a href="#chiens">Chiens</a></nav>
  <h1>Soins animaliers</h1>
  Si vous n'êtes pas intéressé par les écureuils volants, vous pouvez
sélectionner les ancres "Chat" ou "Chien" pour aller directement à votre sujet
de prédilection. </header>
<section ID="fsquirrel">
  <header>
    <h2>Bien-être des écureuils volants</h2>
  </header>
  <h3>Hangars</h3>
  <h3>Pistes d'atterrissage</h3>
  <h3>Formation au vol</h3>
  <h3>Nourriture</h3>
  <h3>Bagages (ces écureuils en ont beaucoup...)</h3>
</section>
<section ID="chats">
  <header>
    <h2>Bien-être des chats</h2>
  </header>
  <h3>Panières</h3>
  <h3>Grattoirs</h3>
  <h3>Litières</h3>
  <h3>Nourriture</h3>
  <h3>Souris en peluche</h3>
</section>
<section ID="chiens">
  <header>
    <h2>Bien-être des chiens</h2>
  </header>
  <h3>Niches</h3>
  <h3>Promenades</h3>
  <h3>Dressage</h3>
  <h3>Nourriture</h3>
  <h3>Balles en mousse</h3>
</section>
<footer>
  <nav><a href="#fsquirrel">Écureuils volants</a>  | <a
href="#chats">Chats</a>  | <a href="#chiens">Chiens</a></nav>
```

```
</footer>
</body>
</html>
```

Quand on utilise des ID CSS3 à la place d'ancres sur des terminaux mobiles, on trouve que la conception n'est pas trop contraignante eu égard à la petite taille de l'écran. Comme vous pouvez le voir à la figure 7.5, les ancres facilitent la navigation sur la page d'un terminal mobile.

À la figure 7.5, l'écran de gauche illustre la page initiale sur un navigateur Opera Mini. Quand on clique sur le lien Chiens, la page se positionne directement sur l'information relative aux chiens. Vous remarquerez que le menu se situe à la fois au sommet et au bas de la page. En général, si votre page est suffisamment longue pour nécessiter des ID pour se déplacer, il est préférable d'avoir un menu en haut et en bas de la page. Si la page est très longue, vous pouvez fournir un ID à l'élément `nav` puis lier chaque section au menu.

Si vous voulez définir directement un lien vers un ID ou une ancre, vous pouvez simplement ajouter `#nom_ID` à l'URL. Par exemple, si vous voulez créer sur votre site (ou même sur un autre site) un lien direct sur les informations relatives aux chats, il suffit de créer le lien suivant :

```
<a href="http://my.domain.com/IDAncre.html#chats">
```

Figure 7.5 — Utilisation d'ID à la place d'ancres.

Si l'on est dans le même répertoire, il suffit d'écrire :

```
<a href="IDAncre.html#chats">
```

Au chapitre 8, vous verrez comment utiliser les ID et les ancres lors de la planification d'une stratégie de navigation.

7.2.3 Target

Jusqu'à présent, tous les liens remplaçaient la page appelante par une nouvelle page qui était chargée dans la fenêtre du navigateur. Vous pouvez cependant utiliser l'attribut `target` avec la balise `<a>` pour modifier la façon dont une page apparaît (on appelle cela le contexte de navigation). Il existe plusieurs contextes de navigation avec l'attribut `target` :

- `_self` remplace la page en cours ; c'est le comportement par défaut si aucun contexte de navigation n'est précisé.
- `_blank` ouvre la nouvelle page dans une nouvelle fenêtre de navigateur, ce qui crée un nouveau contexte de navigation.
- `_parent` ouvre une nouvelle page dans le document « parent » de la page en cours. Le document parent est en général la fenêtre du navigateur qui a provoqué l'ouverture de la page en cours.
- `_top` ouvre la nouvelle page dans la totalité de la fenêtre du navigateur.

Ces contextes de navigation sont assignés de la manière suivante :

```
<a href="somePage.html" target="_blank">
```

La barre de soulignement (underscore) dans le nom de tous les contextes de navigation est obligatoire, ce qui signifie que le code suivant ne fonctionnera pas :

```
target =''blank''
```

Dans les versions précédentes de HTML, les éléments `frame` et `frameset`, qui étaient abondamment utilisés, pouvaient tous les deux être utilisés comme valeurs de l'attribut `target`. De la même manière, les contextes de navigation _parent et _top étaient utilisés pour ouvrir une page dans un cadre différent. En HTML5, la principale utilisation de l'attribut `target` consiste à sélectionner le contexte _blank par rapport au comportement par défaut `_self`.

Nouveaux contextes de navigation dans les navigateurs pour ordinateur personnel

Quand on utilise l'attribut `target` de l'élément `a` pour créer un contexte de navigation `_blank` sur un ordinateur, la page actuelle reste à l'écran et la page demandée apparaît dans une nouvelle fenêtre de navigateur ou un nouvel onglet. Le programme suivant (`LienVersCible.html` disponible dans les fichiers du chapitre) est une simple illustration de la manière dont cela fonctionne.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
```

```
h1 {
font-family:"Arial Black", Gadget, sans-serif;
color:#060;
}
a {
color:#900;
}
h3 {
font-family:Verdana, Geneva, sans-serif;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Ouvre une nouvelle page</title>
</head>
<body>
<header>
  <h1>Page originale</h1>
</header>
<nav>
  <h3><a href="http://www.w3.org" target="_blank">World Wide Web</a></h3>
</nav>
</body>
</html>
```

En fonction des paramètres du navigateur, la nouvelle page peut apparaître dans un nouvel onglet au lieu d'une fenêtre séparée. Vous pouvez déplacer l'onglet pour créer une fenêtre séparée de telle sorte que les deux pages puissent être visualisées de manière simultanée.

Nouveaux contextes de navigation dans les navigateurs pour terminal mobile

Quand une page Web utilise un contexte de navigation `_blank` sur un périphérique mobile, on n'a pas la possibilité de visualiser plusieurs pages à la fois dans une seule fenêtre. Au lieu de cela, la page appelante est traitée comme une page précédente (dans Opera Mini) à laquelle on peut accéder en appuyant sur la flèche de retour ou par toute autre méthode. Le navigateur Safari de l'iPhone possède une icône dans le coin inférieur droit qui indique le nombre de pages actuellement chargées. Quand l'utilisateur tape sur cette icône, il peut visualiser jusqu'à huit pages dans une fenêtre où il a la possibilité de les faire glisser pour les afficher séquentiellement. La figure 7.6 illustre ce fonctionnement sur le navigateur Safari de l'iPhone.

Si la page est ouverte en utilisant un contexte de navigation `_blank` dans Safari mobile, elle ne possède pas de lien de retour comme dans le navigateur Opera Mini ; cependant, elle est ouverte dans une nouvelle fenêtre de navigateur en même temps que la page appelante.

7.3 UTILISATION DES IFRAMES

L'élément `iframe` déclare un cadre inséré (iframe est l'abréviation de *inline frame* que l'on traduit en général par cadre inséré). En utilisant des cadres insérés, vous pouvez

Figure 7.6 — Affichage de plusieurs pages dans Safari sur un iPhone.

charger d'autres pages Web ou d'autres médias au sein d'une seule page Web. L'élément représente ce que l'on appelle un contexte de navigation imbriqué. La section « Target » de ce chapitre a traité des différents contextes de navigation en termes de fenêtres et d'onglets différents, mais un *contexte de navigation imbriqué* se produit quand une page est imbriquée à l'intérieur d'une autre page. Fondamentalement, une balise `<iframe>` place une page Web à l'intérieur d'une autre page Web.

On peut se poser la question de l'intérêt d'une telle fonctionnalité alors qu'il suffit d'ouvrir une nouvelle fenêtre ou un autre onglet. Cela permet en fait aux utilisateurs d'avoir une idée de ce qu'il y a dans ces pages pour qu'ils puissent ensuite s'y rendre s'ils trouvent leur contenu pertinent.

On peut aussi utiliser les iframes pour placer des vignettes graphiques sur une page, et ainsi permettre de sélectionner différentes vignettes afin d'afficher l'image dans sa taille réelle. Vous pouvez ainsi créer une seule page Web où l'utilisateur peut visualiser plusieurs images différentes et sélectionner les liens graphiques qui affichent l'image sur la même page, sans qu'il soit nécessaire d'utiliser JavaScript ou Ajax.

7.3.1 Imbrication de pages Web

L'élément HTML5 `iframe` possède plusieurs attributs, dont certains sont nouveaux en HTML5, mais pour commencer, vous avez uniquement besoin de connaître la balise

de base et la manière dont elle fonctionne. Le code suivant constitue le squelette d'une balise `<iframe>` avec une page Web imbriquée :

```
<iframe src="http://www.w3.org"></iframe>
```

Cette balise place simplement la page Web source dans le coin supérieur gauche de la page appelante. Afin de mieux comprendre les options de la balise iframe, le programme suivant (`iframe.html` disponible dans les fichiers du chapitre) embarque deux pages Web différentes à l'intérieur d'une page Web et ajoute plusieurs attributs que vous pouvez voir.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
/*657BA6,F2EDA2,F2EFBD,F2DCC2,D99379*/
body {
background-color:#F2EDA2;
}
h1 {
font-family:Verdana, Geneva, sans-serif;
color:#657BA6;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Iframe</title>
</head>
<body>
<!DOCTYPE HTML>
<html>
<body>
<article>
  <header>
    <h1>Avant iframes</h1>
  </header>
  <section>
    <iframe name="info" width="480", height="320" sandbox="allow-same-origin"
seamless src="http://www.smashingmagazine.com"></iframe>
    <iframe name="info2" width="480", height="320" sandbox seamless
src="http://www.w3.org"></iframe>
  </section>
  <footer>
    <h1>Après iframes</h1>
  </footer>
</article>
</body>
</html>
</body>
</html>
```

Dans les deux balises `<iframe>`, vous pouvez voir plusieurs attributs, dont certains ont déjà été utilisés avec d'autres éléments. L'élément `iframe` lui-même possède les sept attributs suivants (plus les attributs HTML5 globaux) :

- src
- srcdoc
- name
- sandbox
- seamless
- width
- height

Les attributs srcdoc, sandbox et seamless sont nouveaux. Lors de la rédaction de cet ouvrage, l'attribut srcdoc n'avait pas encore été implémenté dans un seul des navigateurs testés, mais quand il le sera, il permettra l'affichage d'un fichier source texte/HTML avec des informations spécifiques pour l'iframe. L'attribut sandbox, qui est implémenté dans le navigateur Google Chrome, est utilisé pour restreindre, à des fins de sécurité, les types de contenu et les fonctionnalités qui sont disponibles dans un iframe. L'attribut seamless n'a pas encore été implémenté non plus, mais quand il le sera, tous les liens seront ouverts dans le contexte de navigation parent et non pas dans le contexte de navigation imbriqué (à l'intérieur de l'iframe). Les anciens navigateurs et les navigateurs HTML5 qui ne les ont pas encore implémentés ignorent tous ces nouveaux attributs iframe. Par conséquent, vous pouvez ajouter les attributs aux balises <iframe> pour prendre de bonnes habitudes de telle sorte que lorsqu'ils seront disponibles, ils puissent améliorer la sécurité de vos pages Web. La figure 7.7 illustre la manière dont les pages imbriquées apparaissent sur l'écran d'un ordinateur.

Les titres h1 avant et après les pages imbriquées ne sont pas soumis au style CSS3 de la page parente. Vous pouvez aussi voir que chaque page est à l'intérieur d'une page (avant et après l'insertion des deux autres pages Web).

Si vous examinez le code, vous verrez que les dimensions (320 x 480) suggèrent que la résolution d'affichage a été optimisée pour un périphérique mobile. Quand on teste le code sur un terminal mobile, l'iframe s'ouvre cependant pour afficher la totalité des pages imbriquées. Comme il n'y a pas de barres de défilement sur les terminaux mobiles, la seule alternative pour afficher la totalité du contenu des pages imbriquées consiste donc à leur permettre de défiler horizontalement et verticalement à l'intérieur de l'iframe. De prime abord, cela peut sembler contraire à ce qui est prévu pour l'utilisation des iframes sur les terminaux mobiles, mais vous verrez au chapitre 8, comment les iframes peuvent être employés comme des sites Web à page unique optimisés pour les navigateurs mobiles.

7.4 À VOUS DE JOUER !

La conception d'un site Web peut se révéler très amusante et l'un des défis à relever consiste à ce que tous les liens fonctionnement parfaitement de concert. Dans le prochain chapitre, vous allez étudier les stratégies de navigation, mais pour le moment, vous devez vous entraîner à préparer toute une série de liens et d'icônes. Voici votre travail :

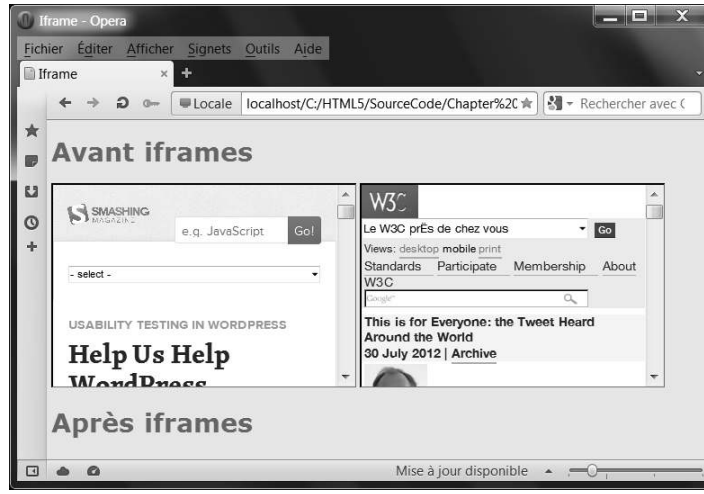


Figure 7.7 — Imbrication de pages Web au sein d'une même page Web.

1. Créez trois pages Web dans lesquelles vous inclurez plusieurs sections avec des titres et des sous-titres, de telle sorte que l'utilisateur devra faire défiler l'écran pour voir les sections du bas.
2. Sur chacune des pages Web, définissez un lien vers une icône (voir « Icônes de liens » dans ce chapitre). C'est à vous de choisir si vous voulez que chaque page ait une icône différente ou bien qu'il y ait une icône de site qui soit identique pour toutes les pages.
3. Créez deux feuilles de styles différentes CSS3 (externes) et proposez un accès à ces feuilles de styles alternatives sur tous les pages (voir « Feuilles de styles alternatives » dans ce chapitre).
4. Créez une troisième feuille de styles qui ne contiendra que des ID qui seront utilisés à la place d'ancres. Placez un ID dans chaque section de vos pages.
5. Pour finir, créez des liens sur chacune des trois pages qui renverront aux deux autres pages et sur tous les ID de chaque page.
6. Prenez le temps de réaliser cet exercice. Vous pouvez créer des pages sur n'importe quel sujet et il n'y a pas de raison de se prendre au sérieux (à moins que vous ne pensiez à une application pour un client !). Vous n'avez donc pas à vous préoccuper du contenu et vous avez carte blanche pour exprimer votre créativité.

8

Stratégies de navigation

Objectif

On désigne habituellement le parcours d'un site Web sous le terme de *navigation*, et HTML5 accrédite cela en proposant une balise `<nav>`. La navigation reste simple quand les sites le sont, mais n'importe quel site Web peut virtuellement être exposé à une navigation inadaptée. De la même manière, une bonne navigation peut rendre convivial le site le plus complexe et permettre à l'utilisateur de trouver tout ce qu'il recherche.

Dans la mesure où ce livre est consacré à HTML5, ce chapitre va vous montrer comment définir différents systèmes de navigation avec les éléments spécifiques de HTML5. Mais avant de rentrer dans les détails, vous devez comprendre quelques concepts généraux de la navigation Web.

8.1 CONCEPTS DE NAVIGATION WEB

Quand ils réfléchissent à la navigation, les concepteurs Web prennent en considération les éléments suivants :

- **Conception de l'interface** : Jennifer Tidwell décrit parfaitement la conception d'interfaces pour le Web dans son ouvrage *Designing Interfaces*. La plupart des processus et des modèles que traite Tidwell dans son livre sont étudiés dans ce chapitre, mais je n'ai pas la prétention d'être aussi profond et exhaustif sur ce sujet ; c'est la raison pour laquelle si vous souhaitez obtenir plus d'information sur la conception d'interfaces, vous ne devez pas hésiter à consulter son livre.

- **Conception de l'information** : Edward Tufte, grand spécialiste de la conception de l'information, a montré comment différents types d'informations peuvent être présentés de telle sorte qu'ils soient plus facilement assimilables. En ce qui concerne la conception de la navigation Web, il a introduit l'idée que l'information est l'interface. En d'autres termes, la navigation est de l'information disposée de telle sorte que les utilisateurs puissent trouver ce qu'ils cherchent.

Les concepts de Tidwell ou de Tufte ne peuvent pas être résumés dans une jolie définition. On peut parler d'interaction comme d'une réponse à une autre action, comme quand deux personnes sont en train de discuter. Il s'agit d'interaction sociale et c'est quelque chose que nous faisons tout le temps (y compris l'interaction par le biais d'un ordinateur, comme dans le cas du chat). Le même concept s'applique au traitement d'une page Web qui peut être considérée comme une personne. L'utilisateur fait quelque chose et la page Web répond à partir d'un ensemble fini de choix que le concepteur a créé. Plus le concepteur soigne son travail, plus l'utilisateur se sentira à l'aise. L'objectif de toute conception d'interaction qui se respecte est de tenter de créer un environnement d'interaction confortable.

8.1.1 Navigation de concepteur et navigation d'utilisateur

La conception de navigation comporte un nombre presque infini de possibilités, et il est souhaitable de déterminer un parcours navigationnel qui permette aux utilisateurs de se déplacer facilement. La première chose que vous devez vous demander est de savoir quel sera l'utilisateur typique de votre site. Ensuite, vous devez vous persuader que vous ne représentez pas l'utilisateur lambda de votre site. Jennifer Tidwell en fait d'ailleurs sa maxime en matière de conception d'interface : « Connaissez vos utilisateurs car ils ne vous ressemblent pas ! ». On peut ajouter deux corollaires à cette maxime :

- Meilleur est le concepteur, plus il y a de risques que l'interface soit mauvaise.
- Les développeurs excellents conçoivent presque toujours de piètres interfaces.

Dans ces conditions, si vous aspirez à devenir un bon concepteur ou un bon développeur, vous allez probablement concevoir de mauvaises interfaces, à moins que vous n'y fassiez attention. Voici pourquoi : les bons concepteurs se concentrent sur l'aspect de la page, et pas sur la facilité de déplacement des utilisateurs sur l'ensemble du site. Les concepteurs veulent montrer leur créativité, ce qui est une bonne chose, mais quand cela empêche les utilisateurs de naviguer d'une page à une autre, cela constitue un problème.

L'une des pires interfaces utilisateur jamais conçues est celle du site Web du MoMA, le musée d'art moderne de New York. La navigation est basée sur une pile de cubes dépourvus de libellés. Les utilisateurs sont censés positionner leur souris sur chaque cube pour que le libellé de l'article pointé apparaisse. Pour que le site du MoMA fonctionne, il a pourtant fallu une programmation élaborée. Le code aurait sans doute ravi n'importe quel développeur, mais il a conduit à la catastrophe car, comme le concepteur, le développeur a pensé à ses talents de codeur et non pas à l'utilisateur du

site. Afficher le nom du lien quand la souris survole un cube nécessite des compétences en programmation qu'un concepteur ne possède pas forcément. Si vous voulez créer un système de navigation franchement mauvais, il ne vous reste donc plus qu'à associer le meilleur concepteur et le meilleur développeur !

Est-il possible d'être un bon concepteur et/ou un bon développeur tout en créant de bonnes interfaces ? Assurément, mais vous devez y réfléchir. Vous devez adopter le point de vue de l'utilisateur typique de votre site Web. Qui sont vos utilisateurs ? Est-ce que ce sont des enfants ou des adultes ? Votre public est-il plutôt féminin ou masculin, ou bien équitablement réparti ? Quelle est sa moyenne d'âge ? Quel est le style de l'utilisateur ? S'agit-il d'hommes d'affaire ? Si tel est le cas, quelle est leur activité et quelle place occupent-ils au sein de leur entreprise ? Est-ce qu'ils sont dirigeants ou bien employés ? Vous devez déterminer qui sont vos utilisateurs (car vous savez déjà qui vous êtes).

Si vous êtes un concepteur et que vous créez un site Web pour d'autres concepteurs Web, voulez-vous leur montrer quel bon concepteur vous êtes ou bien comment ils peuvent devenir de meilleurs concepteurs ? De la même manière, si vous êtes un développeur et que vous créez un site pour d'autres développeurs, vous voulez absolument leur montrer du code qui leur permettra d'améliorer leurs compétences en programmation. Les développeurs veulent voir du code, alors que ce n'est pas le cas des concepteurs qui sont plus intéressés par les outils et les techniques de conception (les concepteurs adorent bien entendu le code CSS3 !). Vous devez donc travailler votre projet de navigation en gardant toujours à l'esprit ce qui est souhaitable pour vos utilisateurs.

Le meilleur moyen de savoir si votre interface est correcte consiste à la tester avec des utilisateurs représentatifs. Si vous créez un site éducatif pour des écoliers, il faut le tester avec des écoliers. De la même manière, si vous vendez des produits de haute couture à des femmes aisées, il n'est pas souhaitable de faire tester votre navigation par des adolescentes. Cela vous prendra un peu plus de temps, mais vous aurez un bien meilleur site si vous testez votre site avec le type de public qui l'utilisera.

Le fait de connaître vos utilisateurs ne signifie que vous pouvez vous dispenser de soigner la conception et d'utiliser des technologies sophistiquées. Cela signifie que vous devez imaginer ce qu'ils vont penser de votre site. Vous n'allez pas changer vos utilisateurs et vous devez donc réaliser le site pour eux, et non pas pour vous. Si le site ne leur convient pas, ils n'y retourneront pas.

8.1.2 Navigation globale

La navigation globale dans les pages Web fait référence aux grandes catégories de navigation qui peuvent être placées sur chacune des pages d'un site Web. La navigation globale aide les utilisateurs à toujours savoir où ils se trouvent sur un site, ainsi, quel que soit l'endroit où ils sont allés, ils ont toujours à portée de clic une feuille de route qui leur est familière.

Si vous projetez un voyage qui vous mène de la Californie jusqu'à la côte est des États-Unis, vous allez emprunter les grandes autoroutes qui permettent de traverser les

différents états. Ces autoroutes peuvent être considérées comme les éléments globaux d'un périple de 5 000 kilomètres qui permet de relier les deux côtes. Cependant, vous serez obligé d'emprunter quelques routes secondaires pour vous connecter au grand réseau autoroutier.

De la même manière, dans la navigation globale, vous devez prendre en compte la navigation entre les liens principaux. Par exemple, supposons que vous ayez un très grand site avec une navigation globale décomposée en trois catégories :

- Animal
- Végétal
- Minéral

Il est possible d'organiser les liens de cette navigation globale en créant sur chaque page la structure suivante :

Animal | Végétal| Minéral

Mais à l'intérieur de ces catégories générales, vous allez avoir besoin de catégories plus spécifiques. Par exemple, supposons qu'un utilisateur veuille trouver une race particulière de chien : un Saint-Bernard. Un concepteur pourrait proposer le chemin suivant :

- Animal
- Mammifère
- Chien
- Race
- Saint Bernard

Chaque sous-menu ayant de nombreux choix, il nous faut examiner les éléments disponibles en HTML5 pour gérer ces chemins de navigation dans le cadre d'une navigation globale.

Utilisation des listes en navigation globale

Pour implémenter la navigation globale, il est possible d'utiliser des listes. Examinez, par exemple, le script suivant (`NavigationParListe.html` disponible dans les fichiers du chapitre) qui utilise la navigation globale et la navigation locale.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
/* 3C514C,98AB98,D3DFD3,A6A47D,8C1616 */
body {
color:#3C514C;
background-color:#D3DFD3;
font-family:Verdana, Geneva, sans-serif;
}
h3 {
color:#8C1616;
```

```
background-color:#A6A47D
}
a {
color:#8C1616;
font-size:11px;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Navigation par listes</title>
</head>
<body>
<nav> <a href="#">Animal</a> | <a href="#">Végétal</a> | <a
href="#">Minéral</a> |
  <ul>
    <h3> Animaux</h3>
    <li><a href="#">Mammifères</a></li>
    <li><a href="#">Poissons</a></li>
    <li><a href="#">Oiseaux</a></li>
    <ul>
      <h3> Mammifères</h3>
      <li><a href="#">Chiens</a></li>
      <li><a href="#">Chats</a></li>
      <li><a href="#">Autres</a></li>
      <ul>
        <h3> Chiens</h3>
        <li><a href="#">Golden Retriever</a></li>
        <li><a href="#">Setter irlandais</a></li>
        <li><a href="#">Berger allemand</a></li>
        <li><a href="#">Saint Bernard</a></li>
      </ul>
    </ul>
  </ul>
</nav>
</body>
</html>
```

En vous contentant d'étudier le code, vous allez sans doute vous rendre compte que ce type de système de navigation va rapidement submerger la page. La figure 8.1 illustre le résultat même quand le nombre de choix possibles a été considérablement réduit.

Avec un écran suffisamment grand et des choix abrégés comme ceux que nous avons utilisés dans notre exemple, on peut envisager d'avoir un système de navigation avec des balises `<ul>`, mais ce système de listes est totalement irréaliste dans le cas de terminaux mobiles. La figure 8.2 montre comment la navigation occupe la totalité de la fenêtre sur un téléphone mobile.

Comme vous pouvez le constater clairement à la figure 8.2, il est nécessaire d'envisager un autre système de navigation de telle sorte que les rubriques puissent être visualisées. En fait, ce système de navigation ressemble plus à la carte du site, sujet que j'aborderai plus tard dans ce chapitre, et il ne peut donc pas être utilisé comme navigation globale.

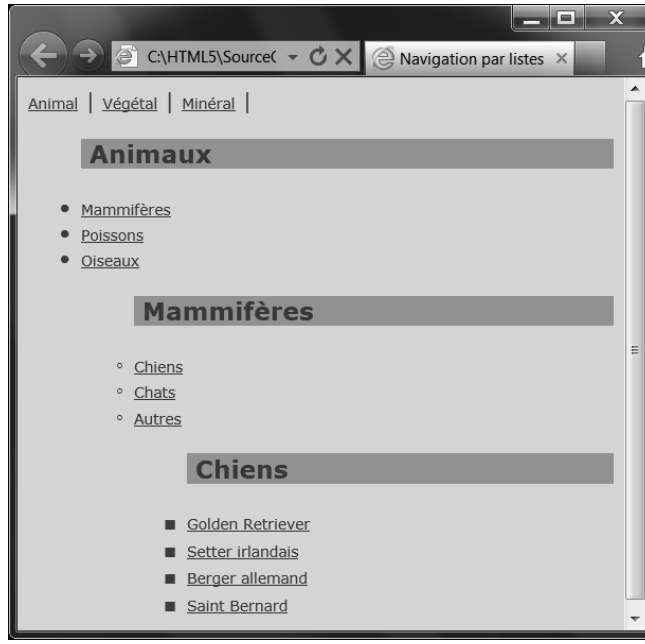


Figure 8.1 — Navigation par liste.

Figure 8.2 — La navigation par liste occupe toute la zone d'affichage sur un périphérique mobile.

8.1.3 Menus déroulants et navigation globale

Si l'on veut expérimenter une autre approche de la navigation globale à l'aide de liens textuels, il est possible d'utiliser des éléments qui offrent plus d'informations dans un espace plus petit, notamment la balise `<select>`. L'élément `select` affiche le premier élément d'une liste d'options qui peuvent être vues uniquement quand l'utilisateur clique sur la fenêtre qui apparaît. La syntaxe comprend une balise `<select>` avec une balise `<option>` imbriquée au sein du conteneur `select`. Chaque conteneur `option` contient un objet qui est visible quand s'ouvre le menu déroulant. L'extrait de code suivant indique la syntaxe de base :

```
<select id="animaux" name="global1">
    <option value="chevaux">Chevaux</option>
    <option value="chiens">Chiens</option>
    ...
</select>
```

Cela peut constituer un moyen pratique de placer tous les liens d'un site dans un petit emplacement et de s'en servir comme menu global. Vous pouvez imbriquer autant de balises `<option>` à l'intérieur du conteneur `<select>` que vous le souhaitez. Afin de voir comment cela peut s'organiser dans un système de navigation globale, le script HTML5 suivant (`NavSelect.html` disponible dans les fichiers du chapitre) illustre un exemple simple.

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Menu déroulant</title>
</head>
<nav>
  <label for="animals">Animaux </label>
  <select id="animals" name="global1">
    <option value="horses">Chevaux</option>
    <option value="dogs">Chiens</option>
    <option value="cats">Chats</option>
    <option value="rabbits">Lapins</option>
    <option value="raccons">Ratons laveurs</option>
  </select>
  <label for="vegetables">Légumes </label>
  <select id="vegetables" name="global2">
    <option value="carrots">Carottes</option>
    <option value="squash">Courges</option>
    <option value="peas">Petits pois</option>
    <option value="rice">Riz</option>
    <option value="potatoes">Pommes de terre</option>
  </select>
  <label for="minerals">Mineraux </label>
  <select id="minerals" name="global3">
    <option value="tin">Étain</option>
    <option value="gold">Or</option>
    <option value="copper">Cuivre</option>
```

```
    <option value="iron">Fer</option>
    <option value="mercury">Mercure</option>
  </select>
</nav>
<body>
</body>
</html>
```

Avec toutes ces balises HTML5, vous vous attendez sans doute à une page Web de taille importante, mais la figure 8.3 montre que cela prend très peu d'espace.

Le code HTML5 n'a pas de code CSS3 pour mettre en forme le texte, et comme vous pouvez le constater, la police par défaut du corps du texte est une police avec empattement alors que la police par défaut du menu est une police sans empattement. Quand on utilise CSS3 pour styler, on travaille avec la balise `<select>` pour attribuer le style et non pas la balise `<option>`. Si vous stylez l'élément `option`, vous pouvez styler la famille de police avec d'excellents résultats, mais les autres formes de stylages sont imprévisibles entre les différents navigateurs.

Figure 8.3 — Affichages des choix d'un menu avec la balise `<select>`.

Si les catégories apparaissent un peu superficielles, vous pouvez ajouter plus de détails dans un format hiérarchique en utilisant la balise `<optgroup>`. Avec chaque balise, on ajoute un nouveau sous-groupe. Vous pouvez imbriquer les sous-groupes sur plusieurs niveaux si vous le souhaitez. Le listing suivant (`OptionGroup.html` disponible dans les fichiers du chapitre) montre l'utilisation de l'élément `optgroup` avec les balises `<select>` et `<option>`.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
select {
background-color:#F2EFBD;
color:#657BA6;
font-family: Verdana, Geneva, sans-serif;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Menu déroulant stratifié</title>
</head>
<nav>
```

```
    <label for="animals">Animaux</label>
    <select id="animals" name="global1">
    <optgroup label="Chiens">
      <option value="hounds">Chien de meute</option>
      <option value="work">Chien de travail</option>
      <option value="terrier">Terriers</option>
      </optgroup>
    <optgroup label="Chevaux">
      <option value="race">Pur-sang</option>
      <option value="work">Cheval de trait</option>
      <option value="show">Cheval de parade</option>
      </optgroup>
      <optgroup label="Lapins">
      <option value="pets">Lapin domestique</option>
      <option value="pests">Lapin nuisible</option>
      <option value="easter">Lapin de Pâques</option>
      </optgroup>
    </select>
  </nav>
  <body>
  </body>
  </html>
```

L'affichage des titres des catégories générées par l'élément `optgroup` diffère selon les navigateurs. La figure 8.4 illustre les différents aspects du même menu sur plusieurs navigateurs.

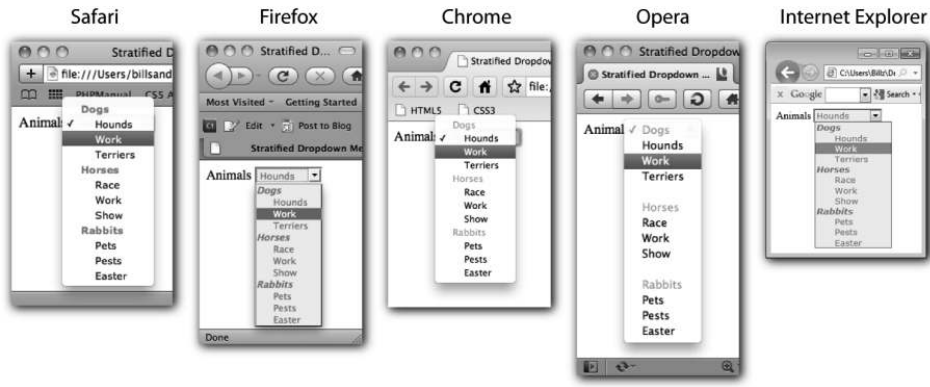


Figure 8.4 — Utilisation de la balise `<optgroup>`.

Parmi les cinq navigateurs testés, seuls Firefox et Internet Explorer mettent en italique les titres `optgroup` et conservent la combinaison des couleurs quand le menu s'ouvre. Les autres navigateurs n'affichent la bonne combinaison de couleurs que lorsque le menu est fermé.

8.2 UTILISATION DE JAVASCRIPT POUR APPELER UNE PAGE LIÉE

Tout système de navigation globale a besoin d'un moyen d'appeler différentes pages Web et les menus déroulants ont besoin d'un moyen d'appeler un élément sélectionné. Jusqu'à présent, la balise `<a>` nous avait donné satisfaction pour la gestion des liens, mais vous avez probablement remarqué que les menus déroulants n'avaient pas de liens. La balise `<select>` doit fonctionner avec l'élément `form` (qui est étudié en détail au chapitre 14) et une fonction JavaScript (le chapitre 12 fournit des informations sur l'utilisation de JavaScript). Du point de vue de HTML5, l'extrait de code suivant fournit les éléments essentiels à connaître :

```
<form name="menuNow">
    <label for="animals">Animals</label>
    <select id="animals" name="global1" onChange="optionMenu()">
      <option value="animals/horses.html">Chevaux</option>
      <option value="animals/dogs.html">Chiens</option>
```

Les noms des éléments `form` et `select` sont importants car JavaScript les utilise comme chemin vers l'`option` sélectionnée (si vous êtes familier avec les tableaux, les options sont toutes traitées comme des éléments de tableaux).

Le code JavaScript est placé dans un fichier séparé car si vous voulez l'utiliser dans un système de navigation globale, il n'est pas souhaitable de le réécrire sur chaque page. Le code JavaScript suivant doit être enregistré dans un fichier texte nommé `globMenu.js`.

```
function optionMenu()
   {
            var choice = document.menuNow.global1.selectedIndex;
            var urlNow = document.menuNow.global1.options[choice].value;
            window.location.href = urlNow;
   }
```

Cela reflète le DOM (Document Object Model) HTML5. Le `document` est la page Web, `menuNow` est le nom de l'élément `form`, `global1` est le nom de l'élément `select`, et `selectedIndex` est l'option sélectionnée. Comme `selectedIndex` est un nombre compris entre 0 et le nombre d'options de la balise `<select>` du conteneur, il peut être utilisé pour choisir l'élément de tableau (option) qui est sélectionné. Quelle que soit la valeur stockée, l'option est passée à la variable nommée `urlNow`. Par exemple, la ligne suivante a une URL relative égale à `animals/dogs.html` :

```
<option value="animals/dogs.html">Dogs</option>
```

La dernière ligne du code JavaScript, `window.location.href = urlNow`, a la même fonction que le code HTML5 suivant :

```
<a href="animals/dogs.html">
```

Dans ce contexte, une fonction JavaScript différente devrait être écrite pour chaque balise `<select>` car la fonction utilise une référence spécifique à cette balise (`global1`). Un code JavaScript plus sophistiqué pourrait être développé afin d'utiliser des variables pour les différents noms des éléments, mais la fonction employée ici est relativement courte et plus facile à implémenter.

Pour tester cet exemple, créez des pages Web simples ayant les noms suivants :

- `horses.html`
- `dogs.html`
- `cats.html`
- `rabbits.html`
- `raccoons.html`

Les pages Web peuvent être minimalistes et ne contenir que leur nom. Puis, dans le même répertoire, saisissez le code HTML5 suivant (`NavSelectJS.html` disponible dans les fichiers du chapitre).

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript" src="globMenu.js" />
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Menu déroulant</title>
</head>
<body>
<article>
  <header>
    <nav>
      <form name="menuNow">
        <label for="animals">Animaux</label>
        <select id="animals" name="global1" onChange="optionMenu()">
          <option value="animals/horses.html">Chevaux</option>
          <option value="animals/dogs.html">Chiens</option>
          <option value="animals/cats.html">Chats</option>
          <option value="animals/rabbits.html">Lapins</option>
          <option value="animals/raccoons.html">Ratons laveurs</option>
        </select>
        <label for="vegetables">Légumes</label>
        <select id="vegetables" name="global2">
          <option value="carrots">Carottes</option>
          <option value="squash">Courges</option>
          <option value="peas">Petits pois</option>
          <option value="rice">Riz</option>
          <option value="potatoes">Pommes de terre</option>
        </select>
        <label for="minerals">Mineraux</label>
        <select id="minerals" name="global3">
          <option value="tin">Étain</option>
          <option value="gold">Or</option>
```

```
            <option value="copper">Cuivre</option>
            <option value="iron">Fer</option>
            <option value="mercury">Mercure</option>
          </select>
        </form>
      </nav>
    </header>
  </article>
  </body>
  </html>
```

Testez la page en utilisant Google Chrome ou Opera car à l'époque de la rédaction de ce livre, c'étaient les deux seuls navigateurs à avoir implémenté cet aspect de HTML5.

Pour le moment, nous ne ferons rien des deux autres menus déroulants, mais à la fin du chapitre, nous verrons comment les compléter avec quelques ajouts dans le fichier JavaScript.

8.3 MAINTENIR LA COHÉRENCE

L'une des caractéristiques les plus importantes d'un bon système de navigation est la cohérence. L'utilisateur doit être capable de savoir où trouver le système de navigation, quelle que soit la page qu'il est en train de consulter. Si une page a le système de navigation au sommet et que sur le même site ce n'est pas le cas de la page suivante, les utilisateurs seront perdus. On trouve dans l'essai de Ralph Waldo Emerson, « La confiance en soi », une maxime sur la cohérence qui est très souvent mal citée. En ne citant qu'une partie de la réflexion d'Emerson, de nombreuses personnes croient à tort que la cohérence est une mauvaise chose. Voici cette célèbre citation erronée : « ... la cohérence est le diablotin des petits esprits... », alors que la citation complète est : « Une cohérence mal pensée est le diablotin des petits esprits, adorée par les petits hommes d'État, les petits philosophes et les petits théologiens. Avec cohérence, une grande âme n'a tout simplement plus rien à faire. » Emerson n'a en fait jamais dit que la cohérence était une mauvaise chose, car c'est la cohérence mal pensée qui est le problème, pas la cohérence.

En matière de navigation, la cohérence est essentielle et vous devez par tous les moyens éviter la cohérence mal pensée. En d'autres termes, il ne faut pas installer un mauvais système de navigation puis le dupliquer sous le prétexte que cela est cohérent. S'il y a cohérence, vous n'avez pas à réinventer le système de navigation à chaque nouvelle page. Une grande âme aurait différentes cohérences selon les publics et les types de sites, mais à l'intérieur d'un même site, il faut que la cohérence soit constante.

Dans ses travaux sur le regroupement des éléments, Jennifer Tidwell évoque l'utilisation des codes couleurs pour les sections, afin d'aider les utilisateurs à se repérer. En employant des couleurs, on peut ajouter de la clarté à la navigation globale. Les trois catégories globales qui ont été sélectionnées pour la navigation (animal, végétal et minéral) constituent un bon exemple de cohérence multiple (chaque menu est cohérent avec les autres menus). Pour la catégorie animale, vous pouvez utiliser des

tons de marron, pour la catégorie végétale, des tons de vert et pour la catégorie minérale des tons argentés. La figure 8.5 illustre un exemple de navigation globale où les différentes pages ont un modèle de couleurs qui les différencie les unes des autres, tout en les plaçant dans les groupes appropriés.

Figure 8.5 — Navigation globale et regroupement par couleur.

À la figure 8.5, vous noterez que la navigation globale incorpore la palette de couleurs des catégories respectives. Ce serait un bel exemple de cohérence mal pensée si les modèles de couleurs étaient identiques, alors que dans cet exemple, la navigation globale est cohérente et chaque page est cohérente avec les autres pages de la même catégorie.

8.3.1 Navigation verticale et horizontale

Outre l'utilisation du plan horizontal du haut et du bas de la page, les concepteurs d'interface réservent aussi souvent une portion du côté de la page Web pour la navigation. La figure 8.6 illustre la conception générale de cette approche.

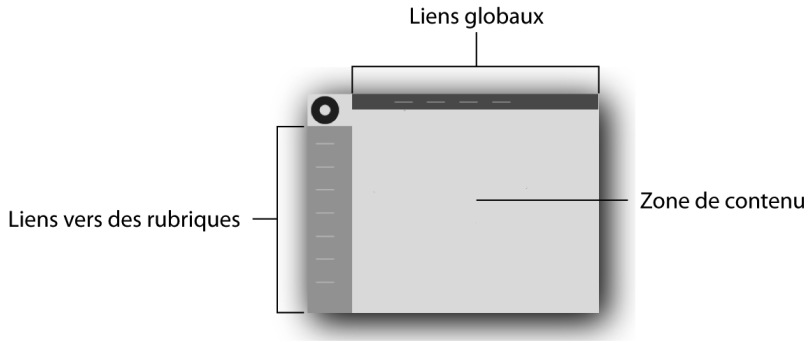


Figure 8.6 — Navigation verticale et horizontale.

Quand on utilise les plans horizontal et vertical, il est possible de voir simultanément tous les liens globaux et les liens de la rubrique en cours. Une grande partie

de la zone d'affichage est prise par le système de navigation, mais avec les écrans de grande taille qui deviennent la norme sur les ordinateurs, cela ne constitue pas un problème. Avec les tablettes comme l'iPad qui ont des écrans plus petits, cela prend de l'espace sur la zone d'affichage utile, mais il n'y a rien de catastrophique. En revanche, sur les téléphones mobiles, particulièrement en mode d'affichage vertical, l'espace du contenu est considérablement réduit.

Pour créer une zone de liens verticaux avec HTML5, on a simplement besoin de définir une page sur deux colonnes en dessous de la zone généralement réservée au logo et à la barre de navigation globale.

8.3.2 Application de pseudo-classes CSS3

Quand on gère des systèmes de navigation plus complexes, il est souhaitable de créer des pseudo-classes CSS3. Il s'agit de définitions de classes qui sont ajoutées à un élément. Pour la navigation, les quatre pseudo-classes suivantes sont importantes car elles sont associées à la balise `<a>` :

- `link`
- `visited`
- `hover`
- `active`

Chacune de ces classes a la même mise en forme que pour d'autres éléments, mais elle est déclarée avec le nom de l'élément séparé par le caractère deux-points. Par exemple, l'extrait de code suivant illustre le stylage de la pseudo-classe `hover` :

```
a:hover
{
     color:#A69055;
}
```

Quand ce code est ajouté à une feuille de styles, et lorsque la souris survole le lien (balise `<a>`), il modifie la couleur de la définition `hover`. Bien entendu, les couleurs définies pour la balise `<a>` doivent être différentes de celles de `hover`, mais vous pouvez ajouter à l'intention de l'utilisateur des signaux subtils ou flagrants que le texte constitue un lien. De la même manière, vous pouvez changer d'autres caractéristiques en utilisant les pseudo-classes. Les exemples suivants vous donneront quelques idées :

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
a {
     font-family:Verdana, Geneva, sans-serif;
     font-size:11px;
}
a:link {
```

```
        color:#cc0000;
        text-decoration:none;
}
a:hover {
        font-size:14px;
}
a:visited {
        color:#00cc00;
        text-decoration:none;
}
a:active {
        background-color:#ffff00;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Pseudo-classes en liens</title>
</head>
<body>
<a href="#">Cliquez ici</a>
</body>
</html>
```

Quand on utilise des pseudo-classes pour la navigation, il est souhaitable de ne pas perdre de vue l'utilisateur. L'ajout d'effets bizarres à l'aide de pseudo-classes peut être amusant, mais vous devez vous poser la question de savoir si ces effets vont aider les utilisateurs ou bien les perturber. Si vous pouvez ajouter un effet que les utilisateurs associent à la prise de décision, il est alors probable que cet effet sera utile. Par exemple, on peut agrandir la police quand la souris passe sur un lien, ce qui est une idée reprise du dock du Macintosh où les icônes s'agrandissent quand la souris les survole. Cependant, vous devez vous demander si le fait de modifier la couleur d'un lien ou bien de changer la taille de la police d'un lien est une bonne idée. Est-ce que cela constitue réellement une aide pour l'utilisateur ? Vous devez aussi tester ces modifications sur différents navigateurs et voir si les résultats sont cohérents. N'oubliez pas que ce n'est pas parce que vous pouvez modifier l'apparence d'un lien que cela signifie que vous devez le faire.

8.3.3 Mécanisme HTML5 de la navigation verticale

La partie la plus importante de la création d'une section verticale à utiliser pour la navigation de votre site est le découpage d'une portion de la page où l'on puisse placer les liens. Cet exemple utilise la balise `<aside>` pour délimiter la navigation verticale. Mais comme il s'agit de navigation, la balise `<nav>` est aussi utilisée de telle sorte que toute référence JavaScript au DOM (Document Object Model) puisse reconnaître la section comme une section utilisée pour la navigation. Le listing suivant (`VerticalHorizontal.html` disponible dans les fichiers du chapitre) montre comment réaliser cela.

```
<!DOCTYPE HTML>
<html>
<head>
```

```
<style type="text/css">
/*141919,2D2B21,A69055,C9B086,FFB88C -- Art japonais*/
body {
font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;
color:#2D2B21;
background-color:#C9B086;
font-size:12px;
}
.content {
display:table-cell;
width:600px;
padding:15px;
}
aside {
display:table-cell;
width:100px;
background-color:#FFB88C;
padding-right:5px;
}
h1 {
font-family:Papyrus;
color:#2D2B21;
text-align:center;
}
h2 {
color:#A69055;
}
a {
font-family:Verdana, Geneva, sans-serif;
font-size:10px;
text-decoration:none;
color:#141919;
}
a:hover {
color:#A69055;
}
.centerNav {
text-align:center;
}
.indentNav {
margin-left:15px;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Web Services à gogo</title>
</head>
<body>
<img src="designLogo.png" width="64" height="66" align="left">
<nav class="centerNav"> <a href="#">Conception graphique</a> |  <a
href="#">Développement</a> |  <a href="#">Conception
d'interface</a> |  <a href="#">Architecture de sites</a></nav>
<header>
  <h1>Web Services honorables</h1>
</header>
<aside>
  <nav class="indentNav"> <a  href="#">Aperçu général</a><br>
    <br>
```

```
      <a href="#">Navigation</a><br>
      <br>
      <a href="#">Abonnement RSS</a><br>
      <br>
      <a href="#">Iframes</a><br>
      <br>
      <a href="#">Styles de navigation CSS3</a><br>
      <br>
      <a href="#">Mesures d'audience</a><br>
      <br>
      <a href="#">Tests de qualité</a><br>
      <br>
      <a href="#">Ajout d'options mobiles</a><br>
      <br>
    </nav>
</aside>
<section class="content">
    <header>
      <h2>Conception d'interface</h2>
    </header>
    Web Services honorables peut concevoir toutes vos interfaces. Vous pouvez
choisir dans la liste suivante les services que vous désirez.
    <ul>
      <li>Interfaces simples de liens textuels</li>
      <li>Menus déroulants</li>
      <li>Liens avec des boutons</li>
      <li>Liens avec des listes de données</li>
      <li>Navigation dans des Iframes</li>
    </ul>
    Sélectionnez l'un des liens du menu de gauche pour obtenir plus
d'informations. Assurez-vous également de consulter nos services de conception
graphique, de développement et d'architecture dans le menu situé en haut de la
page.</section>
</body>
</html>
```

Quand on exécute ce programme, on peut voir que la présentation est claire bien qu'elle offre une grande variété de choix à l'utilisateur. La navigation globale en haut de la page offre la totalité des choix principaux, puis sur chaque page à l'intérieur d'une collection globale, les utilisateurs peuvent choisir des options spécifiques pour chaque rubrique sélectionnée. La figure 8.7 illustre le résultat dans un navigateur HTML5 sur l'écran d'un ordinateur.

Quand on visualise la même page sur un périphérique mobile, l'espace occupé sur la gauche, où le menu vertical a été placé, pousse le contenu vers le bas. Les utilisateurs doivent ainsi faire défiler un peu plus le contenu. Vous noterez aussi que le menu horizontal en haut de la page est resserré si bien qu'il tient à présent sur trois lignes. La figure 8.8 illustre le résultat sur un téléphone mobile.

Pour les périphériques mobiles, des barres de navigation horizontale à deux niveaux (qui ne poussent pas le contenu en dessous de la zone d'affichage) semblent préférables. Comme vous pouvez le voir en comparant les figures 8.7 et 8.8, la barre de navigation horizontale tient sur trois lignes sur le téléphone mobile sans occuper pour autant trop de place. En revanche, la barre de navigation verticale repousse le contenu (y

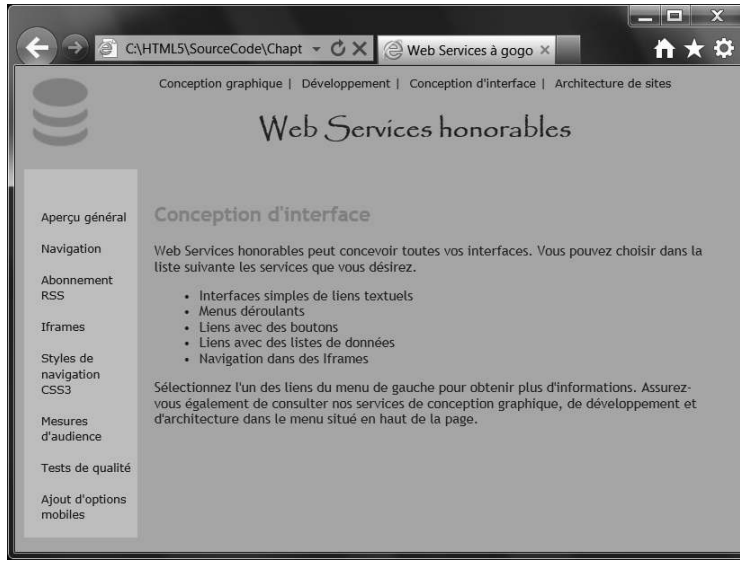


Figure 8.7 — Choix de navigation horizontale et verticale.

compris la barre de navigation elle-même) et le force à descendre en dessous de la zone d'affichage.

8.3.4 Utilisation d'icônes pour la navigation

Il est aussi possible d'utiliser des fichiers graphiques (JPEG, PNG ou GIF) pour créer des liens vers d'autres pages. L'emploi d'images pour les liens peut aider les utilisateurs à trouver rapidement ce qu'ils cherchent. Par exemple, une flèche vers la droite ou vers la gauche peut être rapidement identifiée comme un lien vers la page suivante ou la page précédente. De telles images transcendent les différentes langues et parlent à un public plus large. De la même manière, les jeunes enfants comprennent souvent plus facilement certains symboles que certains mots.

La syntaxe pour utiliser des images afin d'identifier les liens est la même que pour les liens textuels. Cependant, au lieu de placer du texte dans un conteneur `<a>`, on utilise une référence à une image. L'extrait de code suivant illustre la syntaxe de base :

```
<a href="page2.html"><img src="flecheDroite.jpg"></a>
```

Les utilisateurs voient l'icône d'une flèche et cliquent dessus au lieu de cliquer sur un message textuel. Souvent, les concepteurs utilisent à la fois du texte et une image pour créer un lien vers une autre page, comme dans l'exemple suivant :

```
<a href="page2.html"><img src="flecheDroite.jpg">Page suivante</a>
```

Figure 8.8 — Menus verticaux et horizontaux sur un terminal mobile.

Certains concepteurs créent aussi des icônes qui incorporent du texte, comme cela est illustré à la figure 8.9.

Figure 8.9 — Image de lien avec du texte.

Il y a un avantage à utiliser du texte graphique : le concepteur peut utiliser n'importe quelle police sans craindre que l'utilisateur ne dispose pas de cette police sur son système. Les symboles graphiques avec du texte favorisent la navigation car ils sont faciles à repérer et à comprendre.

8.4 SITES À PAGE WEB UNIQUE AVEC DES IFRAMES

Représentez-vous un site Web comme une zone de chargement. Quand on clique sur un lien, on charge une autre page (des images et tout le reste). Parfois, on ne souhaite

charger qu'une seule chose, ce qui gagnerait du temps car l'utilisateur n'aurait pas à charger ou recharger la totalité de la page. Si vous connaissez quelques rudiments de JavaScript et d'Ajax, vous pouvez réaliser cela, mais qu'en est-il si l'on se contente de HTML5 ? Cela est possible !

Cette section étudie la manière dont on fait un lien vers une image et on modifie l'image dans un iframe. Quand on crée des applications conçues spécifiquement pour des périphériques mobiles, on souhaite utiliser le moins de bande passante possible. En modifiant une seule chose sur une page Web, le terminal mobile n'a besoin de charger ou recharger qu'un seul élément, si bien que le temps de réponse est plus rapide.

8.4.1 Lien vers une image

Généralement, quand on évoque l'ajout d'une image sur une page, on pense à la balise `<img>`. Après tout, cette balise est ce que l'on utilise pour placer des images sur une page Web. Pourtant, on peut aussi utiliser la balise `<a href>` pour charger une image. Au lieu d'assigner un chemin de page Web à `href`, on peut tout aussi bien lui assigner une image. Par exemple, la ligne de code suivante charge une page vide avec une image :

```
<a href=="myGraphic.jpg">Mon image</a>
```

Quand l'utilisateur clique sur le texte du lien, la page en cours disparaît et l'image apparaît dans le coin supérieur gauche de la nouvelle page.

Le fait de placer une image dans un élément `iframe` fonctionne comme si l'on plaçait une page Web dans un iframe (voir le chapitre 7). Le lien pointe vers la cible à l'intérieur de l'iframe au lieu de pointer vers une autre page Web. Cela signifie que la page Web en cours reste en place et que l'image s'ouvre dans l'iframe.

Le script suivant utilise des icônes pour la navigation, mais au lieu de passer à une autre page, la navigation place une image différente dans la zone d'affichage principale (un iframe). En réalisant des versions miniatures des images à afficher (que l'on appelle des vignettes), les utilisateurs voient d'abord leur sélection dans la conception de la navigation. Cela signifie que les vignettes guident l'utilisateur vers l'affichage en plein écran.

8.4.2 Réalisation et utilisation d'icônes servant de vignettes

Afin de préparer l'application, il faut d'abord créer les versions en pleine taille des images, puis les vignettes associées. Les images et les vignettes doivent toutes être de la même taille. Dans l'exemple suivant, les images ont une taille de 250 x 312 pixels, et les vignettes de 63 x 79 pixels. Les vignettes doivent être suffisamment petites pour servir de bouton de navigation, mais aussi suffisamment grandes pour que l'utilisateur ait une idée de l'aspect de l'image en pleine taille. Vous remarquerez que les dimensions de l'iframe sont identiques à celles des images en pleine taille. Une fois que les images sont réalisées, elles sont placées dans des répertoires séparés (un répertoire pour les

vignettes, nommé thumbs, et un autre pour les images, nommé portraits). Le listing `NavigationIFrame.html` est disponible dans les fichiers du chapitre.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
/*F2CF8D,401E01,F2AA6B,8C3503,F28D52*/
body {
font-family:Verdana, Geneva, sans-serif;
background-color:#F2CF8D;
color:#401E01;
font-size:11px;
}
h1 {
font-family:"Harrington", Arial, sans-serif;
font-size:36px;
color:#8C3503;
margin-left:10px;
}
h4 {
font-family:"Arial Black", Gadget, sans-serif;
color:#8C3503;
margin-left:86px;
}
aside {
margin-left:10px;
}
h5 {
margin-right:40px;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Navigation Iframe</title>
</head>
<body>
<!DOCTYPE HTML>
<html>
<body>
<article>
  <header>
    <h1>Portrait Studio</h1>
  </header>
  <aside>
    <iframe name="fullSize" width="250", height="312" seamless
src="portraits/man.jpg"></iframe>
  </aside>
  <section>
    <nav> <a href="portraits/man.jpg" target="fullSize"><img
src="thumbs/thumbMan.jpg"></a> <a href="portraits/woman.png"
target="fullSize"><img src="thumbs/thumbWoman.png"></a> <a
href="portraits/boy.jpg" target="fullSize"><img src="thumbs/thumbBoy.jpg"></a>
<a href="portraits/girl.png" target="fullSize"><img
src="thumbs/thumbGirl.png"></a>
      <h4>Selectionnez un portrait</h4>
    </nav>
```

```
  </section>
  <section>
    <h5> Toutes les créations sont de <b>Mo Digli Anni</b>, artiste peu connu
habitant Spunky Puddle, dans l'Ohio. En cliquant sur les boutons des vignettes,
vous pouvez envoyer l'image dans une fenêtre de visualisation plus grande.</h5>
  </section>
</article>
</body>
</html>
```

Quand on teste l'exemple, on voit le portrait d'un homme puis les quatre vignettes représentant un homme, une femme, un garçon et une fille en dessous de l'image à l'intérieur de l'iframe. La figure 8.10 illustre la page affichée sur l'écran d'un ordinateur.

Comme vous pouvez le constater à la figure 8.10, on demande à l'utilisateur de cliquer sur les boutons des vignettes pour voir les différents portraits. L'interface est assez intuitive et l'utilisateur sait ce qui l'attend quand il clique sur l'un des boutons graphiques. L'intérêt de la chose est que seule l'image du portrait sélectionné est chargée dans l'iframe et non pas la totalité d'une nouvelle page avec tous les boutons graphiques et les autres composantes de la page.

Figure 8.10 — Images utilisées pour la navigation.

8.4.3 Utilisation d'iframes sur des terminaux mobiles

Quand on teste l'application sur un périphérique mobile, le résultat dépend du navigateur HTML5 utilisé. La figure 8.11 illustre le navigateur Opera Mini sur la gauche ; comme vous pouvez le constater, le texte en dessous de l'image est mis en forme de telle sorte qu'il soit lisible. Pourtant, pendant les tests, le navigateur Opera Mini semblait recharger la totalité de la page quand chaque bouton était sélectionné.

L'image de droite sur la figure 8.11 représente le navigateur mobile Safari. Le texte en bas ne respecte pas la mise en forme CSS3 et continue au-delà de la bordure droite de l'écran. Pour autant, les images dans l'`iframe` fonctionnent parfaitement et quand on clique sur une vignette, l'image en pleine taille se charge sans recharger la totalité de la page.

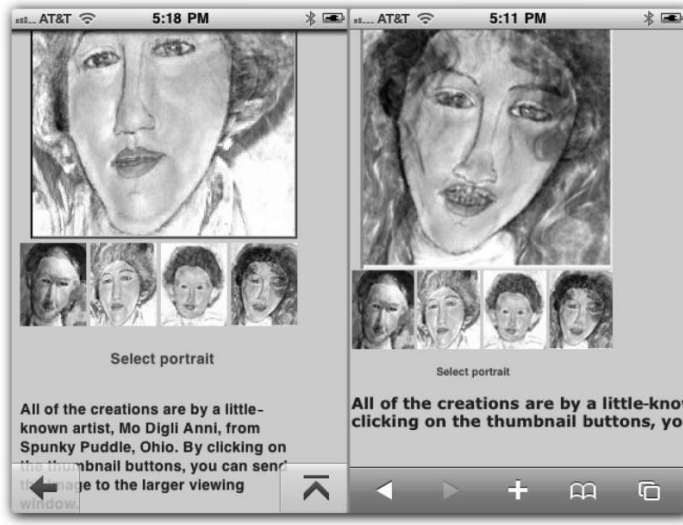


Figure 8.11 — Des navigateurs mobiles différents gèrent le texte différemment.

Plusieurs sites de réseaux sociaux et sites commerciaux utilisent des applications similaires. Par exemple, les photographes professionnels utilisent des vignettes de leurs photographies sur lesquelles on peut cliquer pour visualiser les images en pleine taille. De la même manière, les sites de réseaux sociaux utilisent des pages semblables pour afficher et charger les images des amis sans avoir à recharger la page.

Comme les périphériques mobiles ont une zone d'affichage réduite, l'utilisation des iframes facilite la navigation. Il peut être assez difficile de cliquer sur de petits liens textuels, mais comme vous pouvez le voir sur les deux copies d'écran de terminaux mobiles de la figure 8.11, les boutons graphiques sont faciles à voir et il est facile de cliquer dessus pour charger l'image en taille réelle ou tout autre type de contenu dans l'espace de l'iframe.

8.5 À VOUS DE JOUER !

Ce chapitre présente deux défis différents :

- **Défi JavaScript** : le premier défi consiste à achever le chaînage JavaScript de la section « Utilisation de JavaScript pour appeler une page liée ». La page HTML5 nommée `NavSelectJS.html` comporte trois balises différentes `<select>` (pour les animaux, les végétaux et les minéraux). Seule la balise `<select>` pour les animaux contient une fonction événementielle JavaScript. En ajoutant deux fonctions supplémentaires au fichier JavaScript (`globMenu.js`) qui sont similaires à la première fonction, mais avec un nom différent, vous devriez être capable de créer des fonctions pour les balises `<select>` des menus pour les végétaux et les minéraux (en fait, il suffit de copier et de coller deux fois la fonction originale, puis de simplement modifier le nom de la fonction). Ensuite, il faut ajouter l'attribut `OnChange` aux deux autres balises `<select>`. Les deux autres balises `<select>` ont les noms `global2` et `global3` que vous pouvez ajouter aux fonctions JavaScript (vous noterez l'emplacement de `global1` dans le fichier original JavaScript). Ne vous inquiétez pas si vous n'arrivez pas à réaliser cet exercice car si vous ne connaissez pas JavaScript, cela risque d'être difficile.
- **Défi Iframe** : Vous pouvez placer autant d'éléments `iframe` sur une page que vous le voulez. Supposons que vous vouliez comparer différents ensembles d'objets (des voitures, des habits ou des téléphones mobiles). Admettons que vous construisiez un site pour comparer différents modèles de voitures (par exemple, des Ford et des Toyota). Les Ford apparaissent dans l'`iframe` de gauche et les Toyota dans celui de droite. En dessous de chaque `iframe`, il y a des liens qui affichent différents types de voitures (économique, berline, hybride, camionnette, 4x4). Chaque marque comporte des liens pour chaque type de voiture si bien que vous pouvez les comparer. Votre tâche consiste à créer un tel site avec le contenu de votre choix. Edward Tufte recommande vivement que les informations comparatives soient présentées de telle sorte que les utilisateurs puissent les visualiser sur le même écran.

Ces deux défis, qui utilisent les notions qui ont été étudiées dans ce chapitre, peuvent être mis en pratique dans de nombreuses applications.

TROISIÈME PARTIE

Multimédia : images, sons et vidéos

9

Images

Objectif

Une des caractéristiques les plus intéressantes de HTML5 est sa capacité à utiliser les fichiers SVG (Scalable Vector Graphics). Ceux qui se servent de programmes comme Adobe Illustrator pour créer des dessins vectoriels peuvent enregistrer leurs fichiers au format `.svg` et les placer directement sur leurs pages Web. Comme les fichiers `.svg` sont des images au format vectoriel, on peut agrandir ou rétrécir les images sans perte de résolution, ce qui n'est pas le cas avec les fichiers bitmap. Pour autant, il est toujours possible d'utiliser vos images favorites au format bitmap (`.jpg`, `.gif` ou `.png`) pour les affichages statiques.

Ce chapitre tente de clarifier l'utilisation des images sur le Web en étudiant les principaux types d'images que l'on peut utiliser, la manière de les placer là où on le souhaite sur la page Web, et leur optimisation pour un usage sur le Web. Par nécessité, une grande partie de ce chapitre utilise des applications graphiques que vous ne possédez peut-être pas. Parmi ces applications, on peut citer Adobe Illustrator, Adobe Photoshop et Adobe Fireworks. Vous pouvez cependant les remplacer par d'autres applications, comme Microsoft Paint ou Corel Draw. Enfin, pour les dessins et les photographies, vous allez devoir compter sur vos propres talents artistiques et vos capacités à utiliser les logiciels graphiques (à la rigueur, vous pouvez télécharger sur le Web des images libres de droits dans le format de fichier dont vous avez besoin).

9.1 INTRODUCTION AUX FICHIERS IMAGE HTML5

La vérité première des fichiers graphiques sur le Web est qu'ils ont un poids. Dans le contexte d'une page HTML5, cela signifie que la taille ou le poids d'une image se

mesure au nombre de pixels qu'elle contient. En général, les images de grande taille en haute définition contiennent plus de pixels. La taille des images a une conséquence sur le Web : les images les plus lourdes mettent plus de temps à circuler sur Internet et à se charger dans une page HTML. Si vous vous êtes déjà trouvé dans la situation de celui qui regarde fixement son écran en attendant le chargement d'une page Web contenant une image de taille importante, vous savez combien cela est frustrant et peut provoquer des clics incontrôlés pour actionner le bouton Précédent du navigateur.

Si vous arrivez à comprendre les subtilités des différents types de fichiers graphiques et à optimiser leur taille, vous tirerez le meilleur parti des images sur vos pages, à la fois en termes d'aspect et de temps de chargement.

9.1.1 Les formats et les pixels ont de l'importance

Ce qui compte le plus sur un écran, c'est l'aspect d'une image. L'apparence d'une image dépend en grande partie de la résolution de l'écran. Plus la résolution d'un moniteur ou d'un périphérique mobile est importante, meilleur sera l'aspect de l'image. De plus, une image contenant plus d'informations paraîtra meilleure qu'une image contenant moins d'informations. Cela signifie aussi qu'une image qui occupe plus de place à l'écran va nécessiter plus d'informations qu'une image plus petite et mettra par conséquent plus de temps à se charger.

Pour bien comprendre comment créer de belles images qui ne consomment pas trop de bande passante et qui se chargent rapidement, il faut étudier les différents types de formats graphiques du Web. Les quatre sous-sections suivantes fournissent un aperçu de chacun des formats.

Format SVG (Scalable Vector Graphics)

Les images vectorielles sont des dessins créés en utilisant des formules mathématiques qui spécifient des points et tracent ensuite des lignes entre ces points. Les images au format bitmap (en mode point) placent des bits colorés à différents endroits. Par exemple, si vous dessinez une ligne droite dans une image vectorielle, l'ordinateur prend le Point A et le Point B puis trace une ligne entre ces deux points. La même ligne tracée dans une image au format bitmap spécifie tous les points afin de placer tous les bits qui composent la ligne (cette explication simplifie à l'excès le processus, mais elle fournit une idée générale de la différence entre une image vectorielle et une image bitmap).

Comme les images vectorielles utilisent des formules, elles ne sont pas pixellisées quand on change leur taille, ce qui n'est pas le cas des images bitmap. Imaginez une ligne de 100 pixels de long que l'on veut agrandir en une ligne de 400 pixels de long. Avec une image vectorielle, il suffit de modifier la distance entre les deux points. Avec une image bitmap, il faut ajouter 300 pixels. Si vous tentez de modifier une ligne au format bitmap sur une page Web en faisant passer sa largeur de 100 pixels à 400 pixels, cela va étirer la ligne originale pour qu'elle atteigne une largeur de 400 pixels et c'est la raison pour laquelle on verra les pixels.

L'autre caractéristique importante des fichiers au format SVG est que l'on peut modifier différents aspects de l'image dynamiquement. Avec JavaScript, vous pouvez prendre un fichier `.svg` affiché sur une page Web et le changer de manière dynamique, non pas en permutant des figures, mais en modifiant un paramètre. Heureusement, certains outils récemment sortis facilitent la création de zones séparées à modifier et génèrent le code nécessaire pour effectuer ces modifications (voir la section « Application pour les fichiers SVG dynamiques à partir de fichiers CS5 Adobe Illustrator » plus loin dans ce chapitre).

Format GIF (Graphic Information Format)

L'avantage des fichiers GIF est qu'ils peuvent produire des fichiers de taille très réduite et qu'ils prennent en charge la transparence de l'arrière-plan. En grande partie, cela est dû au fait qu'ils ne peuvent gérer que 256 couleurs et qu'ils obtiennent la transparence en désactivant l'une des couleurs. L'ensemble des couleurs, connu sous le nom de couleurs « web-safe », est basé sur le fait que les fichiers GIF ne peuvent gérer que 256 couleurs. Ce format est extrêmement limitatif pour les concepteurs qui veulent une plus grande palette de couleurs, et il n'est pas recommandé pour les photos numériques autres que celles en noir et blanc avec des niveaux de gris limités.

Les fichiers GIF peuvent être animés. Si un fichier GIF animé est chargé, il fait défiler séquentiellement les images qui le composent, ce qui donne l'impression d'une animation. En raison du fait qu'un GIF animé est stocké en un seul fichier, il peut être chargé directement dans une page HTML5 avec la balise `<img>`. En général, les fichiers GIF animés sont relativement courts, sinon l'ensemble des fichiers à l'intérieur du GIF animé serait trop grand pour permettre un chargement rapide.

Outre le nombre limité de couleurs disponibles pour la création d'images au format GIF, CompuServe et Unisys détiennent un brevet sur ce format et revendiquent le paiement d'une licence d'utilisation. Plutôt que de risquer d'être poursuivis, la plupart des développeurs privilégient simplement d'autres formats graphiques.

Format JPEG (Joint Photographic Experts Group)

La plupart des photos numériques sur le Web utilisent le format JPEG. De la même manière, toute image complexe avec plusieurs couleurs et teintes privilégie le format JPEG pour préserver l'aspect voulu par le photographe ou l'artiste. Cela a pour conséquence que la plupart des images des sites Web qui proposent des services ou des produits sont au format JPEG. Les fichiers JPEG ont tendance à être plus grands que les fichiers GIF, mais avec l'accroissement de la bande passante sur Internet, la taille n'est plus aussi problématique qu'autrefois.

Le format JPEG ne prend pas en charge la transparence comme les fichiers GIF, et il n'a pas non plus de format animé. De plus, les fichiers JPEG utilisent ce que l'on appelle la compression avec perte, ce qui peut réduire la fidélité de l'image. Comparée à la compression sans perte qui implique une réplication exacte des données originales, la compression avec perte est plus considérée comme une approximation des données originales qui composent l'image.

La norme JPEG est un format open source qui ne nécessite l'octroi d'aucune licence. Il faut cependant noter que certaines fonctionnalités des images au format JPEG sont brevetées et peuvent nécessiter une licence, mais ces caractéristiques ne sont pas incluses dans la plupart des fichiers JPEG si bien que les développeurs et les concepteurs peuvent utiliser librement le format JPEG.

Format PNG (Portable Network Graphics)

Le format PNG a été développé pour partie comme une alternative au format GIF qui est breveté et nécessite une licence. L'autre motivation concernait aussi la volonté d'avoir plus de 256 couleurs et un affichage sans perte. Le format PNG prend aussi en charge la transparence et un canal alpha.

Pendant un certain temps, tous les navigateurs ne prenaient pas en charge le format PNG et certains développeurs ne l'utilisaient pas en dépit de ses nombreux avantages. Mais ce temps est révolu et tous les navigateurs qui supportent HTML5 prennent en charge le format PNG. Cela a pour conséquence que tout développeur ou concepteur HTML5 peut utiliser des fichiers PNG sans crainte que le navigateur ne puisse les charger.

9.1.2 Préservation des calques dans les images Web

L'un des principaux avantages des fichiers PNG est qu'ils préservent les calques. Les concepteurs qui utilisent des outils comme Adobe Illustrator, Adobe Fireworks et Adobe Photoshop organisent leurs images en calques. La légende d'une image constitue une simple application des calques. Par exemple, supposons que vous ayez une photographie que vous avez légendée et enregistrée au format JPEG, comme cela est illustré à la figure 9.1.

Après avoir terminé l'image et l'avoir enregistrée au format JPEG, vous vous apercevez que vous vous êtes trompé de légende (il ne s'agit pas d'une marguerite, mais d'une gloire du matin). En raison du fait que le fichier est enregistré au format JPG, le calque avec la légende Daisy n'a pas été préservé. Quand vous allez modifier le fichier, vous constaterez que la légende est fusionnée avec le reste de l'image.

Avec un fichier PNG, non seulement les calques sont préservés, mais si l'on utilise un arrière-plan transparent, il sélectionne l'arrière-plan de la page Web, et la transparence est également conservée. La figure 9.2 montre que la simple permutation des calques règle le problème de légende et offre un arrière-plan transparent.

Avec les images comportant plusieurs calques, l'enregistrement de l'image finale Web au format PNG économisera du temps lors de l'édition. Dans cet exemple particulier, l'effacement de la légende erronée dans le fichier JPEG et son remplacement par la bonne légende dans l'espace en dessous de l'image n'est pas une tâche trop difficile, mais avec des images plus complexes qui comprennent plusieurs calques, les concepteurs n'ont pas à refaire la totalité de l'image et ils peuvent se contenter de modifier juste un calque.

Le seul problème fâcheux dans la préservation des calques des fichiers PNG est que cela accroît la taille du fichier. Le fichier JPEG ne pèse que 33 Ko, alors que le

Figure 9.1 — Image JPEG avec une légende intégrée.

Figure 9.2 — Fichier PNG avec un calque préservé et un arrière-plan transparent (illustré dans un éditeur graphique).

fichier PNG en fait 225. Nous verrons cependant dans la section suivante comment conserver les calques tout en réduisant la taille du fichier, de manière à avoir quand même un fichier qui se charge rapidement.

9.2 TAILLES DES FICHIERS GRAPHIQUES

Étant donné les différentes sortes de fichiers graphiques du Web qui peuvent être chargés, la tentation est grande d'utiliser le format qui a la plus petite taille de fichier. Dans certains cas, ce sera effectivement l'attitude à adopter, mais quand votre site a besoin d'une qualité très élevée, l'astuce consiste à étudier comment obtenir la meilleure qualité en utilisant le moins de bande passante. À moins que vous n'utilisiez le format SVG, rappelez-vous l'axiome de base du Web concernant les images bitmap :

Ne changez jamais les dimensions d'une image bitmap avec les attributs HTML5 au sein d'un élément.

Vous pouvez modifier à loisir les dimensions d'une image avec une application graphique comme Adobe Photoshop ou Microsoft Paint, mais quand on change la taille d'une image bitmap avec des attributs HTML5 tels que `width` et `height`, cela provoque, particulièrement quand on tente d'agrandir un objet, une pixellisation de l'objet ou son écrasement. La figure 9.3 montre trois images GIF, et vous pouvez constater que l'image agrandie a des bords dentelés et que les pixels commencent à apparaître sous la forme de petites boîtes.

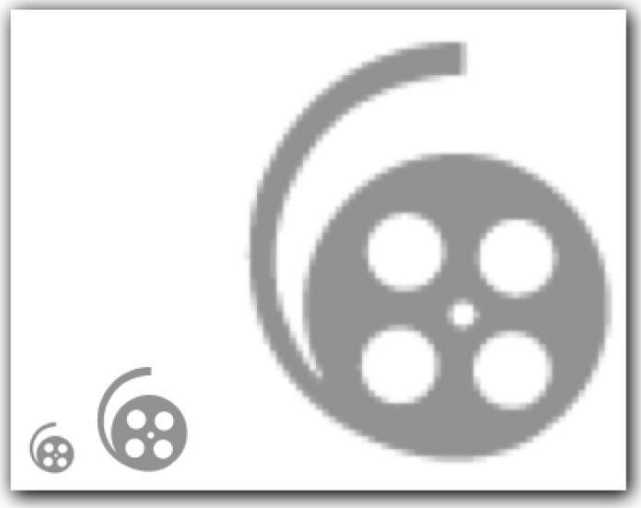


Figure 9.3 — Fichier GIF agrandi avec des attributs HTML5.

La figure du milieu est l'image originale avec ses dimensions originales. Si l'on avait utilisé un outil graphique pour agrandir l'image, elle n'apparaîtrait pas avec des bords dentelés. Vous pouvez constater à la figure 9.4 que le même phénomène se produit avec des photos numériques.

L'image originale est à l'extrême gauche alors que l'image agrandie montre des bords dentelés et commence à se brouiller. L'image à l'extrême droite est si petite qu'il est difficile de distinguer les détails et de déterminer si elle est écrasée (les pixels

ont tendance à se tordre). Utilisez le programme suivant (`DistortionImage.html` disponible dans les fichiers du chapitre) pour tester vos propres images.

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Distortion d'une image Web</title>
</head>
<body>
<!-- Originale -->
<img src="photo.jpg" width="100" height="127">
<!-- Élargie à 400% -->
<img src="photo.jpg" width="400" height="508">
<!-- Réduite à 50% -->
<img src="photo.jpg" width="50" height="63.5">
</body>
</html>
```

Figure 9.4 — Photo numérique JPG agrandie et rétrécie avec des attributs HTML5.

Pour trouver la largeur et la hauteur d'une image, utilisez la souris pour sélectionner le fichier et :

- Sous Windows, faites un clic droit et sélectionnez Propriétés → Détails puis lisez les valeurs Largeur et Hauteur. Vous pouvez trouver les dimensions d'un fichier image en plaçant le pointeur de la souris sur le fichier.
- Sous Mac OS X, faites un Ctrl + clic sur le fichier image et sélectionnez Obtenir les informations. Dans la section Plus d'informations, affichez les Dimensions qui indiquent Largeur x Hauteur.

La plupart des outils Web, comme Dreamweaver, fournissent des informations sur les dimensions des images. De la même manière, pratiquement tous les éditeurs graphiques affichent les dimensions des images quand un fichier est chargé.

9.2.1 Utilisation d'applications graphiques pour modifier la taille des fichiers image

Quand on évoque la taille d'une image, on fait référence à deux notions différentes :

- Les dimensions de l'image, c'est-à-dire son nombre de points (largeur x hauteur) ;
- Le nombre d'octets du fichier.

Normalement, le contexte permet de préciser de quelle acception il s'agit quand on parle de taille d'une image, mais la plupart du temps, le terme taille fait référence au nombre d'octets du fichier, alors que le terme dimensions fait référence à la taille de l'image à l'écran.

Adobe Photoshop est une application couramment utilisée pour retoucher la taille et la qualité des images. De plus, Photoshop fournit des informations visuelles que les concepteurs et les développeurs peuvent utiliser pour décider du pourcentage de compression que l'image peut supporter avant que son apparence ne se dégrade. Les figures 9.5 et 9.9 illustrent ce processus (les figures 9.6 à 9.8 montrent des informations sur les fichiers et la manière dont elles apparaissent sur une page Web).

Modification de la taille des fichiers JPEG

Si l'on commence par un très gros fichier TIFF qui doit être converti en un fichier PNG, JPEG ou GIF, le processus d'édition du fichier démarre par trois niveaux de qualité : maximum, moyen et bas. La figure 9.5 illustre le fichier original TIFF et trois fichiers JPEG.

Le fichier original TIFF situé dans le coin supérieur gauche pèse près d'un demi-mégaoctet et il doit donc être considérablement allégé et converti en un format lisible par les navigateurs HTML5. La figure dans le coin supérieur droit est au format JPEG en qualité maximale (100). La figure dans le coin inférieur gauche est en basse qualité (2), et celle du coin inférieur droit est en qualité moyenne (60). Le plus petit fichier Web ne pèse que 8,6 Ko et le plus grand fait 127,1 Ko. Un rapide coup d'œil montre très peu de différences entre ces deux images.

Figure 9.5 — Affichage d'une image et de sa taille sous quatre formes différentes.

Pour se faire une idée plus précise, les deux fichiers extrêmes en termes de qualité Web sont enregistrés sur le disque, puis (sur un Macintosh) on visualise la taille de chaque fichier (voir la figure 9.6).

Si l'on examine la figure 9.6, on s'aperçoit que les deux fichiers ont les mêmes dimensions (432 x 343), mais que l'un pèse 12 Ko alors que l'autre en fait 139 Ko. Les études sur la perception ont montré que l'examen des différences minimes a tendance à les atténuer, alors que les différences extrêmes apparaissent clairement ; c'est la raison pour laquelle quand on commence à faire des ajustements sur les images, il est préférable de commencer par les grandes différences. La figure 9.7 illustre deux fichiers sur une page Web.

Comme vous pouvez le voir, l'image de qualité la plus basse (à gauche) et l'image de qualité la plus élevée (à droite) sont très similaires. Avec d'autres images sur une page Web, des différences de qualité pourraient apparaître, mais avec les caractéristiques de celles qui sont illustrées à la figure 9.7, la réduction de la taille ne modifie pas tellement l'apparence sur le Web.

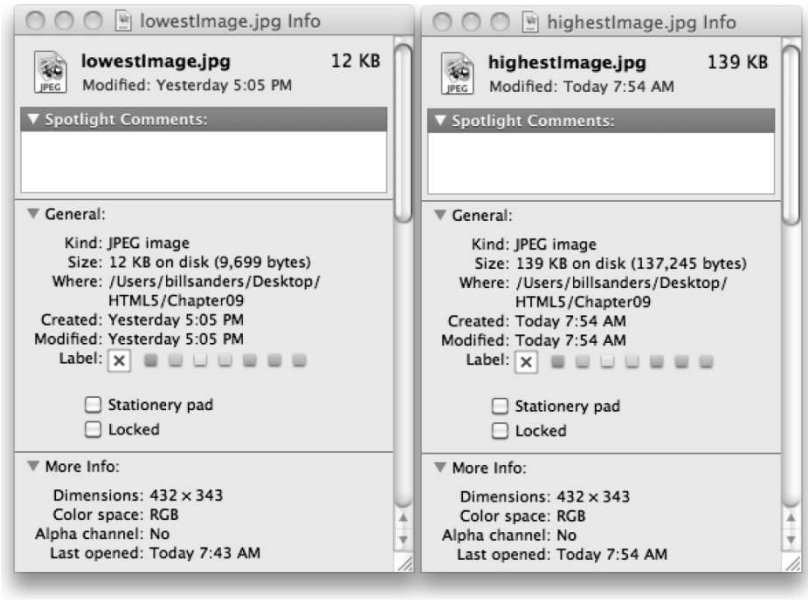


Figure 9.6 — Affichage des propriétés des fichiers image.

Figure 9.7 — Fichiers JPEG en haute qualité et en basse qualité sur une page Web.

On peut constater une plus grosse différence avec des photos numériques au format JPEG. À la figure 9.8, la photo de gauche est en qualité la plus basse (8 Ko) et celle de droite est en qualité la plus élevée (115 Ko).

Les différences entre les deux photos sont minimes, mais la différence en kilooctets est assez importante (8 Ko contre 115 Ko). Sur un écran d'ordinateur, l'image de gauche de la figure 9.8 a une définition plus pauvre autour des bords, mais si la majeure partie du public d'un site Web a une très faible bande passante disponible, la réduction de la taille des fichiers JPEG ne réduira pas de manière significative la qualité des images.

Figure 9.8 — Photos numériques au format JPEG en qualité basse et élevée.

L'image de la figure 9.8 a été prise avec une webcam, et des photos numériques prises par un appareil doté d'un capteur de grande qualité montreraient beaucoup plus de détails qui seraient susceptibles d'être perdus avec la perte d'informations induite par la réduction de la taille des fichiers. Mais les photos numériques en très haute résolution doivent être réduites considérablement si l'on veut s'en servir pour le Web.

Un bon éclairage économise la bande passante

Quel que soit le type d'appareil photo que vous utilisez, une image bien éclairée paraîtra toujours meilleure qu'une image mal éclairée. Tout ce que l'on voit (et votre appareil photo voit) est la réflexion de la lumière sur les objets. Si vous accordez un peu d'attention à l'éclairage de votre sujet, vos photos numériques seront meilleures.

Vous n'avez pas besoin de la lumière d'un studio pour prendre de bonnes photos, mais en ajoutant correctement de la lumière vous pourrez supprimer plus d'informations du fichier tout en conservant une qualité suffisante pour l'afficher sur le Web. Voici quelques conseils :

1. **Utilisez un éclairage indirect.** Si vous prenez des photos quand il y a un ciel nuageux, les photos seront en général meilleures. Cela est dû au fait que les nuages diffusent la lumière (avez-vous déjà vu ces photos où les malheureux sujets plissent les yeux face au soleil ? Non seulement ils font une drôle de tête, mais les photos sont surexposées). Pour les photos d'intérieur, dirigez une lumière sur un papier blanc et faites-la réfléchir sur le sujet. Une feuille de papier aluminium fait très bien l'affaire pour diffuser la lumière.
2. **Utiliser la lumière naturelle si possible.** Si vous prenez des photos d'intérieur, ouvrez les rideaux et les stores pour laisser pénétrer la lumière naturelle.

Modification de la taille des fichiers PNG et GIF

Si l'on aborde maintenant la modification de la taille des fichiers PNG et GIF, les différences ont tendance à être plus significatives en ce qui concerne la réduction de la taille des fichiers et les informations qui sont de ce fait supprimées. Examinez la figure 9.9, et vous verrez immédiatement que les différents paramètres jouent sur les niveaux de qualité.

À la figure 9.9, les deux images du haut sont des fichiers GIF et les deux images du bas des fichiers PNG. Quand les fichiers GIF sont réduits, ils perdent des couleurs. L'image en haut à gauche n'a que 32 couleurs et celle de droite en a 256 (ce qui n'est pas non plus énorme). Si l'on compare la taille des deux fichiers GIF, celui de gauche fait la moitié de celui de droite. Comparez cela aux différents niveaux de qualité des fichiers JPEG utilisés à la figure 9.5.

Les deux fichiers PNG sont libellés PNG-24 (image de gauche avec des couleurs codées sur 24 bits) et PNG-8 (image de droite avec des couleurs codées sur 8 bits). Le format PNG-8 n'a que 128 couleurs, alors que le format PNG-24 peut gérer des millions de couleurs. En bref, le format PNG-24 est de qualité supérieure.

Figure 9.9 — Modification de la taille des fichiers PNG et GIF.

Modification de la taille des fichiers SVG

À la différence des images bitmap, la modification de la taille des fichiers au format SVG est simple et ne change pas l'aspect de l'image. Le code suivant de la page Web illustre un fichier `.svg` de 500 x 400 affiché en différentes tailles qui sont déterminées par les attributs `width` et `height`. Le script suivant (`SVG.html` disponible dans les fichiers du chapitre) n'utilise qu'un seul fichier `.svg` et l'affiche dans différentes tailles sans aucune distorsion.

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Test SVG</title>
</head>
<body style="background-color:#BAD9CB" >
<!-- Safari, Chrome et Opera -->
<img src="logo500x400.svg" width=100 height=80>
<img src="logo500x400.svg" width=200 height=160>
<img src="logo500x400.svg" width=300 height=240><br>
<img src="logo500x400.svg" width=500 height=400>
<!-- Firefox et Opera
<object width=100 height=80 type="image/svg+xml"
data="logo500x400.svg"></object>
<object width=200 height=160 type="image/svg+xml"
data="logo500x400.svg"></object>
<object width=300 height=240 type="image/svg+xml"
data="logo500x400.svg"></object><br>
<object width=500 height=400 type="image/svg+xml"
data="logo500x400.svg"></object>
-->
</body>
</html>
```

Au cours de la rédaction de ce livre, Firefox n'utilisait pas la balise `<img>` avec les fichiers `.svg`, mais nécessitait à la place l'emploi de la balise `<object>` (le navigateur Opera quant à lui fonctionne avec les deux formats). La figure 9.10 illustre le résultat et vous pouvez voir que le logo paraît identique, quelle que soit la taille dans laquelle il est affiché.

Niveaux de gris avec Internet Explorer

Il est intéressant de noter quand on travaille sur les tailles des fichiers que l'on peut utiliser une propriété CSS spéciale qui n'est reconnue que par Internet Explorer. Certains concepteurs emploient des niveaux de gris pour réduire la taille de leurs images ou pour l'effet des nuances de gris. Si vous voulez tester cette option intéressante dans Microsoft Internet Explorer, vous pouvez écrire un bout de code CSS pour convertir des fichiers en couleur en niveaux de gris. Il suffit pour cela d'utiliser l'extrait de code suivant dans une définition de style :

Figure 9.10 — Image SVG modifiée sans distorsion en changeant des attributs.

```
<style type="text/css">
img {
     filter:gray;
}
</style>
```

La figure 9.11 illustre une image en couleur (voir la figure 9.8) qui est transformée en niveaux de gris uniquement par l'utilisation de code CSS.

Grâce à cette technique, vous pouvez voir rapidement l'aspect de l'image en niveaux de gris avant de modifier l'image. Si vous mettez à jour un site et que vous souhaitez visualiser les images en niveaux de gris, vous pouvez ajouter le code CSS et tester d'abord avec Internet Explorer. Si vous voulez diminuer la taille des fichiers tout en conservant une certaine qualité, vous pouvez passer vos images couleur JPEG en niveaux de gris, ce qui divisera leur taille par deux.

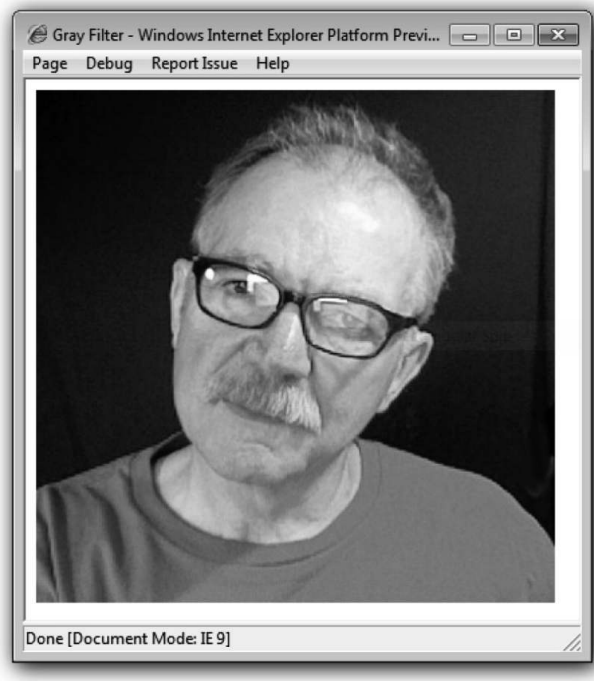


Figure 9.11 — Utilisation du filtre CSS de niveaux de gris d'Internet Explorer.

9.3 PLACEMENT DES IMAGES ET CRÉATION DE PAGES WEB FLEXIBLES

La balise `<img>` est la balise qui est majoritairement utilisée pour afficher des images et bien que CSS3 soit l'outil principal pour positionner correctement les choses sur une page Web, vous pouvez cependant utiliser certains attributs de la balise `<img>` pour le faire. Cette section étudie les options de placement du texte sur une page Web et la manière d'utiliser certains attributs de la balise `<img>`.

9.3.1 Placement des images avec l'attribut align

Nous allons commencer l'étude du placement en examinant l'attribut `align` de la balise `<img>`. Son principal avantage réside dans le fait qu'il n'y a pas de moyen plus facile de positionner rapidement une image par rapport à un texte. Le script suivant (`PlacementImage.html` disponible dans les fichiers du chapitre) illustre le procédé.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
/*048ABF,049DBF,F2F2F2,595959,0D0D0D*/
```

```
body {
background-color:#F2F2F2;
color:#0D0D0D;
font-family:Verdana, Geneva, sans-serif;
}
h1 {
font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;
color:#595959;
background-color:#049DBF;
text-align:center;
}
h2 {
color:#048ABF;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Placement simple</title>
</head>
<body>
<article>
  <header>
    <h1>La gym du développeur Web</h1>
  </header>
  <section>
    <header>
      <h2>Séance d'entraînement du développeur</h2>
    </header>
    <figure> <img src="webDeveloper.gif" width="250" height="263"
align="left"> </figure>
    Vous y avez déjà pensé. N'est-ce pas le moment de faire vos balises ?
Développez vos éléments et vos attributs HTML5 à la Salle de gym du Web. Une
fois que vous serez échauffé, vous pourrez ajouter une petite balise
&lt;canvas&gt; et vous lancer dans une bonne séance de CSS3. La salle est
ouverte tous les jours et 24 heures sur 24 ; de plus, elle est accessible à
partir de n'importe quel endroit de la planète ! Tous vos amis sont là et ils
ont même ajouté des vidéos à leurs pages Web ! Vous aussi, vous pouvez le
faire ! Ne laissez pas passer un autre jour pour réaliser votre rêve de
devenir un développeur Web. Commencez maintenant ! </section>
</article>
</body>
</html>
```

Le placement à droite ou à gauche de l'image est simple car il suffit d'assigner les valeurs `"right"` ou `"left"` à l'attribut `align`. La figure 9.12 illustre le placement de l'image à la fois à gauche et à droite.

À la figure 9.12, la page de droite paraît correcte, mais la page de gauche serre le texte trop près de l'image. Vous noterez également que la page est totalement dépendante de la taille et des paramètres de page de l'utilisateur. En d'autres termes, l'utilisation de l'attribut `align` pour le placement des images peut ruiner l'aspect de vos pages. La figure 9.13 illustre deux autres affichages de la même page qui n'ont pas le même aspect.

À la figure 9.13, l'image de gauche affiche un texte disséminé sur toute la page, alors que l'image de droite, qui est visualisée sur un téléphone mobile, montre que le

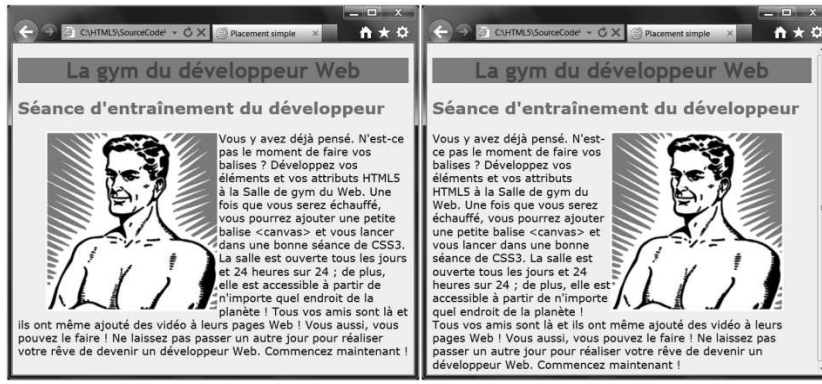


Figure 9.12 — Placement des images avec l'attribut align.

texte ne s'affiche que sur une largeur d'un mot, la totalité du texte n'apparaissant pas dans la zone d'affichage.

Figure 9.13 — Différents affichages d'une même page.

9.3.2 Taille d'image flexible avec un peu de JavaScript

Au chapitre 12, vous comprendrez mieux ce qui se passe, mais j'ai besoin dans cette section de JavaScript pour vous montrer comment vos pages peuvent devenir plus flexibles en affichant des images de taille différente. JavaScript a une propriété appelée `navigator.appVersion`. Quand cette propriété est placée dans un script, vous pouvez récupérer des informations sur le matériel qui est utilisé pour charger la page Web. Si vous déterminez que la page est chargée sur un périphérique mobile, vous pouvez charger une image plus petite au lieu de charger sur la page Web une image en pleine taille.

Pour faire fonctionner cela, nous allons prendre le même fichier GIF que celui qui a été utilisé pour la création de la page Web originale de la précédente section, et nous allons créer une autre image plus petite qui fera à peu près un tiers de l'image originale. Créez un dossier que vous nommerez `flexImages`, et placez-y les deux fichiers GIF

que vous nommerez `WebDeveloper.gif` pour le plus grand et `lilWebDeveloper.gif` pour le plus petit. Saisissez ensuite le programme suivant (`ImageTailleFlexible.html` disponible dans les fichiers du chapitre) et enregistrez-le dans le répertoire qui contient le dossier `flexImages`.

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
var envir=navigator.appVersion;
envir=envir.substring(5,11);
var imageNow=new Image();
var showNow;
function showImage()
{
if(envir=="iPhone" || envir=="(iPhon")
{
showNow="flexImages/lilWebDeveloper.gif";
}
else
{
showNow="flexImages/WebDeveloper.gif";
}
imageNow.src=showNow;
document.pix.src=imageNow.src;
}
</script>
<style type="text/css">
/*048ABF,049DBF,F2F2F2,595959,0D0D0D*/
body {
background-color:#F2F2F2;
color:#0D0D0D;
font-family:Verdana, Geneva, sans-serif;
}
h1 {
font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;
color:#595959;
background-color:#049DBF;
text-align:center;
}
h2 {
color:#048ABF;
}
img {
padding:5px;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Taille flexible</title>
</head>
<body onLoad="showImage()">
<article>
  <header>
    <h1>La gym du développeur Web</h1>
  </header>
```

```
  <section>
    <header>
      <h2>Séance d'entraînement du développeur</h2>
    </header>
    <figure> <img src="flexImages/WebDeveloper.gif" name="pix" align="left">
</figure>
    Vous y avez déjà pensé. N'est-ce pas le moment de faire vos balises ?
Développez vos éléments et vos attributs HTML5 à la Salle de gym du Web. Une
fois que vous serez échauffé, vous pourrez ajouter une petite balise
&lt;canvas&gt; et vous lancer dans une bonne séance de CSS3. La salle est
ouverte tous les jours et 24 heures sur 24 ; de plus, elle est accessible à
partir de n'importe quel endroit de la planète ! Tous vos amis sont là et ils
ont même ajouté des vidéos à leurs pages Web ! Vous aussi, vous pouvez le
faire ! Ne laissez pas passer un autre jour pour réaliser votre rêve de
devenir un développeur Web. Commencez maintenant ! </section>
</article>
</body>
</html>
```

Commencez par tester le programme sur votre ordinateur. Vous devriez voir exactement le même résultat que celui qui est illustré à la figure 9.12. Essayez ensuite ce programme sur un périphérique mobile : au lieu de voir s'afficher une grande image qui pousse le texte sur le côté, vous voyez une image plus petite qui est encadrée parfaitement par le texte, comme le résultat obtenu sur votre ordinateur. Cela est dû au fait que la page Web a été capable d'utiliser JavaScript pour déterminer si la page était chargée sur un iPhone ou sur une autre plateforme.

Placez la page Web ainsi que le dossier contenant les deux images dans le même répertoire sur un serveur. Quand vous testez cette page, on dirait qu'elle a été faite pour un iPhone, mais en fait elle a été faite pour iPhone ou pour n'importe quel autre périphérique. Grâce au code JavaScript, vous pouvez vraiment étendre les possibilités de HTML5.

Le code JavaScript utilisé dans cet exemple minimaliste peut être expliqué de la manière suivante :

- On place le contenu de `navigator.appVersion` dans une variable nommée `envir` (abréviation d'environnement).
- Comme `navigator.appVersion` génère une description longue, on ne récupère que la partie des résultats qui nous intéresse et on regarde si elle contient « iPhone » ou non.
- On crée un nouvel objet image appelée `imageNow`.
- On initialise une variable nommée `showNow` (que nous utiliserons dans la fonction).
- On crée une fonction qui demande : « S'agit-il d'un environnement d'iPhone ? » Si tel est le cas, on utilise la petite image. Dans le cas contraire, on utilise la grande image (il y a une bizarrerie dans Opera Mini : JavaScript retourne `"(iPhon"` quand on extrait les six premiers caractères de `navigator.appVersion` ; le code doit donc demander s'il trouve les deux valeurs, `"(iPhon"` ou `"iPhone"`), ce qui montre le côté accommodant de JavaScript).

Figure 9.14 — Affichage d'une petite image pour un téléphone mobile.

Bien entendu, il y a beaucoup plus de modèles de périphériques mobiles et il vous faudra adapter le code JavaScript pour ajouter d'autres modèles en plus de l'iPhone, mais la logique reste la même (il vous faudra simplement un peu plus de JavaScript).

Au fait, si vous n'avez jamais travaillé avec JavaScript, ne soyez pas étonné de ne pas comprendre le code de la page Web. Cette démonstration n'a pour but que d'illustrer ce qui peut être fait avec JavaScript. L'avenir du Web passe par l'intégration de nombreuses plateformes différentes de navigation et cette petite démonstration n'est qu'un avant-goût de ce que vous pouvez réaliser (si vous être un programmeur JavaScript chevronné, vous pouvez créer quelque chose d'un peu plus élégant !).

9.3.3 Application pour les fichiers SVG dynamiques à partir de fichiers CS5 Adobe Illustrator

Adobe Illustrator CS5 (en abrégé AI) possède une nouvelle fonctionnalité, Adobe Illustrator CS5 HTML5 Pack, qui est disponible à `http://labs.adobe.com`. Cette fonctionnalité est conçue pour permettre aux graphistes utilisant AI de convertir facilement leurs fichiers `.ai` en fichiers `.svg` contenant des parties qui peuvent être

modifiées dynamiquement avec HTML5. Pour vous donner une idée de la manière dont cela fonctionne, l'exemple suivant commence avec une image simple dans AI. Elle possède deux calques et sur l'un des calques, le concepteur désire une couleur variable qui puisse être codée en HTML5. Le calque auquel on va donner une caractéristique variable est sélectionné et affiché dans le panneau Apparence (voir la figure 9.15).

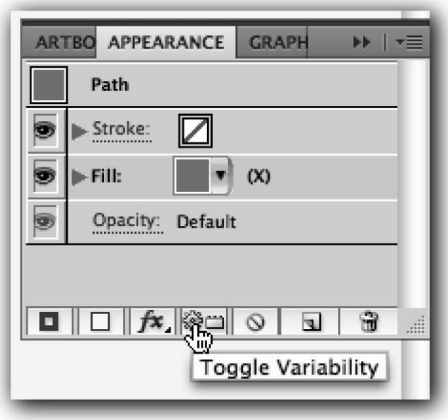


Figure 9.15 — La couleur de remplissage est définie comme une variable dans Adobe Illustrator CS5.

Le (X) de la figure 9.15 indique que la couleur de remplissage (Fill) est une variable qui peut être modifiée avec HTML5. Afin de pouvoir modifier cette caractéristique (la couleur de remplissage dans ce cas), AI génère le code pour le fichier SVG qui peut être visualisé et/ou modifié pendant la conversion du fichier `.ai` en fichier `.svg`. Au cours du processus de conversion, le concepteur clique sur le bouton Afficher le code SVG, et trouve le nom du calque qui possède une caractéristique variable. Dans cet exemple, voici le code spécifique SVG :

```
<g id="Button">
     <ellipse fill="param(SVGID_2__FillColor) #A35563" cx="50" cy="50" rx="40"
ry="40.5"/>
</g>
```

L'identifiant ayant la valeur `Button` est le nom du calque dans AI. La valeur `param`, `SVGID_2__FillColor`, est générée automatiquement par AI.

Pour exploiter les informations SVG dans un programme HTML5, le fichier `.svg` doit être référencé dans un élément `<object>` et le paramètre d'une balise `<param>`. Le fichier JavaScript `Param.js` est aussi généré automatiquement par AI et doit être chargé dans conteneur `<head>` afin que Firefox analyse correctement le code. Le code suivant (`AI2svg.html` disponible dans les fichiers du chapitre) fonctionne avec les navigateurs Firefox, Safari, Chrome et Opera, mais il existe certaines différences d'affichage.

```
<!DOCTYPE HTML>
<html>
<head>
<script src="Param.js"></script>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>AI -> SVG</title>
</head>
<body>
<article>
  <section>
    <figure>
      <object type="image/svg+xml" data="butnBkground.svg">
        <!--Pas de balises param-->
      </object>
    </figure>
    <figure>
      <object type="image/svg+xml" data="butnBkground.svg">
        <param name="SVGID_2__FillColor" value="#cc0000" />
      </object>
    </figure>
  </section>
</article>
</body>
</html>
```

Afin d'illustrer la séquence du processus, la figure 9.16 illustre le fichier original AI et le résultat de l'affichage dans Opera quand la page `AI2svg.html` se charge.

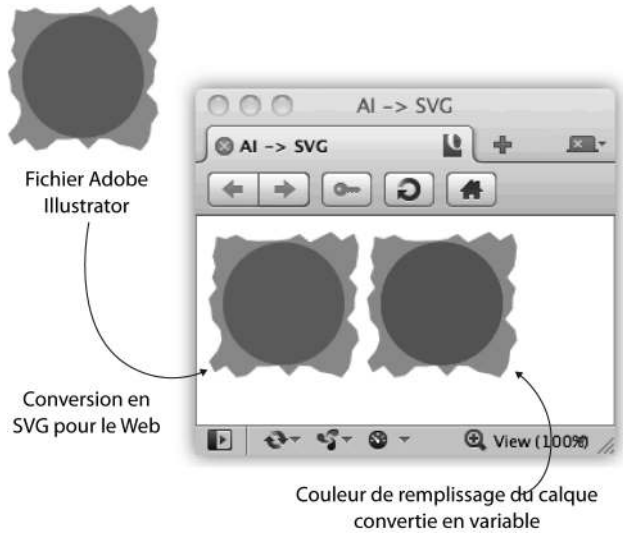


Figure 9.16 — Le fichier original AI est transformé en un fichier au format SVG avec une couleur de remplissage variable.

Les fichiers `Param.js` et `.svg` doivent être dans le même dossier que la page HTML5, tout comme les feuilles de styles externes CSS3 et les fichiers graphiques sont censés se trouver dans le même dossier que le code HTML5 qui les appelle, ou bien dans le

chemin spécifié par le code HTML5. Ce qui est intéressant dans cette technique, c'est que les concepteurs et les développeurs peuvent se concentrer sur les balises HTML5 alors qu'Adobe Illustrator CS5 s'occupe de la génération du code JavaScript et des noms des paramètres. Bien entendu, cela implique que les concepteurs sachent utiliser les images vectorielles et créent des fonctionnalités dynamiques dans leurs images AI.

9.4 À VOUS DE JOUER !

Le premier exercice est une chasse au trésor Web. Vous pouvez trouver de nombreux outils gratuits sur le Web qui peuvent être utilisés pour modifier la taille d'une image (à la fois ses dimensions et son poids en octets). Même si vous disposez d'Adobe Photoshop ou de Microsoft Paint, explorez le Web pour trouver une application qui fonctionne sur votre ordinateur (vous pouvez même en trouver plusieurs si vous le souhaitez).

Prenez un fichier graphique qui ne soit pas au format JPEG, PNG ou GIF. Vous pouvez, par exemple, trouver un fichier `.tif` ou `.tiff` (il peut s'agir d'une photo numérique ou d'un dessin, voire une combinaison des deux). Réalisez ensuite les étapes suivantes :

1. **Convertissez le fichier dans les formats JPEG, PNG et GIF.**
2. Vous avez à présent quatre fichiers : le fichier original et les trois formats Web.
3. **Faites une copie de tous les fichiers graphiques Web en les nommant de telle sorte que le nom du fichier indique qu'il s'agit d'une qualité basse.**
4. **Par exemple, si vous avez un fichier nommé `voiture.jpg`, copiez-le en le nommant `voitureBasse.jpg`.**
5. **En utilisant l'application graphique que vous avez dénichée sur le Web, créez une version en qualité supérieure et une version en qualité basse de chacun des trois types de fichiers.**
6. **En utilisant HTML5 et CSS3, créez une page Web comportant trois lignes. Sur le côté gauche placez les images en qualité supérieure et sur le côté droit, placez les images en qualité basse.**
7. **Entre toutes les images, placez un texte de remplissage de votre choix.**
8. Vous pouvez chercher sur Internet le célèbre texte de remplissage *lorem ipsum*. La figure 9.17 illustre le format général de la page Web.

Cet exercice a deux objectifs :

- Vous entraîner à placer du texte autour d'images en utilisant les techniques CSS3 que vous avez apprises précédemment car l'emploi de l'attribut `align` de la balise `<img>` a de sérieuses limites.
- Faire passer l'idée que toutes les modifications sur les images doivent être réalisées avec un logiciel graphique avant d'assembler votre page Web.

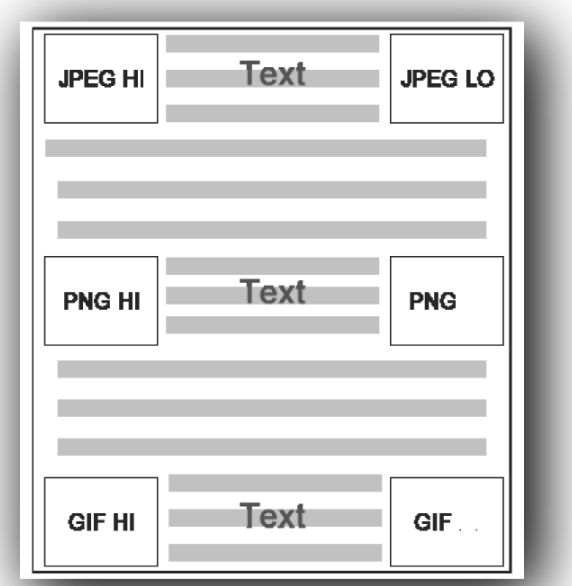


Figure 9.17 — Affichage des différentes sortes et qualités d'images avec du texte.

Ceux qui désirent exploiter les images vectorielles peuvent tester Adobe Illustrator CS5 HTML5 Pack. Si vous ne possédez pas Adobe Illustrator CS5, vous pouvez télécharger gratuitement une version d'évaluation valable 30 jours. Tentez de créer des parties variables dans une conception AI en utilisant plusieurs calques avec des noms qui deviendront les noms des ID du paramètre que vous modifierez.

10

Son

Objectif

L'ajout de sons aux pages Web permet aux développeurs de créer une large palette de sites Web. Les sites qui jouent de la musique, qui fournissent un contenu éducatif ou bien qui ajoutent des effets sonores étendent les possibilités de ce que l'on peut réaliser avec HTML5. Ce chapitre étudie la manière de préparer des sons pour le Web et leur utilisation pour rendre vos pages Web sonores.

Vous apprendrez à travailler avec les différents attributs et paramètres de la balise `<audio>`. Vous verrez aussi comment les différents navigateurs gèrent le son et les différents fichiers sonores. Comme pour les images, des programmes spécialisés sont disponibles pour créer des fichiers audio et les modifier. Après avoir étudié les éléments et les attributs HTML5 de base, ce chapitre vous montrera donc comment créer des sons pour vos sites Web.

10.1 INTRODUCTION À LA GESTION DES CONTENUS AUDIO EN HTML5

La balise `<audio>` est l'une des nouvelles balises HTML5 les plus intéressantes. Grâce à elle, vous pouvez jouer des fichiers audio sur les haut-parleurs de votre ordinateur ou bien dans les écouteurs d'un périphérique mobile. Voici la syntaxe de base pour sélectionner un fichier audio à jouer :

```
<audio src="jazz.mp3"></audio>
```

L'attribut `src` fonctionne de la même manière que pour une balise `<img>` : il s'agit d'une référence à la source du fichier. Mais pour écouter le fichier audio, on a besoin d'attributs.

10.1.1 Autoplay

L'attribut `autoplay` se passe d'explication. Dès que la page se charge, le fichier sonore commence à être joué. Avant d'ajouter l'attribut `autoplay`, il est souhaitable de s'assurer que l'utilisateur est bien d'accord pour écouter les sons que vous jouez. Si vous voulez faire fuir vos utilisateurs, le plus sûr moyen est d'avoir un son continu qui est joué automatiquement. Ce souci mis à part, le script suivant (`AudioBasique.html` disponible dans les fichiers du chapitre) montre comment créer une page simple qui commence à jouer de la musique dès qu'elle est chargée :

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Audio de base</title>
</head>
<body>
Le son est entre les lignes<br>
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
-
<br>
<audio src="jazz.wav" autoplay></audio>
<br>
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
-
</body>
</html>
```

Vous pouvez tester ce script avec n'importe quel navigateur, excepté Google Chrome car c'est le seul qui ne reconnaît pas les fichiers sonores au format `.wav`. Utilisez à la place un fichier `.mp3` ou `.ogg` pour tester Chrome.

10.1.2 Controls

Comme nous vous l'avons fait remarquer, si votre son (musique, effets sonores, conversation, etc.) ennuie les utilisateurs, ils ne reviendront pas sur votre site. Dans ces conditions, comment faire pour contrôler le son ? Le moyen le plus simple est d'ajouter l'attribut `controls`. Comme pour `autoplay`, vous n'avez pas à fournir de valeur et il suffit de l'inclure à la balise `<audio>` pour qu'il apparaisse automatiquement. Testez le programme suivant (`Controles.html` disponible dans les fichiers du chapitre) :

```
<!DOCTYPE HTML>
<html>
<style type="text/css">
/* 694703,A83110,E89F06,F5D895,B3CF83 */
```

```
body {
background-color:#B3CF83;
font-family:Verdana, Geneva, sans-serif;
color:#694703;
}
h1 {
font-family:Braggadocio, "Arial Black";
color:#A83110;
}
</style>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Contrôles</title>
</head>
<body>
<article>
  <header>
    <h1>Veillée Jazz</h1>
  </header>
  <section>
    <p>Cliquez sur le triangle pour démarrer la musique : </p>
    <audio src="mists.ogg" controls></audio>
    <p>Le symbole des deux barres verticales (||) arrête la musique.</p>
  </section>
</article>
</body>
</html>
```

Quand vous exécutez ce programme, assurez-vous d'utiliser un navigateur compatible avec le fichier audio (employez un fichier `.wav` si le type de fichier `.ogg` ne fonctionne pas avec votre navigateur). En fonction du type de navigateur utilisé, vous verrez que les contrôles sont différents. La figure 10.1 illustre l'aspect des différents navigateurs (Google Chrome est présenté alors que la musique est en cours de lecture).

La barre de contrôle audio est classique et le bouton de démarrage en forme de triangle se situe à gauche, alors que le bouton de contrôle du volume se situe à droite. Le bouton stop/pause fonctionne de manière identique sur tous les navigateurs, mais les graphismes sont uniques (ces différences ne plairont sans doute pas aux concepteurs qui tentent de créer des pages avec un contenu audio totalement compatible avec tous les navigateurs).

Le fait de donner une certaine forme de contrôle aux utilisateurs est essentiel. Le navigateur Chrome offre une belle grosse barre qui permet à l'utilisateur de voir clairement où il en est de l'écoute du fichier audio. Pour les contenus pédagogiques, la barre verticale que vous pouvez voir dans le navigateur Chrome à la figure 10.1 est importante car l'étudiant peut revenir en arrière pour réécouter certaines parties d'une leçon qui sont difficiles à comprendre.

10.1.3 Preload

L'attribut `preload` de la balise `<audio>` peut se révéler important car il commence le préchargement du contenu audio avant qu'il ne soit joué. De cette manière,

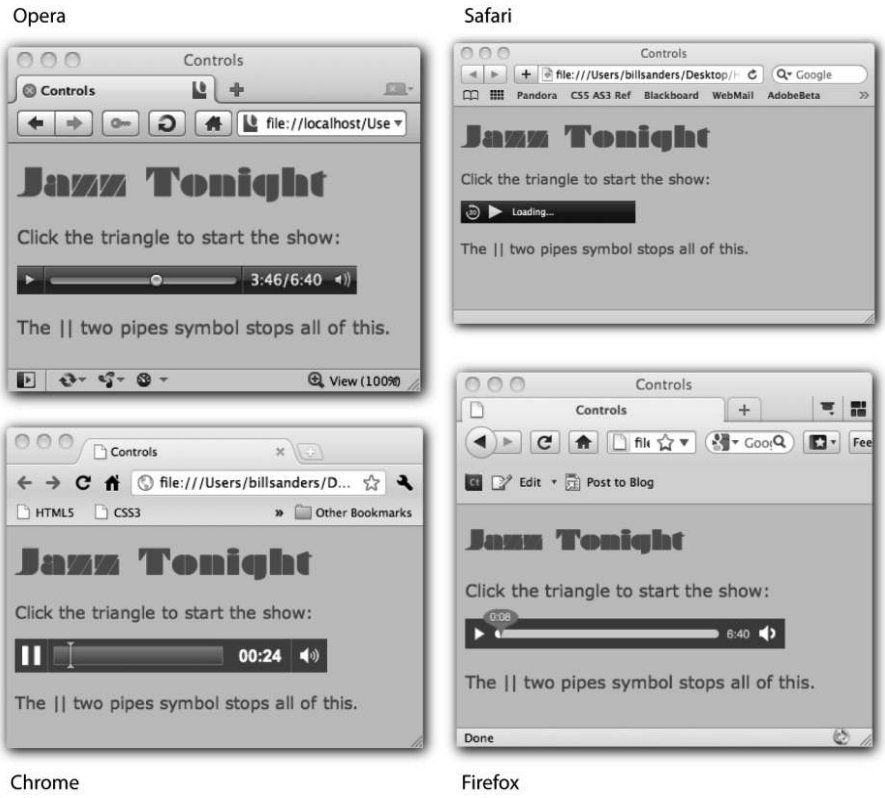


Figure 10.1 — Utilisation des contrôles d'un lecteur audio.

l'utilisateur n'a pas à se tourner les pouces en attendant le chargement du fichier audio une fois qu'il a cliqué sur la touche de lecture. Dans sa forme la plus simple, l'attribut `preload` n'a pas besoin de valeur et doit être appelé de la manière suivante :

```
<audio src="Shadows.wav" preload controls></audio>
```

Quand on emploie le préchargement, on peut utiliser l'autoplay, mais je ne suis pas certain que cela soit très logique. L'autoplay démarre la lecture de l'audio dès que la page est chargée, alors que le préchargement est utilisé pour charger un fichier audio avant que la commande de lecture ne soit déclenchée.

Vous pouvez assigner des valeurs à l'attribut `preload` :

- `none` : cette valeur peut sembler étrange, mais certains navigateurs peuvent être paramétrés pour précharger automatiquement les fichiers audio. Si l'utilisation d'un fichier audio particulier est peu probable, le développeur peut décider de ne pas utiliser de ressources Internet et assigne donc la valeur `none` à l'attribut `preload`.
- `metadata` : tous les fichiers audio (et vidéo) contiennent des métadonnées comme la durée ou d'autres données que l'auteur du contenu audio place dans

le fichier. Quand l'utilisation d'un fichier audio est incertaine, il est cependant raisonnable de charger les métadonnées car cela consomme peu de ressources Internet.

- `auto` : si l'attribut `preload` est présent, il précharge automatiquement les informations du fichier audio. L'assignation auto agit simplement comme un rappel que le fichier va être préchargé (c'est comme si l'on n'avait assigné aucune valeur à l'attribut `preload`).

Plus votre public sera varié, plus vous aurez de contenus audio sur vos pages Web, ce qui rendra souhaitable l'utilisation des options de l'attribut `preload`.

10.1.4 Boucle

Quand on veut qu'un son se répète indéfiniment, on utilise une boucle. Cela a l'avantage que vous pouvez prendre un morceau de musique relativement court et le répéter de telle manière qu'il apparaîtra comme une composition complète. De cette manière, vous minimisez l'utilisation des ressources Internet et disposez d'une musique en continu. La syntaxe d'utilisation est similaire à celle des autres attributs qui agissent comme des interrupteurs : ils sont actifs ou inactifs. Voici un exemple :

```
<audio src="Shadows.wav" autoplay loop></audio>
```

Cette ligne porte en elle les germes de sa propre destruction. Pour de multiples raisons, les utilisateurs ont tendance à vouloir désactiver le son. Vous pouvez utiliser JavaScript pour réaliser cela, mais il est plus simple d'ajouter l'attribut `controls` et de laisser l'utilisateur désactiver le son. Cependant, certains concepteurs, à juste titre, ne souhaitent pas incorporer la barre de contrôle dans leurs pages Web car ils pensent qu'une jolie musique fait partie intégrante de la conception. Dans ce cas, il faut songer à élaborer une routine JavaScript pour désactiver la musique. Même si le morceau de musique est magnifique, sa répétition incessante s'apparente à du lavage de cerveau, ce qui est interdit par la Convention de Genève.

10.2 PRISE EN CHARGE DE L'AUDIO PAR LES NAVIGATEURS

Au moment de la rédaction de cet ouvrage, j'ai testé différents formats audio et je n'ai pas réussi à trouver un seul format qui soit pris en charge par tous les navigateurs HTML5. Le tableau 10.1 illustre la répartition.

Comme vous pouvez le voir, le seul format audio qui est presque pris en charge par tous les navigateurs est le format `.wav`. La bonne nouvelle est que les fichiers `.wav` sont largement disponibles et que vous pouvez trouver pratiquement n'importe quel son dans ce format. Mais si une partie significative de votre public préfère le navigateur Google Chrome, vous allez avoir besoin d'un plan B.

Tableau 10.1 — Navigateurs et prise en charge des formats audio

Navigateur	MP3	WAV	OGG
Chrome	Oui	Non	Oui
Firefox	Non	Oui	Oui
Internet Explorer 9	Oui	Non	Non
Opera	Non	Oui	Non
Safari	Oui	Oui	Non

10.3 SOURCE À LA RESCOUSSE : PLAN B

Habituellement, si vous avez à déterminer quel navigateur fonctionne avec quelles ressources, vous devez faire appel à JavaScript. Heureusement, HTML5 possède un élément qui propose différents formats audio et laisse le navigateur choisir celui qui est compatible.

La balise `<source>` peut être placée à l'intérieur du conteneur `<audio>` en indiquant la source et l'URL de la ressource audio au sein de la balise `<source>`. Supposons que vous exploitiez un site Web qui héberge des contenus audio pédagogiques. Au lieu de dire à chaque utilisateur qu'il doit employer un certain type de navigateur, il vous suffit de disposer des fichiers audio pour tous les navigateurs possibles et de laisser le navigateur sélectionner ceux qui lui conviennent. Admettons par exemple que vous affichiez la troisième leçon sur une page Web. L'extrait de code suivant fournit une sélection de fichiers qu'aucun navigateur ne pourrait refuser :

```
<audio controls>
  <source src="lecon3.ogg">
  <source src="lecon3.mp3">
  <source src="lecon3.wav">
</audio>
```

La corvée d'avoir à créer plusieurs versions des fichiers audio risque d'être fastidieuse, mais même si vous programmez cela en JavaScript, vous aurez besoin de plusieurs versions des médias (au chapitre 9, rappelez-vous qu'il nous a fallu plusieurs versions des fichiers graphiques pour les plateformes mobiles et non mobiles et que nous avons utilisé JavaScript pour savoir quelle page devait être affichée sur un iPhone).

10.3.1 Attribut type

Quand on définit plusieurs types différents de sources audio afin de s'assurer que les navigateurs HTML5 pourront les lire, on peut améliorer le processus en ajoutant l'attribut `type` à la balise `<source>`. L'information de l'attribut `type` dit au navigateur s'il doit tenter de charger le fichier, comme dans l'extrait suivant :

```
<source src="mists.ogg" type="audio/ogg">
```

L'inclusion de l'attribut `type` sert à gagner du temps. L'interpréteur du navigateur lit la ligne et détermine s'il pourra ou non jouer le type de fichier indiqué. S'il ne peut pas le jouer, il ne s'embêtera pas à essayer de le faire. Supposons par exemple que vous ayez le choix de passer deux examens : l'un sur HTML5 et l'autre en physique quantique. À moins que vous n'ayez une formation en physique quantique, vous n'allez même pas gaspiller votre temps à passer cet examen, alors que vous allez tenter votre chance avec le devoir sur HTML5. Il en va de même avec l'attribut `type`. Quand il voit le type et détermine qu'il ne peut pas le jouer, il n'essaye même pas.

Si l'attribut `type` n'est pas précisé, le navigateur va tenter de charger le contenu audio, et s'il n'arrive pas à le jouer, il va poursuivre avec la balise `<source>` suivante et tenter de la jouer.

L'extrait suivant liste tous les types :

```
<audio controls>
  <source src="lecon3.ogg" type="audio/ogg">
  <source src="lecon3.mp3" type="audio/mpeg">
  <source src="lecon3.wav" type="audio/wav">
</audio>
```

Toutes les valeurs doivent être des types MIME valides. Les types valides suivent la règle `media-type` définie dans les spécifications W3C de HTML5. L'attribut `type` est optionnel, mais si votre site a un trafic important, il vous aidera à supprimer tout appel inutile. Si vous voulez creuser cette question, vous devez aussi vous pencher sur le paramètre `codec` qui est expliqué dans la section suivante.

10.3.2 Paramètre codec du type de la source

En général, si vous saisissez une valeur pour l'attribut `type`, le type général est la seule information que vous avez à fournir. Cependant, quand plusieurs codecs sont disponibles, vous devez ajouter les codecs que le navigateur sait lire. Encore une fois, la spécification du codec ne va pas permettre au navigateur d'accéder à un certain codec s'il n'est pas capable de le lire. Au contraire, cela fournit une information au navigateur de telle sorte que s'il ne peut pas le lire, il ne tentera même pas de le faire. C'est comme un vendeur de journaux qui demanderait : « Que voulez-vous ? Nous avons des journaux en français, en anglais et en espagnol. » Si vous savez lire le français et l'anglais, vous pouvez choisir des journaux dans ces langues, mais si vous ne lisez pas l'espagnol, vous n'allez pas tenter de lire un journal dans cette langue.

Avant de détailler le paramètre codec, nous allons expliquer ce que c'est. Le terme codec est une combinaison des mots compression et décompression, ce qui signifie que quand on parle de codec, on décrit la manière dont un fichier est encodé (habituellement compressé) et décodé (décompressé pour qu'il puisse être lu).

Le type de codec, même si les types de fichiers sont identiques, peut être différent. Afin d'accélérer le processus qui détermine si le fichier peut être lu, l'ajout du paramètre

codec permet de filtrer les types de codecs que le navigateur ne sait pas lire. Par exemple, tous les fichiers `.ogg` suivants ont des codecs différents :

```
<source src="songFest.ogg" type="audio/ogg; codecs=vorbis">
<source src="songFest.spx" type="audio/ogg; codecs=speex">
<source src="audio.oga" type="audio/ogg; codecs=flac">
```

Vous devez donc vous souvenir que les codecs et les types de fichiers sont deux choses différentes. Si vos pages Web peuvent utiliser des informations complètes sur les codecs d'un fichier, il est préférable de le faire. Dans le cas contraire, certains navigateurs peuvent tenter de jouer un son pour finir par trouver que le codec est incompatible.

Certains types de fichiers audio possèdent une large gamme de codecs. L'extrait de code suivant illustre des codecs typiques pour tous les fichiers sonores HTML5 qui peuvent être lus par les navigateurs HTML5 :

```
<audio controls>
  <source src="sound.ogg" type="audio/ogg; codecs=vorbis">
  <source src="jazz.mp3" type="audio/mpeg; codecs=mp3">
  <source src="Shadows.wav" type="audio/wav; codecs=wav">
</audio>
```

L'extrait ci-dessus n'affiche pas tous les codecs possibles de tous les types audio, mais il représente les principaux types de codecs utilisés sur Internet.

10.4 CRÉATION DE FICHIERS AUDIO

Windows 7 et Macintosh OS X incluent tous les deux des programmes que vous pouvez utiliser pour vos propres fichiers audio. Ils sont intégrés à votre ordinateur et à moins que vous ne les ayez supprimés, ils sont prêts à enregistrer du son.

Les versions précédentes de Windows possédaient une application d'enregistrement du son, mais elles avaient un aspect différent de celle qui est utilisée dans cet exemple. L'application Magnétophone qui faisait partie de Windows XP enregistrait les fichiers au format `.wav` de telle sorte qu'ils étaient prêts pour les pages Web, mais la nouvelle version du Magnétophone qui est livrée avec Windows 7 n'enregistre les fichiers qu'au format `.wma`, ce qui nécessite une conversion pour que les fichiers soient reconnus par les navigateurs HTML5.

10.4.1 Enregistreur de son de Windows 7

La première chose à faire quand on réalise un enregistrement est de définir le type de microphone. La plupart des ordinateurs qui tournent sous Windows 7 ont un microphone intégré et vous pouvez donc l'utiliser. Dans le cas contraire, vous devez brancher un microphone et vous assurer qu'il est correctement configuré. Habituellement, votre ordinateur peut trouver les pilotes audio dont il a besoin,

mais certains microphones sont livrés avec des pilotes que vous devez installer (les instructions d'installation sont fournies avec le microphone).

Pour choisir un microphone, utilisez le chemin suivant : Panneau de configuration > Matériel et audio > Gérer les périphériques audio. Quand la fenêtre Son s'ouvre, sélectionnez l'onglet Enregistrement, et vous verrez une liste de périphériques d'enregistrement disponibles, comme ceux qui sont illustrés à la figure 10.2.

Vos périphériques d'enregistrement peuvent être différents, mais en général vous avez au moins le choix entre microphone interne ou microphone ligne entrée. Après avoir fait votre choix, cliquez sur le bouton OK, et vous êtes maintenant prêt à ouvrir l'application Magnétophone.

À partir du menu Démarrer, sélectionnez Tous les programmes > Accessoires > Magnétophone (si vous exécutez Windows XP, choisissez Tous les programmes > Accessoires > Divertissement > Magnétophone). La figure 10.3 illustre l'aspect du Magnétophone quand il est prêt à enregistrer (en haut) et quand il est en train d'enregistrer (en bas).

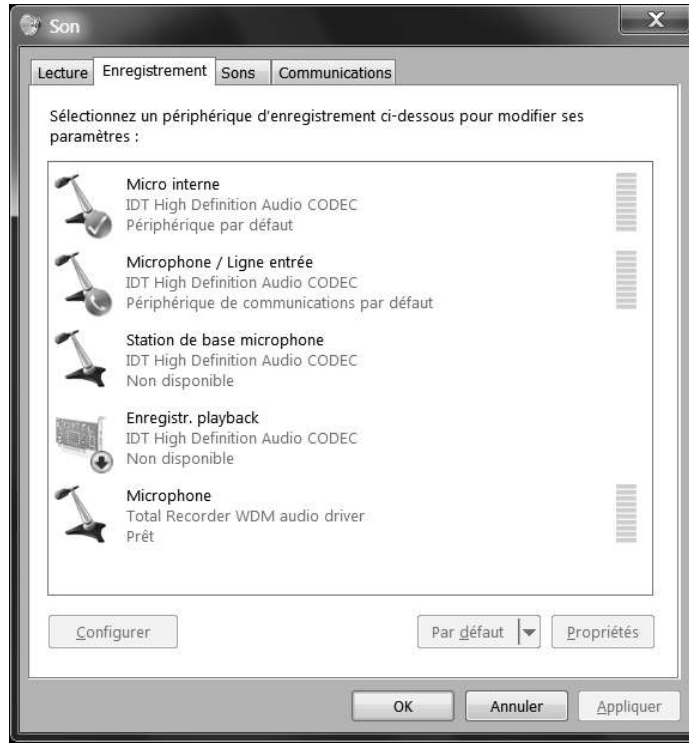


Figure 10.2 — Sélection du périphérique d'enregistrement sous Windows 7.

Quand vous êtes prêt à démarrer l'enregistrement, cliquez sur le point rouge et commencez à parler. Au fur et à mesure que vous parlez, vous allez voir une barre verte qui apparaît à côté du chronomètre. Si cette barre ne se déplace pas quand vous parlez, cela signifie que votre microphone ne fonctionne pas correctement. Dans le

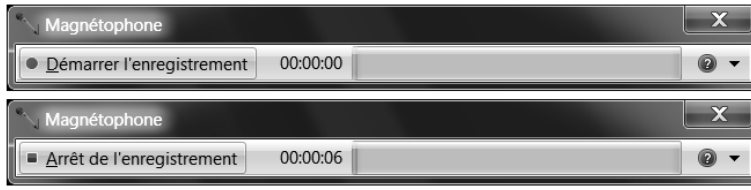


Figure 10.3 — Application Magnétophone sous Windows 7.

cas contraire, vous allez voir la barre verte se déplacer au fur et à mesure que vous parlez. Quand vous avez terminé, cliquez sur le bouton Arrêt de l'enregistrement (un carré bleu ; dans l'application Magnétophone sous Windows XP, le bouton Arrêt de l'enregistrement est un rectangle noir juste à côté du point rouge qui démarre l'enregistrement).

Quand on clique sur le bouton Arrêt de l'enregistrement, une nouvelle fenêtre Enregistrer sous s'ouvre et vous pouvez sélectionner le répertoire où vous voulez sauvegarder votre enregistrement audio. Comme nous l'avons déjà mentionné, Windows 7 ne permet d'enregistrer qu'au format `.wma` (Windows Media Audio). Si vous utilisez la version Windows XP, sélectionnez Enregistrer ou Enregistrer sous pour ouvrir une boîte de dialogue et sauvegarder le fichier au format `.wav` (et pas `.wma` !).

Conversion des fichiers

Si vous utilisez des fichiers audio pour des publics variés, vous allez avoir besoin d'une de ces deux applications :

Un éditeur de sons qui enregistre les fichiers audio dans les formats `.wav`, `.mp3` ou `.ogg`.

Un programme de conversion. Une simple recherche sur le Web vous en offrira plusieurs. Par exemple, si vous utilisez le Magnétophone de Windows 7, vous aurez besoin d'un programme pour convertir le format `.wma` en `.mp3`, `.wav` ou `.ogg`. En général, le processus est assez simple, qu'il s'agisse d'un Mac ou d'un PC.

Une grande variété de programmes de conversion est disponible, mais on peut en trouver pour Windows 7 à http://software-download.name/audio-converter-windows-7/. Pour Mac, j'ai testé Switch Sound File Converter (http://download.cnet.com/Switch-Audio-Converter/3000-2140_4-10703967.html) et je l'ai trouvé facile à utiliser ; il convertit les fichiers sonores Mac classiques (comme les fichiers `.aiff`) en fichiers audio reconnus par les navigateurs HTML5. Effectuez une recherche sur le Web et vous trouverez beaucoup d'autres convertisseurs pour Windows et Mac.

10.5 EFFETS SONORES

Il y a tellement d'effets sonores sur le Web, qu'ils soient gratuits ou payants, que vous trouverez toujours l'effet sonore que vous recherchez. Le meilleur endroit pour démarrer sa quête est FlashKit (`www.flashkit.com/soundfx`). Même si ce site est

consacré à Flash, il met à votre disposition plus de 7 000 effets sonores gratuits et libres de droits. Vous pouvez de plus les télécharger au format `.wav` ou `.mp3`, si bien qu'ils sont prêts à être utilisés sur une page Web HTML5. Si vous effectuez une recherche sur le Web, vous pouvez trouver pratiquement n'importe quel effet sonore.

Si vous voulez enregistrer vos propres effets sonores, vous pouvez utiliser de simples bruits domestiques et l'application d'enregistrement de votre ordinateur. Par exemple, un chien qui aboie, le bruit d'un avion, ou n'importe quel autre son que vous pouvez enregistrer (soyez cependant prudent avec la musique qui n'est pas libre de droits).

10.5.1 Sons de transition

Un son interactif subtil, mais efficace, peut être utilisé pour ajouter un composant audio aux transitions de pages. Dans un monde tactile de boutons, d'interrupteurs et de poignées de porte, nos actions impliquent souvent des sons. Vous pouvez faire la même chose avec vos liens Web. Accomplissez les étapes suivantes pour créer une transition simple :

1. **Allez à** `www.flashkit.com/soundfx`**.**
2. **Sélectionnez Sound FX dans le menu de la page d'accueil.**
3. **Sélectionnez Interfaces > Clicks à partir de la catégorie Interfaces.**
4. **Sélectionnez un son de clic qui vous plaise.**
5. Vous pouvez choisir l'effet qui vous plaît, mais assurez-vous simplement que sa durée est courte.
6. **Téléchargez les versions** `.wav` **et** `.mp3`**.**
7. Renommez les fichiers en `click.wav` et `click.mp3`**.**
8. **Placez les fichiers** `.mp3` et `.wav` dans un dossier.
9. Dans ce dossier, placez les pages suivantes (`SonTransition.html` et `SonOuverture.html` disponibles dans les fichiers du chapitre).

Page de démarrage sans son

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
a {
font-family:Verdana, Geneva, sans-serif;
color:#cc0000;
font-size:24px;
text-decoration:none;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Son de transition</title>
</head>
<body>
```

```
<a href="SonOuverture.html">Cliquez pour aller sur la page suivante</a>
</body>
</html>
```

Un son est joué quand la page s'ouvre

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
body {
font-family:Verdana, Geneva, sans-serif;
color:#cc0000;
font-size:24px;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Son à l'ouverture</title>
</head>
<body>
<audio preload autoplay>
  <source src="click.wav" >
  <source src="click.mp3" >
</audio>
Cette page clique.
</body>
</html>
```

Enregistrez les pages HTML5 dans le même dossier que les fichiers sonores. Testez les pages HTML5 avec plusieurs navigateurs. Quand on clique sur le lien, cela ouvre une page Web et la balise `<audio>` avec l'attribut `autoplay` doit jouer un son de clic juste après le chargement de la page. L'idée est que le son se joue presque en même temps que le clic, comme si c'était le clic sur le lien qui avait produit le son. Si la page met un peu trop de temps à se charger, le son de clic se produit quand la page s'affiche, ce qui n'est pas l'effet désiré.

Au moment de la rédaction de cet ouvrage, les navigateurs Opera et Firefox sur Macintosh ne fonctionnaient pas quand l'attribut `type` était ajouté à la balise `<source>`, mais cela marchait avec Safari et Chrome. Cependant, quand l'attribut `type` était omis, les pages Web fonctionnaient parfaitement avec tous les navigateurs HTML5 pour Macintosh. Lors des tests sous Windows 7, les dernières versions de Firefox et de Safari ne généraient pas de son, mais Opera et Chrome fonctionnaient (c'est la raison pour laquelle les développeurs Web vieillissent vite). HTML5 est encore jeune et de nombreuses fonctionnalités sont encore en développement, mais il est possible qu'au moment où vous lirez ce livre ces problèmes soient résolus.

10.5.2 Intégration d'effets sonores à une page Web

Il peut être difficile de déclencher des effets sonores quand vous le souhaitez si vous n'utilisez pas l'attribut `controls`. Avec HTML5, le seul moyen de déclencher un son est de placer une page dans un `iframe` et de jouer le fichier audio automatiquement.

Avec JavaScript, on dispose de solutions bien plus élégantes, mais l'utilisation d'`iframe` est fonctionnelle.

Les quatre pages HTML5 suivantes sont composées d'une page qui charge les trois autres pages dans un `iframe`. Quand chaque page se charge, elle joue un effet sonore : un aboiement, un cri et une explosion. L'utilisateur voit l'`iframe` passer à la couleur du bouton du haut-parleur sur lequel il a cliqué et entend l'effet sonore, sans qu'une ligne de JavaScript n'ait été utilisée. La figure 10.4 illustre ce que l'utilisateur voit quand il clique sur l'icône du haut-parleur bleu.

Figure 10.4 — Déclenchement de sons grâce à des liens vers un `iframe`.

Vous aurez besoin de télécharger (ou de créer) trois sons, chacun étant enregistré à la fois au format `.wav` et `.mp3`. Utilisez des effets sonores de courte durée et quand vous cliquerez sur chaque bouton, le son sera joué quand la page sera chargée dans l'`iframe`. La page chargée ne comporte rien d'autre qu'un son et, pour les besoins de la démonstration, elle a une couleur d'arrière-plan qui correspond à la couleur de l'icône du haut-parleur. Placez toutes les pages et les six fichiers sonores dans le même dossier (les fichiers suivants sont disponibles dans les fichiers du chapitre : `SonIFrame.html`, `son1.html`, `son2.html`, `son3.html`).

Une page avec un iframe appelle d'autres pages avec des effets sonores

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
h3 {
color:#cc0000;
font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Sons dans un iFrame</title>
</head>
<body>
```

```
<article>
  <header>
    <h3>Testeur de sons</h3>
    <iframe name="ifSound" width="125" height="10"></iframe>
  </header>
  <section> <a href="son1.html" target="ifSound"><img src="Redspeaker.gif"
width="40" height="40"></a> <a href="son2.html" target="ifSound"><img
src="Greenspeaker.gif" width="40" height="40"></a> <a href="son3.html"
target="ifSound"><img src="Bluespeaker.gif" width="40" height="40"> </a>
</section>
</article>
</body>
</html>
```

Page avec un aboiement et un fond rouge

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
body {
background-color:#cc0000;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Son 1 : Rouge</title>
</head>
<body>
<audio autoplay>
  <source src="dog.wav" >
  <source src="dog.mp3" >
</audio>
</body>
</html>
```

Page avec un cri et un fond vert

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
body {
background-color:#060;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Son 2 : Vert</title>
</head>
<body>
<audio autoplay>
  <source src="scream.wav" >
  <source src="scream.mp3" >
</audio>
</body>
</html>
```

Page avec une explosion et un fond bleu

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
body {
background-color:#0000cc;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Son 3 : Bleu</title>
</head>
<body>
<audio autoplay>
  <source src="boom.wav" >
  <source src="boom.mp3" >
</audio>
</body>
</html>
```

Assurez-vous de tester ces pages sur différents navigateurs HTML5. Essayez aussi de créer vos propres sons (vous pouvez embaucher votre chien, votre chat et un perroquet).

10.6 À VOUS DE JOUER !

L'exercice consiste à créer une bande dessinée parlante. Pensez à une histoire simple qui peut être racontée en quatre cases. Chaque case comportera un dessin (ou une photo numérique), mais pas de texte. Quand l'utilisateur clique sur chaque case, un enregistrement audio « lit » le texte que l'on pourrait trouver dans une bande dessinée. Vous devez utiliser un `iframe` pour déclencher chacun des quatre enregistrements, et chaque case sera un bouton qui reliera la page au fichier audio correspondant. Vous pouvez utiliser des images clip art pour les cases si vous le souhaitez et vous pouvez améliorer l'histoire avec des effets sonores qui accompagneront le texte audio qui est lu.

11

Vidéo

Objectif

La vidéo est l'une des fonctionnalités les plus importantes qui ont été rajoutées à HTML5. Si vous avez déjà fréquenté YouTube, vous êtes conscient de la puissance qu'exerce la vidéo sur le Web. De la même manière, cela fait des années que les utilisateurs d'Adobe Flash ont incorporé des vidéos dans leurs programmes, si bien que la vidéo sur le Web n'est pas vraiment une nouveauté. Pourtant, les nouvelles fonctionnalités de HTML5 permettent d'accéder à une vidéo directement à partir d'une page Web, et c'est quelque chose que HTML n'avait pas été capable de faire dans ses précédentes versions sans un lien vers un fichier Flash (`.swf`) ou tout autre fichier binaire qui diffusait la vidéo indépendamment des balises placées dans le fichier HTML.

Il faut cependant apporter ici une précision importante : la vidéo que vous affichez sur votre page Web n'est pas véritablement une vidéo diffusée en *streaming* ; il s'agit en fait d'un type de téléchargement progressif. Au fur et à mesure que la vidéo est téléchargée à partir du serveur Web, elle est affichée par la page Web, si bien que le débit n'est pas garanti. En fait, la plupart des vidéos créées par les adeptes de Flash appartiennent très probablement à ce type de vidéo. Pour le moment, la diffusion de la vidéo en streaming nécessite un serveur de vidéo streaming tel qu'Adobe Flash Media Server. On peut cependant s'attendre à des développements de véritable streaming au fur et à mesure que la vidéo HTML5 va gagner en popularité.

Si vous avez lu le chapitre 10, vous retrouverez certaines balises que nous avons déjà étudiées, comme la balise `<source>`, mais avec une orientation sur le chargement et la lecture de vidéos.

11.1 CRÉATION D'UNE PAGE HTML5 AVEC DE LA VIDÉO

Pour commencer à travailler sur la vidéo, il vous faut un fichier vidéo. Vous pouvez en créer un sur votre ordinateur ou bien en télécharger un sur le Web. Toute la question est de savoir le type de fichier vidéo qui convient. Il est plus facile de s'y retrouver dans la Tour de Babel que dans tous les codecs vidéo, si bien que cette section va commencer par le plus courant de tous les formats vidéo actuels, H.264. En tant que format vidéo, on désigne habituellement H.264 sous le nom de MPEG-4 (l'extension de fichier est `.mp4`). Ce format vidéo a gagné en popularité car il s'agit du premier format vidéo en haute définition pour le Web. La plupart des gens l'ont d'abord rencontré sur le Web sous la forme de fichiers Flash (`.f4v`), et les résultats étaient bien meilleurs que les précédentes vidéos pour le Web.

Sans surprise, la balise principale qui est utilisée pour la vidéo se nomme `<video>`. Tout comme pour une image ou un fichier audio, le premier attribut dont vous avez besoin est une source et l'attribut `src` est utilisé pour identifier la source. La création de pages Web intégrant de la vidéo est donc très simple. Le listing suivant (`VideoSimple.html` disponible dans les fichiers du chapitre) affiche une vidéo de base dans une page Web HTML5.

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Vidéo simple</title>
</head>
<body>
<video src="mbAux1small.mp4" controls preload="auto"></video>
</body>
</html>
```

Pour tester ce fichier, vous aurez besoin du navigateur Safari sur Macintosh car pendant l'écriture de cet ouvrage, c'était le seul navigateur qui fonctionnait avec ce type de vidéo.

Si vous exécutez le programme sous Safari, vous devez attendre de voir apparaître une image avant de lancer la vidéo. Quand l'image apparaît, cela signifie que la vidéo est prête et vous pouvez cliquer sur la flèche pour jouer la vidéo. La figure 11.1 illustre la vidéo en pause.

Bien évidemment, il est souhaitable que votre vidéo puisse être lue sur plus d'un navigateur car sinon vous allez perdre beaucoup d'utilisateurs. Heureusement, HTML5 a un moyen simple de résoudre ce problème. Au sein d'un conteneur `<video>`, vous pouvez ajouter autant de balises `<source>` que vous le souhaitez. L'attribut source (`src`) est déplacé dans la balise `<source>`. Si vous placez plusieurs balises `<source>` dans un conteneur `<video>`, le navigateur va examiner les fichiers vidéo et sélectionner celui qu'il sait jouer puis le jouer automatiquement. S'il sait jouer plus d'un type de format vidéo, le navigateur commence à jouer le premier qu'il reconnaît et ignore tout le reste. Tout ceci peut être réalisé en HTML5 sans avoir recours à JavaScript. L'extrait

de code suivant montre la syntaxe de base pour accéder aux fichiers vidéo de cette manière :

```
<video>
  <source src="someVid.3gp">
  <source src="someVid.mp4">
  <source src="someVid.ogv">
  <source src="someVid.webm">
</video>
```

Figure 11.1 — Affichage d'une vidéo simple dans le navigateur Safari.

Bien qu'il existe de nombreux formats vidéo différents, nous nous concentrerons dans ce chapitre sur l'utilisation des formats suivants :

- **H.264** : `.mp4` et `.mov`
- **OGG** : `.ogv`
- **WebM** : `.webm`
- **3GP** : `.3gp`

Pendant la rédaction de cet ouvrage (et sans doute à l'avenir) d'autres formats deviendront compatibles avec différents navigateurs, mais pour le moment, grâce à la balise `<source>`, vous pouvez facilement référencer plusieurs navigateurs différents.

Par exemple, le code suivant (`SimpleVideoSource.html` disponible dans les fichiers du chapitre) joue le même fichier vidéo sur tous les navigateurs testés, y compris deux navigateurs mobiles.

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Vidéo sélective</title>
</head>
<body>
<video controls preload="auto">
  <source src="multiformats/mbAux1.3gp">
  <source src="multiformats/mbAux1small.mp4">
  <source src="multiformats/setAux.ogv">
  <source src="multiformats/mbAux1small.webm">
</video>
</body>
</html>
```

Quand j'ai testé ce programme avec différents navigateurs et sous plusieurs plateformes, tous les navigateurs ont été capables de trouver un format vidéo compatible et de jouer à la fois la vidéo et le son. La figure 11.2 illustre la vidéo lue dans le navigateur Safari mobile sur un iPhone.

Figure 11.2 — Vidéo lue sur un iPhone.

La qualité de la lecture était assez constante sur tous les navigateurs. Sur les navigateurs Safari et Perfect Browser pour iPhone, la qualité de la vidéo était plutôt bonne et elle s'est surtout chargée rapidement.

11.2 COMPATIBILITÉ VIDÉO DES NAVIGATEURS

Deux problèmes très différents doivent être traités quand on parle de la compatibilité vidéo avec HTML5. Le premier est tout simplement de savoir quel navigateur fonctionne avec quel format vidéo. J'utilise le terme format vidéo pour désigner une combinaison de conteneurs vidéo (l'emballage dans lequel la vidéo est en fait incluse) et de codecs (technologie de codage et de décodage) à laquelle on se réfère principalement par l'extension qui est associée à ces fichiers. Techniquement, les choses sont plus complexes et il nous faudrait beaucoup plus de place pour être exhaustif sur le sujet, mais pour exploiter la vidéo, vous avez besoin de reconnaître les différents fichiers par leur extension et de savoir avec quel navigateur ils sont compatibles.

Laissons les experts faire leur travail

Les technologues semblent passer leur temps à chercher à savoir pourquoi les différentes sociétés ont choisi tel format particulier de vidéo. Les entreprises comme Apple, Microsoft, Google, Opera, Adobe et Mozilla ont choisi les formats de fichier pour des raisons de propriété intellectuelle, de droit d'utilisation, de licence, et pour des motifs financiers, mais aussi parce que l'intégration de cette technologie faisait partie d'autres projets. La seule chose qui doit vous intéresser est de savoir ce qui fonctionne pour vos sites Web et comment vous allez l'implémenter. Laissez les questions métaphysiques aux experts !

Pour le moment, il est risqué d'aller au-delà de ce qui a été testé et approuvé. J'estime cependant que nous pouvons examiner quatre types différents de conteneurs de fichiers et de codecs et utiliser les quatre formats que nous avons listés au début du chapitre. Le format de conteneur 3GP est lié au format MPEG-4, mais il s'agit en fait d'un format H.263, et il a été principalement adopté pour les périphériques mobiles comme l'iPhone. Le tableau 11.1 liste la matrice de compatibilité des principaux navigateurs avec différents formats vidéo.

À la lecture de ce tableau de compatibilité, il est évident qu'il faut apprendre à convertir les différents formats, ce qui sera le sujet de la prochaine section. La conversion doit d'abord intervenir entre le type de fichier utilisé par le système d'enregistrement (une caméra ou une application de partage d'écran) ou le logiciel d'édition vidéo. Le second type de conversion intervient entre la vidéo totalement préparée pour le Web et les types de fichiers possibles pour les pages HTML5. Une fois que tous les types de fichiers nécessaires sont prêts, il ne vous reste plus qu'à les placer dans des balises `<source>` d'un conteneur `<video>`.

Tableau 11.1 — Navigateurs et formats vidéo pris en charge

Navigateur	H.264	OGG	WebM	3GP
Chrome	Non	Oui	Oui	Non
Firefox	Non	Oui	Inconnu	Non
Internet Explorer 9	Non	Non	Non	Oui
Opera	Non	Oui	Oui	Non
Safari	Oui	Non	Non	Oui
Safari Mobile	Non	Non	Non	Oui

11.2.1 Conversion de fichiers WebM avec Miro Video Converter

De tous les formats de fichier testés, seul Opera fonctionne avec le format WebM. Cependant, plusieurs autres entreprises qui développent des navigateurs sont aussi impliquées dans le projet WebM, et il se pourrait qu'à l'avenir ce format devienne plus important qu'il ne l'est actuellement. Vous trouverez de plus amples informations sur WebM en consultant le site Web du projet WebM à l'adresse `www.webmproject.org`.

Miro Video Converter est un programme de conversion qui fonctionne avec le format WebM. Il est simple à utiliser et fournit de nombreuses options de conversion, et pas seulement pour le format WebM. La figure 11.3 illustre la conversion d'un fichier MP4 en fichier WebM dans Miro Video Converter.

Miro Video Converter est disponible gratuitement à `www.mirovideoconverter.com`. Le processus de conversion commence par le chargement du fichier à convertir dans une fenêtre centrale, puis il suffit ensuite de cliquer sur le bouton Convert. C'est très simple et intuitif.

Pour les fichiers `.ogv`, sélectionnez Theora dans le menu puis cliquez sur le bouton Convert. Le fichier converti a l'extension `.theora.ogv`, mais en supprimant `.theora`, vous pouvez parfaitement le lire uniquement avec l'extension `.ogv`. En convertissant un fichier `.mp4` en fichier `.ogv`, la taille est passée de 54 Mo à 11 Mo !

11.2.2 Conversion au format 3GP avec Adobe Media Encoder CS5

Adobe Media Encoder CS5 (AME) offre de nombreux avantages pour la conversion des fichiers au format 3GP qui est destiné aux périphériques mobiles. L'encodeur est livré avec différents produits Adobe, et pour cet ouvrage, je l'ai testé avec Adobe Premier en éditant des fichiers MP4 générés par une caméra numérique HD (haute définition).

Outre la possibilité de convertir des fichiers au format 3GP, AME permet également de réaliser certains travaux d'édition de base. La fonction la plus importante est la réduction des dimensions de la vidéo et, par conséquent, de la taille du fichier et de

Figure 11.3 — Conversion de fichiers à l'aide de Miro Video Converter.

la quantité de temps nécessaire à la diffusion sur Internet. Cela est particulièrement crucial pour les périphériques mobiles.

La figure 11.4 illustre un fichier qui a été sauvegardé nativement en 720 x 480, puis a été ensuite réduit en 320 x 212. Typiquement, les vidéos sont formatées dans un ratio 4:3, mais le format HD de la caméra utilisait un ratio 16:9, si bien que les dimensions étaient plus larges que ce qui est prévu pour une vidéo créée avec une webcam intégrée à un ordinateur. Quand on prépare une vidéo pour le Web, c'est un problème important qu'il faut prendre en compte. De la même manière, quand on définit les attributs `width` et `height` d'une balise `<video>`, il ne faut pas oublier de modifier les dimensions.

Comme vous pouvez le voir à la figure 11.4, AME fournit de nombreuses informations sur le fichier. Dans le volet de gauche, il affiche le fichier sur lequel on travaille.

Quand la conversion est terminée, AME offre différents formats d'affichage générique. Par exemple, la figure 11.5 illustre le résultat attendu sur un périphérique mobile avec un affichage horizontal.

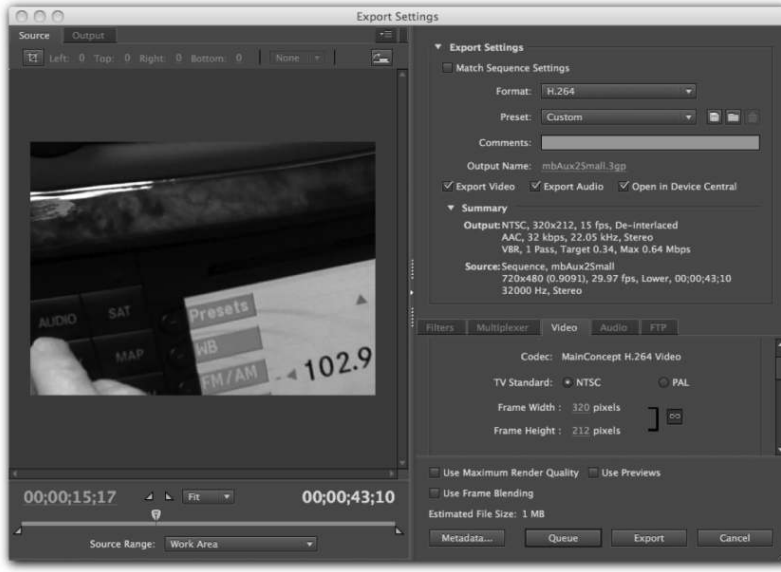


Figure 11.4 — Conversion de fichiers avec Adobe Media Encoder.

Figure 11.5 — Affichage vidéo dans Adobe Device Central.

En examinant la figure 11.5, on a une excellente idée de l'aspect de la vidéo sur le périphérique cible. Adobe Device Central offre plusieurs affichages différents de telle sorte que l'on peut optimiser la vidéo avant de la placer sur le Web.

11.3 CRÉATION DE VIDÉOS POUR LE WEB

Avant d'étudier les nombreux attributs de l'élément `video`, cette section traite le problème global de la création des vidéos et de leur enregistrement sur l'ordinateur. Si l'étendue des types vidéo disponibles pour le Web est large, la création et le stockage des vidéos regroupent aussi de nombreux aspects. Nous nous contenterons dans le cadre de cet ouvrage d'aborder les quatre moyens suivants de produire de la vidéo :

- Webcams
- Petits caméscopes
- Caméscopes standard
- Logiciel de capture vidéo

Nous allons étudier comment récupérer la production de la caméra pour la transformer en un fichier qui puisse être utilisé immédiatement ou converti en un fichier compatible avec HTML5.

11.3.1 Webcams

Aujourd'hui, la plupart des ordinateurs portables sont livrés avec une webcam intégrée, et il en va de même de nombreux ordinateurs de bureau. Pour les ordinateurs qui n'en sont pas pourvus, il existe de nombreux modèles que l'on peut facilement connecter via le port USB.

Pour les utilisateurs de Windows 7, le meilleur logiciel pour réaliser des vidéos à l'aide de la webcam est l'utilitaire qui est livré par le fabricant de la webcam. Par exemple, Logitech et Creative, deux sociétés qui fabriquent des webcams, livrent d'excellents logiciels qui enregistrent et stockent les fichiers vidéo qui peuvent être convertis pour un usage sur le Web. Vous pouvez aussi ajouter des effets spéciaux aux vidéos avec ces logiciels.

Avec Windows 7 et Windows Vista, vous pouvez aussi télécharger gratuitement la nouvelle version de Microsoft Movie Maker à l'adresse : `http://explore.live.com/windows-live-movie-maker`. À la différence de Windows XP, qui est livré avec Windows Live Movie Maker, vous devez télécharger le logiciel d'édition de films sur le site de Microsoft si vous avez Windows 7 ou Windows Vista.

Les ordinateurs Apple Macintosh sont en général livrés avec des webcams iSight. Les ordinateurs portables iMac et MacBook ont des webcams intégrées au sommet de l'écran. Les modèles qui ne sont pas livrés avec des webcams peuvent brancher une webcam iSight sur le port USB ou Firewire.

Pour la création des vidéos, on peut utiliser l'application Photo Booth qui est livrée avec les Mac. Tous les fichiers pris avec Photo Booth sont enregistrés au format QuickTime avec l'extension `.mov`. Ils sont en fait au format MP4, et si vous modifiez l'extension de `.mov` en `.mp4`, ils sont également reconnus comme des fichiers vidéo.

Les webcams sont utiles pour certains types de projets vidéo. Pour les vidéos éducatives pour le Web, l'enseignant peut s'asseoir face à la webcam et faire son cours pour ensuite diffuser la vidéo à son public. La réalisation de pages Web en HTML5 prenant en charge une présentation vidéo rend la création de didacticiels aussi simple que la réalisation de toute présentation similaire n'ayant pas été conçue pour un usage sur le Web.

11.3.2 Petits caméscopes

Le principal inconvénient des webcams pour réaliser des vidéos qui puissent être incorporées en HTML5 est qu'elles sont reliées à un ordinateur, qu'elles soient intégrées ou bien reliées par un port USB ou Firewire. Cela rend l'utilisation mobile des webcams problématique, même pour des ordinateurs portables extrêmement légers.

Il existe des webcams sans fil, mais elles ont un rayon d'action limité et sont plus onéreuses. Plusieurs solutions alternatives que l'on peut emporter partout sont pourtant disponibles. Les plus courantes sont les caméras qui sont intégrées aux téléphones mobiles. Les téléphones mobiles employés pendant les manifestations qui ont suivi les élections iraniennes en 2009 ont fourni au monde entier des témoignages sur les représailles du gouvernement contre ceux qui dénonçaient les fraudes électorales. Comme les journalistes occidentaux étaient empêchés de couvrir les événements à la suite des élections, l'information a été produite par des téléphones mobiles dont les vidéos ont été diffusées sur YouTube et annoncées via Twitter.

Il existe aujourd'hui une nouvelle génération de petits caméscopes HD qui sont parfaitement portables et qui enregistrent la vidéo sur une mémoire flash. Par exemple, la caméro Flip Mino HD avec ses dimensions de 10 cm x 5 cm x 1,60 cm (H x L x P) est plus petite que de nombreux téléphones mobiles. La figure 11.6 présente une caméra Flip typique personnalisée avec un logo d'entreprise.

Outre Flip, il existe des caméras HD fabriquées par Kodak (Pocket Video). Les caméras Flip et Kodak enregistrent sur de la mémoire flash et il n'est donc pas nécessaire d'avoir une cassette numérique ou une carte mémoire flash externe. Ces caméras sont livrées avec des logiciels d'édition vidéo limités et enregistrent la vidéo au format H.264 sur les PC et les Macintosh.

La qualité de la vidéo est aussi élevée que celle de caméscopes plus grands, plus lourds et plus chers. Ces caméras ont été conçues dès le départ pour être utilisées pour la création de vidéos pour les sites de réseaux sociaux comme Facebook et YouTube ; cela a pour conséquence que le format de sortie des fichiers natifs est prévu pour être affiché avec des éléments vidéo HTML5.

11.3.3 Caméscopes standard

Par standard, on entend ici les caméscopes portatifs avec des caractéristiques comme un objectif capable de zoomer, le stockage sur des cassettes numériques, la prise en charge de cartes mémoire flash et d'autres fonctionnalités qui peuvent être embarquées sur des plateformes plus grandes. La gamme des caméscopes s'est élargie

Figure 11.6 — Les petits caméscopes HD sont adaptés pour le Web.

à tel point qu'elle s'étend des matériels peu chers utilisés pour des enregistrements en famille jusqu'aux caméras professionnelles qui sont employées par des réalisateurs indépendants.

Comme pour les petits caméscopes, les caméscopes standard ont des connecteurs USB ou Firewire. Ces connecteurs peuvent être reliés directement aux logiciels d'édition vidéo comme Adobe Premier, Apple Final Cut ou Vegas. La vidéo éditée peut ensuite être enregistrée dans un format qui peut être utilisé par les navigateurs HTML5.

11.3.4 Logiciels de capture vidéo

Les logiciels de capture vidéo considèrent votre écran comme un enregistreur vidéo et le microphone qui est relié à votre ordinateur comme un microphone d'enregistreur vidéo. Camtasia est, par exemple, un logiciel de capture vidéo extrêmement répandu. Il est facile à utiliser et possède plusieurs fonctionnalités pour zoomer, faire des panoramiques et plus généralement pour simuler l'usage d'un caméscope braqué sur votre écran. La figure 11.7 illustre les contrôles de base du logiciel.

Fondamentalement, Camtasia nécessite simplement que l'utilisateur sélectionne l'écran et le microphone, puis clique sur le bouton Rec (voir la figure 11.7). Disponible pour Windows 7 et Macintosh OS X, c'est un logiciel qui est largement utilisé par les formateurs et les enseignants qui ont besoin que les utilisateurs suivent ce qui se passent à l'écran.

Apple Quick-Time Player intègre aussi une application de capture vidéo qui enregistre automatiquement les fichiers au format `.mov` (`.mp4`), ce qui les rend prêts à être utilisés sur un site Web HTML5. Le processus d'enregistrement est extrêmement

Figure 11.7 — Les logiciels de capture vidéo permettent d'enregistrer en direct tout ce qui se passe sur votre écran.

facile et une fois que l'on a sélectionné le microphone, l'opération se déroule en une seule étape.

11.4 ATTRIBUTS DES BALISES VIDEO ET SOURCE

Plusieurs attributs des balises `<video>` et `<source>` sont essentiels pour assurer le succès du déploiement de la vidéo en HTML5. Une fois que vous avez créé, édité et converti vos vidéos pour le Web, l'étape suivante consiste à les placer sur la page Web. Cette section étudie les attributs suivants de l'élément `video` :

- `src`
- `poster`
- `preload`
- `loop`
- `autoplay`
- `controls`
- `width` et `height`

Ces attributs de la balise `<video>` sont traités de concert avec la balise `<source>` car tous les navigateurs ne lisent pas les mêmes types de fichiers si bien que plusieurs sources différentes doivent être listées. La balise `<source>` permet aux navigateurs de choisir quel fichier vidéo est compatible avec leurs propres fonctions d'affichage vidéo (voir le début du chapitre).

11.4.1 Src

L'attribut `type` fait partie de la balise `<source>`. Comme nous l'avons montré au début du chapitre, l'attribut `src` est utilisé pour sélectionner un fichier vidéo à lire. Si le navigateur ne peut pas lire le fichier, il passe au fichier suivant de la liste des sources. Afin d'accélérer ce processus, l'attribut `type` permet au navigateur de savoir quel type de fichier est en attente de lecture et il contient un paramètre MIME qui dit quel codec est utilisé. Cela empêche le navigateur de charger le fichier puis de se rendre compte qu'il ne peut pas le lire. Au lieu de cela, il détermine à partir de l'attribut type si le fichier vidéo est compatible.

```
<source src="fileName.ext" type="video/type; codecs='c1, c2'">
```

L'assignation du type peut être faite avec ou sans le codec. Si vous ne connaissez pas le codec, vous pouvez laisser cette information en blanc et compter sur le type pour laisser le navigateur déterminer s'il peut lire le fichier. Si vous connaissez le codec ou les codecs, vous pouvez en placer plusieurs dans la liste d'assignation. Si vous n'êtes pas sûr, il est préférable de laisser la liste des codecs vide. Le programme suivant (`TypeVideoSource.html` disponible dans les fichiers du chapitre) illustre les assignations des types pour les quatre principales sortes de fichiers vidéo utilisables sur le Web.

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Vidéo sélective</title>
</head>
<body>
<video controls preload="auto">
  <source src="mbAux1.3gp" type="video/3gpp; codecs='mp4v.20.8'">
  <source src="mbAux1small.mp4" type="video/mp4; codecs='mp4v.20.8'">
  <source src="mbAux1small.ogv" type="video/ogg; codecs='theora, vorbis'" >
  <source src="mbAux1small.webm" type="video/webm; codecs='vorbis,vp8'" >
</video>
</body>
</html>
```

Pour déterminer le type et le codec d'un fichier, on trouve plusieurs programmes sur le Web. MediaInfo, qui est disponible gratuitement pour les systèmes d'exploitation Windows, Macintosh, et plusieurs versions de Linux peut être téléchargé à l'adresse `http://mediainfo.sourceforge.net/en`.

11.4.2 Poster

L'attribut `poster` est utilisé avec les grandes vidéos et les connexions Internet lentes. Il est simple à utiliser et si vous savez que votre vidéo sera longue à charger à l'écran, l'attribut poster donne à l'utilisateur quelque chose à voir pendant qu'il attend. Voici un extrait de code qui illustre la syntaxe d'utilisation :

```
<video poster="message.png">
  <source src="multiformats/mbAux1.mp4" type="video/mp4">
</video>
```

Vous noterez que l'attribut `poster` figure dans la balise `<video>` même si toutes les informations concernant le fichier se trouvent dans la balise `<source>`. Il n'y a pas de conflit entre les attributs de la balise `video` et ceux de la balise `source`.

11.4.3 Preload

Il semblerait naturel d'inclure l'attribut `preload` de la balise `<video>` sur toutes les pages Web qui utilisent de la vidéo. Ainsi, dès que la page se charge, la vidéo commence à se charger. Cela peut être important pour une page qui ne comporte qu'une seule vidéo qui est la caractéristique principale de la page. Mais s'il s'agit d'une partie secondaire de la page, ou si plusieurs vidéos sont présentes sur la page, le préchargement peut absorber toutes les ressources. Cet attribut est donc utile, mais il doit être employé judicieusement. Voici sa syntaxe d'utilisation :

```
<video preload="auto">
     <source src="mbAux1small.webm" type="video/webm; codecs='vorbis,vp8'" >
</video>
```

On peut assigner plusieurs valeurs à l'attribut `preload` et elles sont identiques aux valeurs de l'attribut `preload` pour les fichiers audio.

- `none` : cette valeur peut sembler étrange, mais certains navigateurs peuvent être paramétrés pour précharger automatiquement les fichiers vidéo. Si l'utilisation d'un fichier vidéo particulier est peu probable, le développeur peut décider de ne pas utiliser de ressources Internet et assigne donc la valeur `none` à l'attribut `preload`.
- `metadata` : tous les fichiers vidéo contiennent des métadonnées comme la durée ou d'autres données que l'auteur du contenu vidéo place dans le fichier. Quand l'utilisation d'un fichier vidéo est incertaine, il est cependant raisonnable de charger les métadonnées car cela consomme peu de ressources Internet.
- `auto` : si l'attribut `preload` est présent, il précharge automatiquement les informations du fichier vidéo. L'assignation auto agit simplement comme un rappel que le fichier va être préchargé (c'est comme si l'on n'avait assigné aucune valeur à l'attribut `preload`).

Plus votre public sera varié, plus vous aurez de contenus vidéo sur vos pages Web, ce qui rendra souhaitable l'utilisation des options de l'attribut `preload`.

11.4.4 Loop

Une boucle vidéo est quelque chose que vous devez planifier soigneusement de peur de voir fuir tous vos utilisateurs. Une boucle implique que la même vidéo va recommencer depuis le début dès qu'elle se termine. Voici un exemple d'utilisation :

```
<video loop controls>
  <source src="phantom.3gp">
</video>
```

Vous noterez que dans l'extrait ci-dessus, on a inclus un attribut `controls`, ce qui permet à l'utilisateur d'arrêter la vidéo s'il le souhaite. Si vous définissez une boucle avec un autoplay et incorporez le tout sur une page, vous risquez de perdre beaucoup

d'utilisateurs. Si vous créez une publicité en boucle, ne vous attendez pas à ce que les visiteurs soient attirés par le service ou le produit dont vous faites la publicité ; ils le remarqueront, mais ce n'est pas une bonne méthode.

Il y a un certain type de boucle, qui est plus fréquent en musique qu'en vidéo, qui peut être utile. Si la vidéo est suffisamment courte et ne comporte pas de grands mouvements, une boucle peut consommer très peu de ressources et réutiliser la même vidéo qui sera stockée dans le cache. Une démonstration d'un processus ou même une publicité qui est ennuyeuse peut être utilisée de cette façon-là.

11.4.5 Autoplay

Comme l'attribut `loop`, l'attribut `autoplay` doit être utilisé à bon escient quand il est employé avec la vidéo. L'attribut `autoplay` est une combinaison de préchargement et de démarrage automatique de la vidéo. La syntaxe est un interrupteur et il suffit d'intégrer `autoplay` dans la balise `<video>` pour que la vidéo démarre.

```
<video poster="wait.jpg" autoplay>
  <source src="phantom.3gp">
</video>
```

Dans l'extrait ci-dessus, l'utilisateur n'a aucun moyen pour arrêter la lecture de la vidéo, mais s'il n'y a pas d'attribut `loop`, la vidéo ne sera lue qu'une fois et s'arrêtera. Si la page est censée n'être rien d'autre qu'une vidéo, il est assez prudent d'utiliser `autoplay` sans contrôleur. Vous noterez aussi que l'extrait de code comporte un attribut `poster` pour avertir l'utilisateur qu'il va se passer quelque chose, au cas où la vidéo serait longue à charger. Dans le contexte d'un site Web utilisant `autoplay`, assurez-vous d'inclure un lien vers la page suivante au cas où l'utilisateur ne souhaite pas voir la vidéo plus d'une fois.

11.4.6 Controls

L'attribut `controls` génère un panneau de contrôle graphique sous la vidéo qui permet à l'utilisateur d'accomplir les fonctions suivantes :

- Démarrer la vidéo,
- Arrêter la vidéo,
- Couper le son de la vidéo,
- Contrôler le volume de la vidéo,
- Indiquer la position,
- Avancer ou reculer.

L'attribut `controls` est un interrupteur qui est implémenté de la manière suivante :

```
<video controls>
  <source src="multiformats/mbAux1small.webm">
</video>
```

L'implémentation du contrôle diffère d'un navigateur à l'autre (comme pour le contrôle audio). La figure 11.8 illustre l'affichage de la même vidéo dans les navigateurs Opera et Chrome.

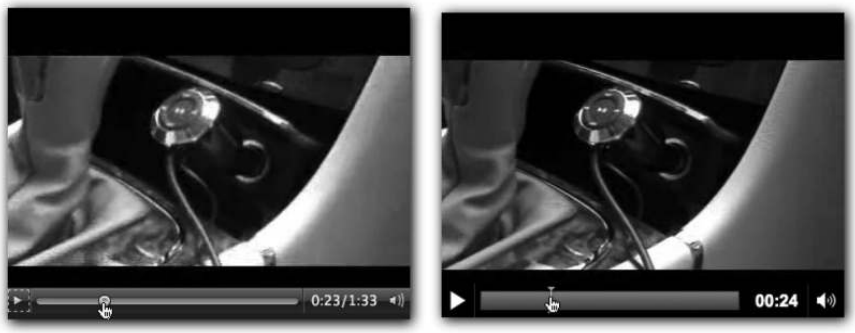


Figure 11.8 — Navigateurs Opera (gauche) et Chrome (droite) affichant un contrôle vidéo.

Les différences dans les contrôles relèvent principalement du style, mais comme vous pouvez le voir dans la comparaison des navigateurs Opera et Chrome, Opera affiche la position de la vidéo en cours de lecture par rapport à la durée totale, alors que Chrome n'affiche que la position en cours de la vidéo.

11.4.7 Width et Height

Les attributs `width` et `height` sont très importants pour la vidéo. Les navigateurs utilisent les valeurs `width` et `height` comme des indices pour afficher la vidéo. Plus les valeurs seront proches des valeurs de la taille réelle de la vidéo, meilleure sera sa lecture. L'extrait de code suivant indique la syntaxe :

```
<video width="352" height="288">
  <source src="multiformats/mbAux1small.ogv">
</video>
```

La plupart des vidéos conservent un ratio 4:3, comme la résolution 320 x 240 ; cependant, avec la vidéo HD, le ratio est différent et l'édition modifie parfois les dimensions d'une vidéo. Vous pouvez sélectionner un fichier vidéo et regarder ses propriétés, mais on ne vous donne pas toujours ses dimensions. Par exemple, sous Mac OS X, les informations sur les dimensions des fichiers `.ogv` et `.webm` ne sont pas fournies quand on demande l'affichage des propriétés, alors que les dimensions d'une vidéo au format MPEG4 sont parfaitement affichées.

11.5 À VOUS DE JOUER !

Cet exercice nécessite une caméra vidéo dont la qualité importe peu. Si vous avez déjà assisté à une présentation réalisée avec Microsoft PowerPoint, vous savez que lorsque le conférencier parle, il affiche différentes diapositives contenant du texte et des images. Dans le cadre de cet exercice, pensez à quelque chose que vous aimeriez apprendre à quelqu'un. En utilisant une combinaison d'images, de vidéo et de texte, créez une présentation Web de trois pages. Quand l'utilisateur se déplace d'une page à l'autre, il y a une vidéo sur chaque page qui démarre automatiquement, mais un contrôleur est présent sur la page pour pouvoir arrêter la vidéo ou exercer d'autres fonctionnalités. Incluez une image pour illustrer le sujet et du texte pour expliquer le contenu de la présentation. Vous pouvez vous assoir face à une webcam pour réaliser la vidéo.

QUATRIÈME PARTIE

Balises HTML5 dynamiques et un soupçon de JavaScript

12

Introduction à JavaScript

Objectif

JavaScript est un langage de programmation Web que l'on peut utiliser avec HTML5. Il sert à accéder à certaines parties des pages Web écrites en HTML5 et permet d'autres choses que l'on ne peut tout simplement pas réaliser sans JavaScript. Ce chapitre présente les fonctionnalités de base qui vont être utilisées spécifiquement avec les éléments HTML5.

JavaScript est considéré comme un langage de script car il est interprété par le navigateur quand la page Web est chargée ; le script n'est donc pas compilé et stocké sur l'ordinateur en tant que fichier binaire. L'implémentation de JavaScript peut légèrement varier d'un navigateur à l'autre, mais comme JavaScript respecte la norme ECMAScript (ECMA-262), ces différences ne sont pas énormes et je ne traiterai dans ce chapitre que les aspects de JavaScript que l'on peut utiliser avec HTML5.

Vous noterez pour finir que JavaScript et Java n'ont rien en commun. JavaScript n'est pas une version interprétée de Java. Si vous voulez trouver sur le Web des informations complémentaires à propos de JavaScript, il ne vous servira donc à rien de rechercher des renseignements sur *Java*.

12.1 INSERTION DE JAVASCRIPT DANS DES PAGES HTML5

Les programmes JavaScript sont placés dans l'en-tête d'une page Web car c'est cette partie de la page qui se charge en premier, si bien qu'elle est prête quand le reste

de la page se charge. Les programmes JavaScript agissent en grande partie comme les scripts CSS3, et comme eux, ils peuvent être placés ailleurs que dans l'en-tête de la page. Pour des raisons de simplicité, je me limiterai dans ce chapitre à placer tout le code JavaScript dans l'en-tête. À titre d'exemple, testez le programme suivant (`PremierJS.html` disponible dans les fichiers du chapitre).

```
<!DOCTYPE HTML>
<html>
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <script type="text/javascript">
      document.write("Une conversation avec HTML5 va s'engager
brièvement....");
    </script>
    <title>Premier programme JavaScript</title>
    </head>
    <body>
</body>
</html>
```

Quand on teste ce programme, on voit du texte sur la page et rien d'autre. L'élément important pour comprendre la relation entre HTML5 et JavaScript se situe dans la fonction : `document.write()`. `Document` fait référence à la page Web, et `write()` est une méthode qui dit à la page Web ce qu'elle doit faire. En l'occurrence, `write()` dit au programme d'écrire le texte entre guillemets sur la page Web.

12.1.1 JavaScript dans des fichiers externes

Tout comme pour les fichiers CSS3, vous pouvez créer des programmes JavaScript dans des fichiers texte et les enregistrer. L'extension `.js` est utilisée pour identifier les fichiers JavaScript. Par exemple, le programme JavaScript suivant ne se compose que d'une seule ligne :

```
document.write("Cela provient d'un fichier externe... ");
```

Enregistrez-le sous le nom `JSexterne.js` dans un fichier au format texte. Saisissez ensuite le programme HTML5 suivant et enregistrez-le dans le même dossier que le programme `JSexterne.js`. L'élément important de la page est la balise `<script>` qui est utilisée pour spécifier le programme JavaScript à utiliser.

```
<!DOCTYPE HTML>
<html>
<head>
<script src="JSexterne.js"></script>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JavaScript externe</title>
</head>
<body>
</body>
```

Quand la page Web s'ouvre, on voit le contenu de l'instruction `document.write()`. La méthode `write()` est une fonction intégrée qui attend une ligne de texte qu'elle affiche à l'écran. Dans ce cas, le texte provient d'un fichier externe, mais on aurait pu tout aussi bien incorporer le texte dans le script d'une page Web.

12.1.2 Fonctions

Les fonctions JavaScript sont des packages de code qui sont exécutés quand ils sont appelés par la page Web. L'avantage des fonctions est que l'on peut les utiliser pour rassembler du code et effectuer des modifications pour ajouter un nouveau contenu. La fonction intégrée `write()` requiert seulement que l'on saisisse du texte pour qu'elle l'affiche dans le document (la page Web). Vous n'êtes pas obligé d'exploiter les fonctions intégrées et vous pouvez parfaitement créer les vôtres. Par exemple, le programme suivant est un fichier JavaScript externe avec une fonction simple qui ouvre une boîte de dialogue `alert()` (il s'agit d'une fonction utilisateur qui emploie une fonction intégrée). Enregistrez le programme JavaScript suivant sous le nom de `fonction.js` :

```
// Document JavaScript
var prenom="Bill";
function bonjour(nom)
{
alert("Bonjour " + nom);
}
bonjour(prenom)
```

Toutes les fonctions sont suivies de parenthèses. Si cela est nécessaire, le développeur peut ajouter un paramètre entre parenthèses. Dans notre exemple, le paramètre s'appelle `nom`. Quand la fonction est appelée, le développeur place un nom, un nombre ou toute autre information souhaitée dans l'espace du paramètre `nom`. Dans notre exemple, on assigne la valeur `Bill` à une variable nommée `prenom`. À la fin du programme, la ligne `bonjour(prenom)` appelle la fonction en plaçant la variable dans le paramètre. La fonction passe la valeur de la variable à la fonction `alert()` à l'intérieur de la fonction `bonjour()`, si bien que l'on peut s'attendre à voir apparaître à l'écran une boîte de dialogue au lancement du programme. Le code HTML5 suivant (`fonctionJS.html` disponible dans les fichiers du chapitre) appelle le programme JavaScript qui appelle la fonction.

```
<!DOCTYPE HTML>
<html>
<head>
<script src="fonction.js"></script>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Fonction externe</title>
</head>
<body>
</body>
</html>
```

Ce programme JavaScript s'exécute dès que la page se charge. La section suivante illustre un usage plus élaboré des fonctions JavaScript en raison de leur capacité à attendre qu'on les appelle en cas de besoin.

12.1.3 Gestionnaires d'événements

La réelle puissance de JavaScript couplé à HTML5 se révèle mieux quand le programme attend que l'utilisateur fasse quelque chose pour exécuter un script. Par exemple, si l'utilisateur clique sur quelque chose, vous pouvez lancer n'importe quel programme JavaScript. On utilise pour cela un gestionnaire d'événements HTML5. La page détecte un type d'action (un événement) et possède une fonction intégrée qui reconnaît cet événement.

HTML5 reconnaît un grand nombre d'événements. Certains se produisent automatiquement, comme quand la page se charge. D'autres événements se produisent quand l'utilisateur fait quelque chose avec la souris ou le clavier. Le tableau 12.1 liste un échantillon de quelques gestionnaires d'événements.

Tableau 12.1 — Échantillon de gestionnaires d'événements HTML5

Gestionnaires d'événements				
onchange	onclick	ondbleclick	ondrag	ondragend
ondragenter	ondragleave	ondragover	ondragstart	ondrop
onkeydown	onkeypress	onkeyup	onmousedown	onmousemove
onmouseout	onmouseover	onmouseup	onmousewheel	onpause
onplay	onplaying	onprogress	onloadstart	onload

Voici la syntaxe générique de tous les événements qui sont liés à des éléments :

```
<élément événement= "FonctionJavascript()">
```

Par exemple,

```
<body onLoad = "annonceQuelquechose()">
```

utilise l'élément `body` avec le gestionnaire d'événement `onLoad` pour déclencher une fonction JavaScript nommée `annonceQuelquechose()`.

Détection d'événements variés

Pour voir comment les gestionnaires d'événements fonctionnent avec JavaScript, le programme suivant (`DetectionClic.html` disponible dans les fichiers du chapitre) possède trois gestionnaires d'événements différents et trois fonctions JavaScript différentes qui sont lancées automatiquement par les événements. Le premier ouvre

une boîte de dialogue quand la page se charge, le second se déclenche quand on clique sur la ligne du haut et le troisième affiche une boîte de dialogue quand on fait un double-clic sur le deuxième lien.

```
<!DOCTYPE HTML>
<html>
    <head>
<style type="text/css">
h1, h2 {
font-family:Tahoma, Geneva, sans-serif;
}
a {
text-decoration:none;
color:#060;
}
</style>
<script type="text/javascript">
function detectLoaded()
{
alert("La page est chargée.");
}
function detectClick()
{
      alert("Vous avez cliqué sur un lien.");
}
function detectDoubleClick()
{
      alert("Vous avez fait un double-clic sur un lien.");
}
</script>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Event Handler</title>
    </head>
    <body onLoad="detectLoaded()">
    <hgroup>
      <h1> <a href="#" onClick="detectClick()">Cliquez sur ceci</a></h1>
      <h2> <a href="#" onDblClick="detectDoubleClick()">Faites un double-clic
sur ceci</a></h2>
    </hgroup>
</body>
</html>
```

Les fonctions JavaScript peuvent effectuer tout ce qu'on leur demande de faire, ce qui permet de communiquer avec les utilisateurs. Vous pouvez ainsi leur donner des instructions, leur proposer des options, les mettre en garde, etc.

Gestion des éléments

Dans la zone « clic » du programme précédent, la balise de lien `<a>` est utilisée pour définir le gestionnaire d'événements, à l'aide de la syntaxe suivante :

```
<a href="#" onClick="GestionnaireÉvénementClick()">
```

Ce type de code n'a rien de nouveau en HTML5. Il est utilisé ici pour une simple raison : quand la souris passe au-dessus du texte à l'intérieur de la balise `<a>`, le curseur est modifié de telle sorte que l'utilisateur sait qu'il est au-dessus d'un lien.

Cependant, vous pouvez définir un gestionnaire d'événements pour n'importe quel élément. Par exemple, examinez la page Web suivante (`ClicP.html` disponible dans les fichiers du chapitre).

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
p {
font-family:Verdana, Geneva, sans-serif;
color:#FF0;
background-color:#00F;
font-size:24px;
text-align:center;
font-weight:bold;
}
</style>
<script type="text/javascript">
function showArticle()
{
alert("Vous venez de cliquer dans un conteneur <article>");
}
function showHeader()
{
alert("Vous venez de cliquer dans un conteneur <header>");
}
function showP()
{
alert("Vous venez de cliquer dans un conteneur <P>");
}
</script>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>OnClick dans un élément</title>
</head>
<body>
<article onClick="showArticle()">
  <header onClick="showHeader()">
  <h1>Ceci est un élément H1 dans l'en-tête</h1>
  </header>
  <section>
    <p onClick="showP()">Cliquez sur ce paragraphe</p>
    Voici simplement du texte dans le conteneur article. Cliquez ici juste
pour voir ce que se passe.</section>
</article>
</body>
</html>
```

En étudiant le programme ci-dessus, vous remarquerez peut-être que certains événements sont incorporés à l'intérieur d'autres éléments qui ont également des gestionnaires d'événements. Par exemple, tous les éléments sont à l'intérieur de la balise `<article>`. Que va-t-il se passer si l'on clique sur le paragraphe qui a

un gestionnaire d'événements ? Ou bien sur l'élément `<header>` ? Vont-ils réagir à l'élément interne ou à l'élément externe ? Regardez attentivement les deux copies d'écran de la figure 12.1.

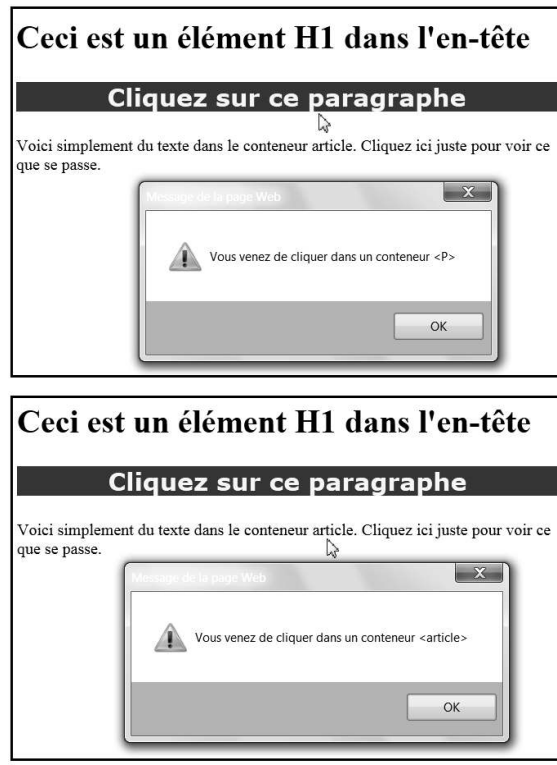


Figure 12.1 — Gestionnaires d'événements imbriqués.

Dans la copie d'écran du haut, dès que l'utilisateur clique sur la ligne « Cliquez sur ce paragraphe, l'événement est notifié dans une boîte de dialogue. Ensuite, quand l'utilisateur clique sur le bouton OK de la boîte de dialogue, une seconde boîte apparaît et lui indique qu'il a aussi cliqué sur un conteneur `<article>`. Il faut appréhender les événements en les remontant, en commençant par le niveau le plus bas de la hiérarchie des éléments puis en montant jusqu'au niveau le plus élevé.

12.2 UTILISATION DU DOM

Le DOM (Document Object Model) pour HTML5 représente une arborescence. À la base (la racine) de chaque page Web ou document se trouve la balise `<html>`, et les autres éléments de la page sont des branches de l'arbre. JavaScript utilise le DOM pour manipuler les pages Web et aller au-delà des possibilités qu'offre HTML5 tout seul. La totalité de l'arborescence du DOM est une représentation du document qui réside dans la mémoire de l'ordinateur.

Quand on accède à une partie du DOM, c'est en référençant un élément de l'arborescence. On accède à chaque élément de l'arborescence dans l'ordre hiérarchique, en commençant par `document`. Les différents éléments d'une page Web sont les différentes propriétés ou méthodes (fonctions intégrées) du `document` séparées par un point (`.`). Par exemple,

```
document.forms.fred;
```

accède à un formulaire nommé `fred` au sein d'un document. Voici à quoi ressemble le balisage HTML5 :

```
<form name= "fred">
```

Parfois, vous verrez une fonction intégrée qui réalise dans le document quelque chose de ce genre :

```
document.write("Cela vient directement du Document");
```

qui affiche du texte à l'écran. Vous noterez aussi que la racine `window` possède plusieurs fonctions intégrées qui sont utiles pour manipuler les zones d'affichage d'une page Web.

12.2.1 Fonctionnement du DOM avec les pages et JavaScript

Afin de mieux comprendre la manière dont le DOM fonctionne avec les pages et JavaScript, il est utile de voir ce qui peut être fait avec la fenêtre d'une page Web (la partie de votre page Web qui est affichée). Le programme suivant (`OuverturePage.html` disponible dans les fichiers du chapitre) montre comment charger une nouvelle fenêtre à partir du document en cours, tout en laissant la page actuelle en place.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
a {
text-decoration:none;
color:#cc0000;
font-size:24px;
}
header {
text-align:center;
}
</style>
<script type="text/javascript">
function someOtherWindow()
{
window.open("AutreFenetre.html","ow","width=400,height=200");
}
</script>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Ouverture d'une autre page</title>
</head>
<body>
<header> <a href="#" onClick="someOtherWindow()">Cliquez pour ouvrir une
nouvelle fenêtre</a> </header>
</body>
</html>
```

Cette page nécessite une seconde page qui s'ouvre en tant que fenêtre séparée. Le programme suivant (`AutreFenetre.html` disponible dans les fichiers du chapitre) provoque l'ouverture d'une page et, en même temps, la fermeture de la fenêtre ouverte grâce à un script basé sur le DOM.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
h1,h4 {
font-family:Verdana, Geneva, sans-serif;
color:#930;
}
a {
text-decoration:none;
color:#cc0000;
text-align:center;
}
</style>
<script type="text/javascript">
function shutItDown()
{
window.close();
}
</script>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Autre fenêtre</title>
</head>
<body>
<h1>Cette fenêtre contient un message important...</h1>
<h4>Attendez que je trouve de quoi il s'agit...</h4>
<a href="#" onClick="shutItDown()">Fermez la fenêtre !</a>
</body>
</html>
```

La figure 12.2 illustre le résultat quand la page Web ouvre une seconde fenêtre.

Jusqu'à présent dans cet ouvrage, quand une page était chaînée à une autre page, la page en cours disparaissait dès que l'utilisateur cliquait sur un lien. Grâce à ce petit bout de code JavaScript, vous pouvez désormais « parler » directement à la page et lui dire que vous voulez ouvrir une nouvelle fenêtre d'une taille bien précise, tout en laissant la fenêtre en cours ouverte.

Figure 12.2 — Ouverture d'une deuxième fenêtre.

12.2.2 Les éléments HTML5 et le DOM

Afin de mieux comprendre comment travailler avec le DOM en HTML5, je vais vous montrer que certains nouveaux éléments nécessitent des références au DOM à l'intérieur des balises elles-mêmes. Parmi ces nouveaux éléments, on peut citer la balise `<output>`. Au cours de la rédaction de cet ouvrage, Opera était le seul navigateur à avoir implémenté complètement ce nouvel élément et il faudra donc d'abord le tester avec ce navigateur. Avant de l'incorporer pleinement à votre site, testez-le avec tous les autres navigateurs car `<output>` peut se révéler très utile et devenir un élément clé de HTML5.

Quand on utilise la balise `<output>`, on peut placer les résultats d'un calcul directement sur la page Web. On n'a pas à créer de fonction JavaScript ni même de script, mais le code au sein d'une balise `<output>` doit suivre les mêmes règles DOM qu'en JavaScript. Le conteneur `output` ne nécessite pas de contenu entre les balises d'ouverture et de fermeture, mais tous les calculs doivent être effectués dans la balise `<output>` elle-même.

L'élément output fonctionne conjointement avec la balise `<form>` qui est traitée en détail au chapitre 14, mais nous nous concentrons pour l'instant sur la structure du DOM dans l'emploi de la balise `<output>`. Le script suivant (`CalculTTC.html` disponible dans les fichiers du chapitre) montre comment incorporer cet élément dans une page HTML5 fonctionnelle.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
/*042B45,FFC54F,FFE6BF,E8A5B5,FF0A03*/
body {
```

```
font-family:Verdana, Geneva, sans-serif;
background-color:#FFE6BF;
color:#042B45;
}
input {
background-color:#FFE6BF;
}
h1 {
color:#E8A5B5;
background-color:#042B45;
text-align:center;
}
h3 {
color:#FFC54F;
background-color:#FF0A03;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Simple calcul</title>
</head>
<body>
<header>
  <h1>Calculatrice TTC</h1>
</header>
<form>
  <input name=cout type=number>
   Coût HT<br>
  <input name=taxe type=number>
   Taxe--Saisissez une valeur décimale (par ex. : 0.196)<br>
  <h3>  Total = €
    <output onforminput="value = cout.valueAsNumber * taxe.valueAsNumber +
cout.valueAsNumber"></output>
  </h3>
</form>
</body>
</html>
```

La balise `<form>` ne comporte aucune information au-delà de sa propre balise. Pour cette application, elle n'a besoin de rien. Dans le conteneur `<form>`, il y a deux champs de saisie nommés `cout` et `taxe`. Dans le contexte du DOM, chaque champ est un objet doté de certaines propriétés, dont l'une est `valueAsNumber`. Quel que soit le caractère numérique présent dans le champ de saisie, il est traité comme un nombre et non pas comme du texte. `valueAsNumber` est une propriété de la balise `<input>` et c'est le type `number` qui a été utilisé dans cet exemple (nous aurions pu utiliser une valeur `text` pour le champ de saisie et obtenir le même résultat avec la balise `<output>`). Le champ de saisie `number` a un contrôle de type « spinner » (qui permet d'incrémenter ou de décrémenter), mais les valeurs du champ de saisie ne sont pas automatiquement converties en données numériques. La figure 12.3 illustre le résultat de la page Web dans un navigateur Opera (le seul navigateur HTML5 qui implémentait le gestionnaire d'événements `onFormInput` au moment de la rédaction de cet ouvrage).

Vous noterez le fonctionnement du gestionnaire d'événements `onFormInput`. Quand l'information est saisie dans le formulaire, le résultat est calculé puis affiché. Au départ, le résultat affiche NaN (Not a Number, ceci n'est pas un nombre) car le

taux de la taxe est nul, ce qui provoque un résultat non numérique. Cependant, dès que le taux de la taxe est saisi, le résultat devient un nombre.

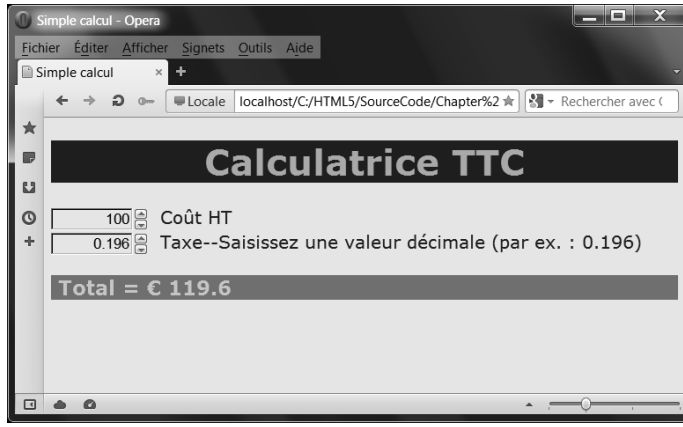


Figure 12.3 — Utilisation de la balise `<output>` pour calculer dans le navigateur Opera.

12.3 STOCKAGE DES VALEURS TEMPORAIRES

Dans ce bref aperçu de JavaScript, nous avons déjà étudié de grandes choses et vous ne devez donc pas vous inquiéter si vous n'avez pas tout compris. Le travail avec le DOM est la chose la plus importante que vous devez retenir sur l'utilisation de JavaScript dans le contexte de HTML5. Dans cette section, je vais vous montrer comment les données sont stockées temporairement dans la mémoire de l'ordinateur quand on consulte une page Web. Les utilisateurs peuvent saisir des données en cliquant sur un bouton, une case à cocher, un bouton radio ou un lien, ou bien en utilisant le clavier (tout ceci a un rapport avec ce qui se passe avec le DOM, croyez-moi !).

Afin d'utiliser les informations que l'utilisateur peut saisir, JavaScript a les moyens de les stocker en mémoire pour les réutiliser ultérieurement au cours de la session. En examinant les différentes structures de JavaScript, vous pouvez comprendre comment cela se passe.

12.3.1 Variables

Une variable est quelque chose qui change (elle varie). Vous pouvez vous représenter une variable comme une boîte avec une étiquette dessus. Par exemple, vous pouvez avoir une boîte avec une étiquette « TéléphoneMobile ». Dans la boîte, vous ne pouvez placer qu'une seule chose. Vous pouvez modifier le contenu de la boîte, ce que l'on appelle la valeur de la boîte. Si vous avez un iPhone dans votre boîte TéléphoneMobile, vous pouvez l'enlever et placer un iPhone différent (un modèle plus récent) ou bien un autre téléphone, comme un téléphone Android. À présent, la

boîte a une valeur différente. La paire étiquette-valeur (ou paire nom-valeur) est la combinaison de l'étiquette de la variable et de sa valeur actuelle.

Vous n'avez pas à placer le nom d'un téléphone mobile dans votre boîte TéléphoneMobile. Vous pouvez en fait y mettre ce que vous voulez : un talkie-walkie ou un éléphant rose. Vous pouvez même assigner n'importe quelle valeur de n'importe quel type à cette variable, y compris une autre variable. C'est cependant une bonne habitude d'utiliser des noms de variables qui peuvent être associés au contenu que l'on assigne à la variable. Par exemple, si vous réalisez un site Web qui sera utilisé pour saisir des prix et des taxes (comme c'était le cas dans la section précédente « Les éléments HTML5 et le DOM »), il paraît logique d'utiliser des noms de variables tels que « cout » et « taxe ».

Pour créer une variable, il suffit de fournir un nom et de lui assigner une valeur, comme dans l'exemple suivant :

```
billVar="Livré par la variable de Bill.";
alert(billVar);
```

qui crée une variable nommée `billVar` dont la valeur est `Livré par la variable de Bill`. Quand la variable est placée dans la fonction alert, vous noterez qu'elle n'est pas encadrée par des guillemets.

Types de données

Quand on assigne des valeurs à une variable JavaScript, on peut assigner n'importe quel type puis changer de type. Commençons d'abord par donner une idée des différents types de données qui sont disponibles. La liste suivante fournit une brève description de chacun des types :

1. String : traité comme du texte, en général entre guillemets ;
2. Number : nombre (entier ou réel) qui peut être soumis à des opérateurs mathématiques ;
3. Boolean : type de données à deux états (true ou false, 0 ou 1) ;
4. Function : ensemble d'opérations JavaScript contenues dans un module ;
5. Object : collection de propriétés encapsulées (variables/tableaux) et de méthodes (fonctions).

Vous avez déjà vu la manière dont les variables string fonctionnent. Quand on place des nombres dans une variable string, ils sont traités comme du texte et non pas comme des nombres. Par exemple, la chaîne suivant traite « 123 » exactement comme « rue Molière », c'est-à-dire du texte.

```
adresse="123 rue Molière";
```

De la même manière, si l'on utilise les assignations suivantes, on obtient toujours du texte à l'arrivée :

```
premierNombre="123";
secondNombre="7";
total=premierNombre + secondNombre;
document.write(total);
```

Au lieu d'afficher « 130 », le résultat affiche « 1237 ». Essayez ensuite le code suivant :

```
premierNombre=123;
secondNombre=7;
total=premierNombre + secondNombre;
document.write(total);
```

À présent, le résultat affiche « 130 » comme prévu quand on ajoute des nombres. Quand l'opérateur plus (+) est utilisé avec du texte (on appelle cela une concaténation), les chaînes de caractères sont simplement mises bout à bout. Si l'on place n'importe quel type de texte dans une liste de nombres à ajouter et qu'un seul des nombres soit du texte, tout le reste est considéré comme du texte et sera par conséquent concaténé.

Mélange de différents types de variables

Le programme suivant (`VariableSimple.html` disponible dans les fichiers du chapitre) utilise tous les différents types de données. Vous devez regarder attentivement les différents types de données pour déterminer les résultats attendus. Les commentaires dans le code devraient vous aider à identifier tous les types de données JavaScript.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
/*BAD9CB,048C3F,7BA651,F2BE5C,F2A950 */
body {
background-color:#BAD9CB;
font-family:Verdana, Geneva, sans-serif;
color:#048C3F;
}
</style>
<script type="text/javascript">
function advertisement()
{
billVar="Livré par la variable de Bill.";
return billVar;
}
//Variable avec une fonction
popUpAd=advertisement();
document.write(popUpAd);
//Variable avec du code HTML5
cr="<br>";
document.write(cr);
// Variable string
adresse=" rue Molière";
```

```
// Variable Boolean
var destin=true;
// Variable string
recherche="Vais-je trouver le vrai bonheur dans HTML5 ? La réponse est : ";
// Variables avec des nombres
fun=100;
maison=23;
// Calcul avec des variables
funPlusMaison=fun + maison;
// Ajout de variable string et number (concaténation)
montreAdresse=funPlusMaison + adresse;
browser=navigator.platform;
document.write(montreAdresse);
document.write(cr);
document.write(recherche);
document.write(destin);
document.write(cr);
document.write(browser);
</script>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Variable simple</title>
</head>
<body>
</body>
</html>
```

En fonction du type d'ordinateur que vous utilisez, la valeur de la variable `browser` sera différente. La page a été exécutée à la fois sur un ordinateur tournant sous Windows 7 et un Macintosh pour voir comme une variable peut varier. La figure 12.4 illustre le résultat différent du même programme.

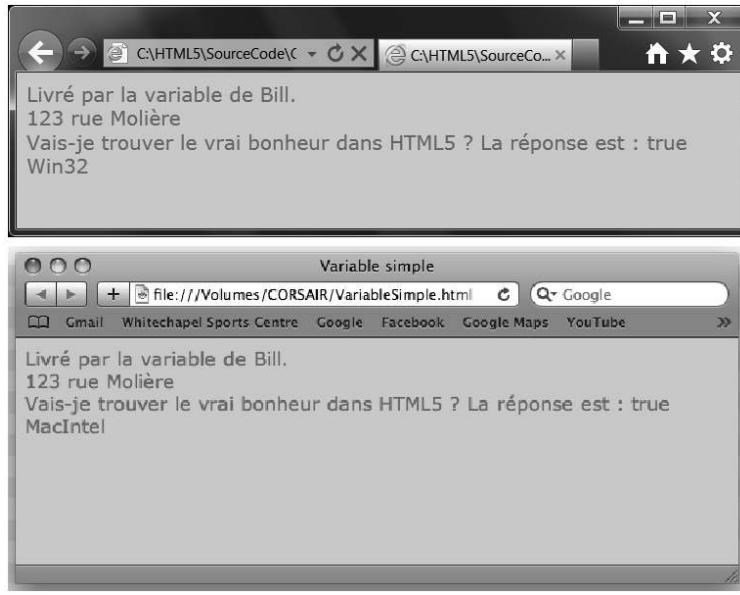


Figure 12.4 — Affichage de différents types d'ordinateur à l'écran.

La valeur de la variable `navigator.platform` est un objet. L'objet `navigator` a une propriété `platform` qui indique sur quel type d'ordinateur le navigateur s'exécute. En testant le programme sous Windows 7 (la copie d'écran du haut de la figure 12.4) avec un système d'exploitation 64-bits, le résultat affiche `Win32`. Cela est dû au fait que les navigateurs testés étaient en 32-bits, y compris une version récente d'Internet Explorer 9. Le résultat `MacIntel` (copie d'écran du bas de la figure 12.4) provient d'un ordinateur Macintosh avec un processeur Intel affiché sur un navigateur Safari.

12.3.2 Tableaux

Une variable ne peut avoir qu'une seule valeur à la fois. Cette valeur peut être le résultat d'un calcul basé sur la combinaison de plusieurs valeurs, mais une fois qu'elle est stockée à l'intérieur d'une variable, elle devient unique. Pour illustrer cela, voici un exemple extrait de la précédente section sur les variables :

```
premierNombre=123;
secondNombre=7;
total=premierNombre + secondNombre;
```

La variable nommée `total` est la somme des deux premières variables et il s'agit d'une entité unique. Cela serait également vrai si les deux variables avaient été concaténées. Vous devez donc simplement retenir que les variables ne peuvent contenir qu'une seule valeur à la fois. La figure 12.5 fournit une illustration graphique de la différence entre les variables et les tableaux.

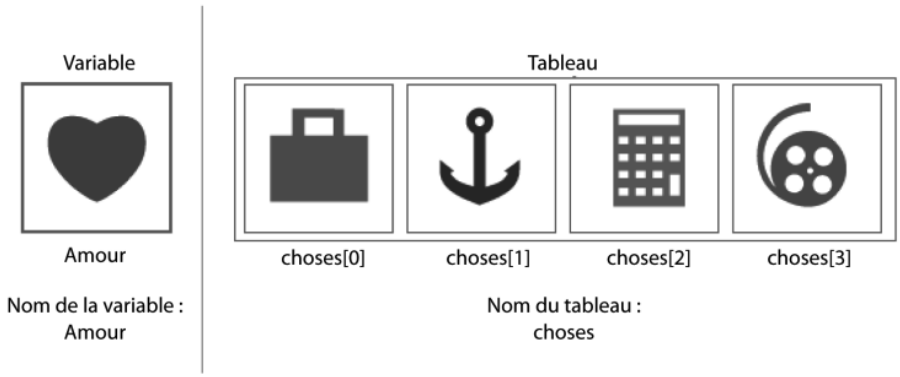


Figure 12.5 — Stockage de données dans des variables et des tableaux.

Comme vous pouvez le voir à la figure 12.5, il n'y a qu'un seul élément stocké dans la variable nommée `Amour`, mais le tableau `choses` contient beaucoup de... choses. On appelle chacune des données stockées avec le nom du tableau suivi d'un numéro entre crochets. Ainsi, `choses[1]` désigne une ancre et `choses[2]` une calculatrice.

Certaines applications nécessitent qu'il y ait plusieurs valeurs dans un seul objet, ce qui facilite l'accès et le stockage des données. Chaque valeur du tableau s'appelle

un élément et on référence chaque élément par un numéro. Les numéros du tableau sont une suite séquentielle commençant par zéro (0) (voir la figure 12.5). Supposons que l'on ait un tableau nommé `fruit`. On peut lui assigner des valeurs de la manière suivante :

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
fruit=new Array();
fruit[0]="framboises";
fruit[1]="pêches";
fruit[2]="pommes";
fruit[3]="prunes";
document.write(fruit[1]);
var monFruit=fruit.pop();
document.write("<br>" +monFruit + "<br>");
document.write(fruit.length);
</script>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Tableau 1</title>
</head>
</html>
```

Le résultat du programme précédent est l'affichage à l'écran des mots « pêches », « prunes » et « 3 ». Le mot « pêches » a été extrait du tableau grâce à un numéro et placé dans une fonction d'affichage à l'écran. Puis en utilisant la méthode `pop()`, l'élément au sommet du tableau a été placé dans une variable nommée `monFruit` et affiché à l'écran. La méthode `pop()` a supprimé un élément du tableau pour le placer dans la variable `monFruit`, si bien que le tableau ne comporte plus que trois éléments. La propriété `length` indique le nombre d'éléments du tableau. Chaque élément du tableau fonctionne exactement comme une variable, la seule différence étant qu'il fait partie d'un objet plus grand, le tableau.

12.3.3 Objets

Le dernier type de données utilisé pour stocker les valeurs est un objet. Tous les objets sont similaires aux tableaux en ce sens où ils peuvent contenir plusieurs valeurs, mais ils ont plusieurs propriétés intégrées. Ces propriétés ont des valeurs fixes (appelées constantes) ou des valeurs qui changent en fonction des circonstances. Si l'on reprend un extrait du programme de la section précédente, on peut afficher la longueur du tableau :

```
...
document.write("<br>");
document.write(fruit.length);
```

La valeur de `fruit.length` est égale à 4 car la longueur d'un tableau est toujours supérieure d'une unité au numéro du dernier élément d'un tableau en raison du fait que la numérotation des éléments commence à zéro.

On appelle méthodes certaines propriétés des objets. Une méthode est une fonction qui réalise quelque chose qui a un rapport avec l'objet. Par exemple, l'objet tableau comporte une méthode `pop()` qui retourne le dernier élément d'un tableau. Grâce à cette méthode, on peut assigner à une variable une méthode d'objet, tout comme on peut assigner à une variable une fonction. Reprenons à nouveau le programme de la section précédente. Cette fois-ci, on assigne à la variable `monFruit fruit.pop()`. Cela signifie que ce qui se trouve au sommet de la pile du tableau est supprimé, mais si l'on utilise une assignation de variable, l'élément supprimé est assigné à la variable, comme l'indique l'extrait suivant :

```
...
var monFruit=fruit.pop();
document.write("<br>" + monFruit + "<br>");
document.write(fruit.length);
```

Quand on teste le programme de la section précédente, on constate que le dernier élément ajouté a la valeur `prunes` (c'est ce qui est affiché à l'écran). La longueur du tableau n'est plus 4, mais 3 à présent car la méthode `pop()` a supprimé un élément du tableau (au fait, `var` devant la variable `monFruit` est un mot-clé optionnel qui sert à déclarer une variable, mais cela permet de distinguer la variable des éléments du tableau dans ce listing).

Création de vos propres objets

Si l'on crée ses propres objets, on peut avoir une idée de la manière dont les objets fonctionnent dans le DOM. Afin de clarifier les choses, nous pouvons dire à présent que les propriétés d'un objet désignent en général à la fois ses propriétés et ses méthodes, mais quand on veut être plus spécifique et évoquer les parties individuelles d'un objet, on distingue bien les propriétés (certaines caractéristiques de l'objet) des méthodes (les fonctions associées à l'objet).

La création des objets est similaire à la déclaration des variables et à leur assignation de valeurs. L'objet en lui-même est une sorte de base d'opérations pour réaliser toute une série de choses différentes ayant des caractéristiques différentes. La première étape de la création d'un objet consiste simplement à utiliser un nom et le mot-clé `new`. Par exemple, le code suivant déclare un objet nommé `Additionneur` :

```
Additionneur=new Object();
```

Ensuite, pour ajouter une propriété, il faut trouver un nouveau nom et lui assigner une valeur. Le nom de l'objet et sa propriété sont séparés par un point (`.`). Par exemple, la code suivant ajoute une propriété nommée `premierNombre` et lui assigne la valeur 4 :

```
Additionneur.premierNombre=4;
```

Tout comme avec une variable, vous pouvez modifier la valeur `premierNombre` et lui assigner une autre valeur.

L'ajout d'une méthode (une fonction) est tout aussi facile, mais au lieu d'utiliser une fonction nommée, on utilise une fonction anonyme. Par exemple, le code suivant ajoute la valeur de deux propriétés de l'objet `Additionneur` et envoie le résultat à l'écran :

```
Additionneur.total=function()
{
     document.write(this.premierNombre + this.secondNombre);
}
```

Le mot-clé `this` est une référence à l'objet `Additionneur`. C'est comme si l'on écrivait `Additionneur.premierNombre`. Vous noterez aussi que le mot-clé `function()` n'a pas de nom, ce qui en fait une fonction anonyme.

Il est temps à présent de rassembler le tout pour voir comment cela fonctionne (voir le programme `ObjetUtilisateur.html` qui est disponible dans les fichiers du chapitre) :

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
Additionneur=new Object();
//Propriétés de l'objet
Additionneur.premierNombre=4;
Additionneur.secondNombre=66;
//Méthode de l'objet
Additionneur.total=function()
{
     document.write(this.premierNombre + this.secondNombre);
}
//Déclenchement de la méthode !
Additionneur.total();
</script>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Objet simple</title>
</head>
</html>
```

Vous noterez que la méthode `Additionneur.total()` utilise la méthode `document.write()` (vous pouvez détecter les méthodes en JavaScript en cherchant les parenthèses). Vous noterez aussi que pour déclencher une méthode, il suffit de lister le nom de l'objet suivi du nom de la méthode. Quand on teste le programme, on constate que le résultat est le total des deux propriétés.

Retour sur le DOM et les objets du navigateur

Ce chapitre a traité un grand nombre de notions très rapidement. En fait, la dernière section est le premier pas vers la programmation orientée objets (POO). Ne vous inquiétez pas si vous n'avez pas tout compris car l'objectif est de vous familiariser avec l'utilisation du DOM en HTML5. Si vous comprenez à présent les termes comme propriétés et méthodes, ils ne vous paraîtront plus étrangers.

Au fur et à mesure que nous découvrons les nouvelles fonctionnalités de HTML5, vous serez plus à l'aise avec la terminologie et vous comprendrez mieux ce qui se passe. En d'autres termes, il sera plus facile d'apprendre. Cela ne signifie pas qu'il faut devenir un programmeur expert en POO pour comprendre tout ceci, mais que quelques notions de base en POO vous aideront à comprendre le DOM et les objets du navigateur qui se révèlent pratiques quand on utilise des éléments comme `canvas`.

Tout au long de cet ouvrage, vous avez rencontré des objets qui appartiennent au navigateur et que je n'ai pas étudiés en tant que tels. Le navigateur possède des objets qui sont importants pour travailler avec HTML5, notamment les objets suivants :

- `History`
- `Location`
- `Navigator`
- `Screen`
- `Window`

Par exemple, dans la section « Types de données » de ce chapitre, vous avez vu comment la propriété `navigator.platform` était utilisée pour trouver le type d'ordinateur sur lequel le programme s'exécutait.

Le DOM HTML5 a beaucoup plus d'objets et le plus utilisé est la propriété `Document`. La liste des objets est identique à la liste des éléments si bien qu'une liste de tous les objets du DOM est une liste de tous les éléments plus d'autres objets qui sont utilisés conjointement avec le DOM. Par exemple, les objets suivants font partie du DOM HTML5 mais ils ne sont pas exactement des éléments :

- `Document`
- `Event`
- `Image`
- `Link`
- `Meta`

Nous avons vu certains de ces objets dans des balises. Par exemple, l'objet image se trouve dans les balises `<img>` et ses propriétés sont similaires aux attributs de l'élément `img`. D'autres objets, comme document, sont impliqués dans le DOM en ce sens où une page Web est un document. L'objet `event` est employé dans la gestion des événements avec des méthodes comme `onClick`. Pour le reste, il s'agit d'éléments avec lesquels vous devez être familier, mais au lieu d'avoir des attributs d'une balise, vous devez vous attendre à trouver des propriétés avec des noms identiques et des fonctions qui sont équivalentes à certains attributs.

12.4 À VOUS DE JOUER !

Les sources de données sont importantes à comprendre et pour mieux appréhender ces notions il faut les pratiquer en utilisant différents types. Voici le défi à relever :

1. **Choisissez une phrase, par exemple « Tous les objets sont composés de propriétés et de méthodes ».**
2. **Assignez cette chaîne de caractères à une variable et utilisez** `document.write()` **pour l'afficher à l'écran.**
3. **Décomposez cette phrase en plusieurs mots et placez chaque mot dans les éléments d'un tableau, puis avec la méthode** `array.pop()` **et** `document.write()` **affichez tous les mots de la phrase à l'écran pour ne former qu'un seul message.**
4. **Pour finir, créez un objet avec une propriété dont le contenu est la phrase que vous avez choisie. Créez une méthode pour la propriété qui affiche la phrase à l'écran.**

13

Amélioration des sites avec les canvas

Objectif

La balise `<canvas>` (en anglais, *canvas* signifie canevas, mais aussi toile de tableau) constitue l'un des ajouts les plus importants de HTML5. Grâce à elle, vous pouvez dessiner ce que vous voulez sur une page HTML5, avec simplement deux attributs, `width` et `height`. L'élément `canvas` est cependant implémenté dans ce que l'on pourrait appeler un style DOM (Document Object Model ; voir le chapitre 12 où le DOM a été décrit en détail). Fondamentalement, un style DOM implique l'écriture du code JavaScript nécessaire avec des références à des objets ainsi qu'à leurs méthodes et propriétés.

Si ce genre de discours vous inquiète, relaxez-vous. Tout au long de cet ouvrage, les balises HTML5 (les éléments) ont utilisé des attributs et les attributs ne sont que des propriétés d'éléments. Pour l'essentiel, l'écriture de code JavaScript se résume à l'assignation de valeurs à des propriétés, et comme vous savez déjà assigner des valeurs à des attributs (par exemple `height="200"`), il n'y a rien de bien nouveau dans l'écriture de cette sorte de code.

JavaScript emploie une sorte de « POO allégée ». Le DOM représente une programmation orientée objets (POO) car il n'y a que des références à des objets et à leurs propriétés. En adoptant un JavaScript au style similaire (création d'objets puis assignation de propriétés et de méthodes), votre code va beaucoup ressembler aux expressions extraites du DOM.

13.1 INTRODUCTION À CANVAS

Dans la mesure où l'élément `canvas` est une partie cruciale de HTML5 qui ne fonctionne qu'avec des navigateurs compatibles HTML5, la première chose à faire consiste à avertir les utilisateurs qu'ils ont besoin d'un navigateur HTML5. Il existe plusieurs méthodes pour déterminer si la balise canvas peut fonctionner avec un navigateur, mais le moyen le plus simple et le plus informatif (pour l'utilisateur) est de placer un message dans le conteneur `<canvas>`. Seuls les utilisateurs ne disposant pas de navigateurs compatibles HTML5 verront le texte dans le conteneur. Par exemple, la ligne suivante affiche un message d'avertissement tout en restant invisible pour les utilisateurs qui ont des navigateurs HTML5 :

```
<canvas id="colorScheme" width="600" height="100" >Vous avez <i>vraiment</i>
besoin de vous procurer un navigateur compatible HTML5 si vous voulez profiter
des <b>canvas !</b></canvas>
```

La figure 13.1 illustre ce qui apparaît quand on ouvre la page avec la balise `<canvas>` sur une version ancienne du navigateur Internet Explorer.

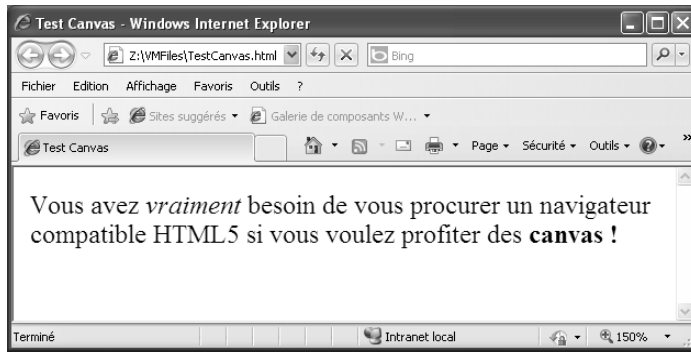


Figure 13.1 — Message affiché par un navigateur qui n'est pas compatible HTML5.

Quand on exécute ce même programme sur la version 9 d'Internet Explorer, on découvre le résultat illustré à la figure 13.2 : le dessin `canvas` apparaît, mais pas le message.

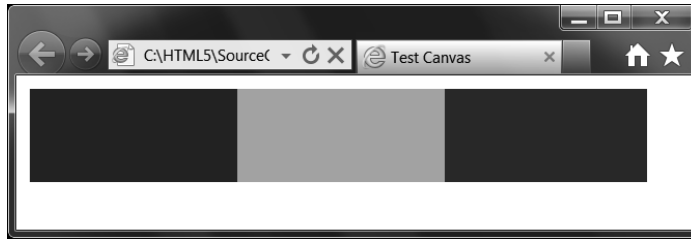


Figure 13.2 — `Canvas` sur un navigateur HTML5.

On pourrait écrire quelque chose de plus sophistiqué, mais le message est suffisamment explicite pour que l'on s'en contente. Si vous avez un navigateur HTML5, tout se passe correctement et si tel n'est pas le cas, l'utilisateur est averti par un message (il est souhaitable d'adapter le message à votre public).

Avant de se lancer dans la création de vos propres dessins, nous allons étudier une autre manière de gérer les utilisateurs qui n'ont pas de navigateur HTML5. Outre l'ajout de texte, vous pouvez aussi intégrer des photos ou n'importe quoi d'autre dans le conteneur `<canvas>`. Par exemple, le script suivant (`PhotoCanvas.html` disponible dans les fichiers du chapitre) offre une alternative à une présentation plus sophistiquée d'une photo à l'aide de `canvas`.

```
<!DOCTYPE HTML>
<html>
<head>
<style type="text/css">
body {
font-family:Verdana, Geneva, sans-serif;
background-color:#060;
color:#0FC;
}
img {
padding-top:10px;
padding-bottom:10px;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Pêcheur en herbe</title>
</head>
<body>
<body onLoad="CanvasMaster.showCanvas()">
<canvas id="photo" width="300" height="272" >Cher visiteur, si vous voyez ce
message, cela signifie que vous n'avez pas de navigateur HTML5 (mais vous
pouvez voir la photo et sa légende).<br>
  <figure> <img src="fisherkid.jpg" width="300" height="272" alt="kid
fishing"/><br>
    <figcaption>Pêcheur en herbe</figcaption>
  </figure>
</canvas>
</body>
</html>
```

Non seulement l'utilisateur qui ne possède pas de navigateur HTML5 obtient le message d'avertissement, mais ce dernier est affiché dans le style qui est défini en CSS3. L'utilisateur est aussi capable de visualiser à la fois la photo et sa légende, comme cela est illustré à la figure 13.3.

Si vous utilisez un navigateur HTML5, le programme précédent affiche un grand écran vide de couleur verte qui ne contient rien. Dans ces conditions, assurez-vous, quand vous employez une alternative pour les utilisateurs qui n'ont pas de navigateur HTML5, d'avoir un véritable contenu dans la balise `canvas`.

Figure 13.3 — Affichage d'un message alternatif pour les navigateurs non compatibles HTML5. © David Sanders

13.1.1 Implémentation d'un simple canvas

Quand on travaille avec Adobe Dreamweaver pour créer une page HTML5, on peut prévisualiser la page en mode Création pour avoir une idée du résultat à l'écran. Cependant, si vous avez un conteneur `<canvas>`, vous ne pourrez visualiser que son contour. Cette silhouette fournit un excellent aperçu visuel de la manière dont `canvas` alloue une certaine partie de la page pour afficher les images, même si elle apparaît sous la forme d'un rectangle vide.

Fondamentalement, on démarre par un élément canvas vide qui est défini par les attributs largeur (`width`) et hauteur (`height`) de la balise `<canvas>`. Si vous vous représentez la première étape de création d'un canvas sur votre page Web comme le fait de tendre une toile sur un cadre, cela vous aidera à visualiser le processus.

Explication du quadrillage

Pour exploiter avec succès les `canvas`, vous devez comprendre comment fonctionnent le quadrillage et le système de coordonnées cartésiennes. Le coin supérieur gauche représente la position 0,0 de la page. Quand on se déplace vers la droite, la première valeur augmente. Si vous vous déplacez de 15 pixels à droite, la valeur passe à 15,0 (c'est l'axe des x). Si vous vous déplacez vers le bas, la seconde valeur (l'axe des y) augmente. Si vous descendez de 20 pixels, la position passe alors à 15,20. Supposons que vous vouliez utiliser cette position comme point de départ pour créer un carré de 100 pixels de côté. Cela aide de visualiser la position et la taille par rapport à la page

Web avec le quadrillage, mais on a une idée plus claire de l'image que l'on crée sans le quadrillage. En fait, les deux systèmes sont utiles.

Préparation du tracé du canvas

Nous sommes maintenant prêts à remplir la boîte vide et nous avons besoin de JavaScript pour ce faire. La seule chose que l'on fait avec la balise `<canvas>` c'est de décrire la zone où l'on place l'image à l'aide d'un contexte de rendu et d'un ID de référence. Nous allons commencer par faire simple et notre premier dessin sera créé par la balise suivante :

```
<canvas id="redHot" width="100" height="100" >
```

Cela doit vous paraître assez familier. Les attributs `width` et `height` ont une valeur de 100 pixels, et le nom de l'objet canvas a été défini à `redHot` (j'ai déjà traité de la balise de fermeture `</canvas>` et du message dans le conteneur). Tout le reste du travail n'est que de la programmation JavaScript communiquant avec le DOM.

Comme je l'ai mentionné plus haut, je vais tenter de simplifier les choses en utilisant un peu de POO dans le code JavaScript pour refléter la structure de programmation du DOM. La première tâche consiste donc à créer un objet et il existe une méthode pour ce faire :

```
CanvasMaster=new Object();
CanvasMaster.showCanvas=function()...
```

Comme vous l'avez vu au chapitre 12, il y a ici une définition d'un objet et de sa méthode (une fonction qui va appeler les opérations JavaScript quand cela sera nécessaire).

Ensuite, le programme a besoin d'un moyen pour accéder au nœud DOM du canvas. Il s'agit de la partie du DOM qui a le `canvas` ainsi que les propriétés et méthodes liées au canvas. La première étape consiste à créer un objet qui contient le nœud du DOM. Plutôt que de penser à assigner un nœud à une variable, représentez-vous la création d'une instance d'un objet qui a les propriétés et les méthodes de l'objet `canvas`.

```
canvasNow = document.getElementById("redHot");
```

Cette ligne crée un objet qui contient l'objet canvas nommé redHot.

Une fois que l'on a une instance d'un objet canvas, le programme a besoin d'un contexte de rendu. Le seul contexte pratiquement disponible se nomme `2d`, ce qui laisse supposer un contexte de rendu en deux dimensions. L'objet canvas (`canvasNow`) possède une méthode nommée `getContext()` pour obtenir le contexte de rendu.

```
contextNow = canvasNow.getContext('2d');
```

L'instance du contexte de rendu se nomme `contextNow` et possède les méthodes et les propriétés du contexte de rendu `2d`.

Réalisation du dessin

Avant d'étudier la réalisation du dessin, vous vous posez peut-être des questions sur les objets `canvasNow` et `contextNow`. S'agit-il réellement de variables ? Après tout, on peut assigner des objets à des variables. C'est en effet une manière de se représenter les choses, mais les objets sont assignés aux variables avec leurs propres méthodes et propriétés. Ne s'agit-il donc pas plutôt d'instances d'objets ? Quand on assigne un nombre réel à une variable, il s'agit avant tout d'un nombre. On peut réaliser des calculs avec la variable comme on le ferait avec un nombre littéral. Au lieu d'ergoter pour savoir si les structures de programme sont en fait des variables ou des objets, traitez-les simplement comme des objets (tout comme les variables avec du texte ou des nombres peuvent être traitées comme des variables de type string ou number).

On va d'abord assigner une couleur au dessin. Vous pouvez utiliser l'une des techniques de création de couleur que nous avons décrites au chapitre 4. Cet exemple emploie le format hexadécimal :

```
contextNow.fillStyle = '#cc0000';
```

La propriété `fillStyle` ne sert que pour la couleur de remplissage et non pas pour le trait (le contour) de l'objet.

La couleur de remplissage a ensuite besoin d'une forme à remplir. Pour remplir un rectangle, on utilise la syntaxe suivante :

```
contextNow.fillRect(5,20,100,100);
```

La figure 13.4 explique chaque détail de la ligne de code en la décomposant.

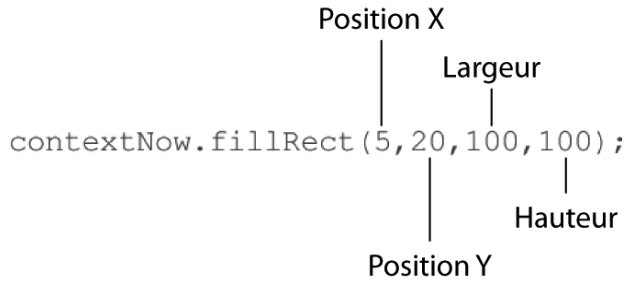


Figure 13.4 — Détails de la méthode `fillRect()`.

Les deux premières valeurs placent le dessin dans la zone du canvas (et non pas sur la totalité de la page Web) et les deux dernières valeurs spécifient la largeur et la hauteur du rectangle.

Pour finir, il faut remplir le rectangle avec la couleur spécifiée, ce qui est accompli par la ligne suivante :

```
contextNow.fill();
```

Peu importe le nombre d'opérations qui sont définies, une seule méthode `fill()` s'occupe de tous les remplissages qui sont définis dans une méthode plus grande.

Maintenant que toutes les pièces sont en place, il faut les assembler dans un programme HTML5. Le listing suivant (`SimpleCarre.html` disponible dans les fichiers du chapitre) contient la totalité du script :

```
<!DOCTYPE html>
<html>
<head>
<script language="javascript">
CanvasMaster=new Object();
CanvasMaster.showCanvas=function()
{
canvasNow  = document.getElementById("redHot");
contextNow = canvasNow.getContext('2d');
contextNow.fillStyle = '#cc0000';  // couleur en hexadécimal
contextNow.fillRect(5,20,100,100); // x, y, largeur, hauteur
contextNow.fill();
}
</script>
<style type="text/css">
body {
font-family:Verdana;
color:#cc0000;
}
</style>
<title>Carré rouge</title>
</head>
<body onLoad="CanvasMaster.showCanvas()">
<figure>
  <canvas id="redHot" width="100" height="100" > Vous avez manqué le carré
rouge ! Adoptez un navigateur HTML5 ! </canvas>
  <figcaption> <br/>
    Carré rouge </figcaption>
</figure>
</body>
</html>
```

Comme vous pouvez le voir, le programme inclut du code CSS3 et une simple légende avec les balises appropriées `<figure>` et `<figcaption>` qui encadrent la balise `<canvas>`. Le résultat de ce script est illustré à la figure 13.5.

Vous noterez aussi que ce script contient un message pour les utilisateurs qui n'ont pas de navigateur HTML5, mais comme la figure 13.5 illustre l'image `canvas`, le navigateur n'affiche pas le contenu du conteneur `<canvas>`.

Réalisation de plusieurs dessins

Maintenant que vous avez vu comment créer un simple dessin, je vais vous montrer comment en créer plusieurs. Et pendant que nous y sommes, nous en profiterons pour tester cela sur un périphérique mobile pour voir comme la balise `<canvas>` et JavaScript fonctionnent dans un environnement mobile.

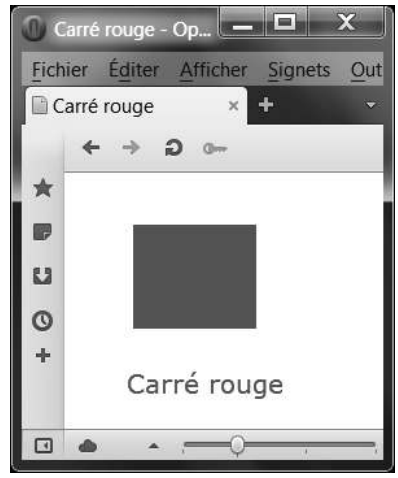


Figure 13.5 — Simple canvas s'affichant dans le navigateur Opera.

Le script suivant (`TortillaFlat.html` disponible dans les fichiers du chapitre) est très similaire au script utilisé pour créer le carré rouge de la figure 13.5, mais quand on dessine plusieurs objets, leur position devient plus importante, comme le script suivant le montre :

```
<!DOCTYPE html>
<html>
<head>
<script language="javascript">
//Valeurs du modèle couleurs collées ici : 8C6E37,BFA380,593723,736055,261F1E
CanvasMaster=new Object();
CanvasMaster.showCanvas=function()
{
canvasNow  = document.getElementById("totillaHues")
contextNow = canvasNow.getContext('2d');
contextNow.fillStyle = '#8C6E37';    // couleur en hexadécimal
contextNow.fillRect(5,20,100,100);   // x, y, largeur, hauteur
      // première couleur
contextNow.fillStyle = '#BFA380';    // couleur en hexadécimal
contextNow.fillRect(105,20,100,100); // deuxième couleur
contextNow.fillStyle = '#593723';    // couleur en hexadécimal
contextNow.fillRect(205,20,100,100); // troisième couleur
contextNow.fillStyle = '#736055';    // couleur en hexadécimal
contextNow.fillRect(305,20,100,100); // quatrième couleur
contextNow.fillStyle = '#261F1E ';   // couleur en hexadécimal
contextNow.fillRect(405,20,100,100); // cinquième couleur
contextNow.fill(); // on remplit tout !
}
</script>
<style type="text/css">
body {
font-family:Verdana;
color:#570026;
```

```
}
</style>
<title>Tortilla Flat !</title>
</head>
<body onLoad="CanvasMaster.showCanvas()">
<figure>
  <canvas id="totillaHues" width="500" height="120" > Pas de  tortillas pour
vous ! Adoptez un navigateur HTML5... rapidement ! </canvas>
  <figcaption> <br/>
    Tortilla Flat
  </figcaption>
</figure>
</body>
</html>
```

Les paramètres importants de ce script sont les deux premiers de la méthode `fillRect()`. Il s'agit des positions x et y, sachant que deux carrés ne peuvent pas occuper le même espace. Les carrés sont alignés sur une ligne horizontale, si bien que la seule chose à laquelle vous devez veiller est la valeur x dans la mesure où la position verticale restera identique.

Une fois que toutes les méthodes `fillStyle()` et `fillRect()` ont été définies, les dessins n'ont besoin que d'une seule méthode `fill()` pour s'afficher tous ensemble. La figure 13.6 illustre le résultat de la figure dans le navigateur mobile Safari sur un iPhone.

Figure 13.6 — Plusieurs dessins affichés dans un navigateur mobile.

L'image de la figure 13.6 vous semblera peut-être familière car au chapitre 4, le programme de définition de modèles de couleurs Adobe Kuler avait une apparence similaire et les couleurs utilisées dans le programme ont d'ailleurs été développées avec ce logiciel.

Ajout de traits et suppression de dessins

Il existe deux méthodes supplémentaires qui sont associées au tracé de rectangles : `strokeRect()` et `clearRect()`. Ces deux méthodes ont des paramètres similaires à la méthode `fillRect()` (à savoir `x`, `y`, `width`, `height`). Elles fonctionnent de manière identique dans la mesure où elles utilisent le même système pour spécifier les zones pour l'ajout de traits ou la suppression des dessins.

Le programme suivant (`TraitEtSuppression.html` disponible dans les fichiers du chapitre) montre comment on peut ajouter trois méthodes à un objet `CanvasMaster`, que je nommerai `ajouterTraits()`, `perforer()` et `raz()`. La première méthode dessine un cadre à l'intérieur de la zone du canvas, la seconde perce un trou au milieu du rectangle, et la troisième supprime tout ce qui existe dans la zone définie.

```
<!DOCTYPE html>
<html>
<head>
<script language="javascript">
//couleurs: 595241,B8AE9C,FFFFFF,ACCFCC,8A0917
CanvasMaster=new Object();
CanvasMaster.showCanvas=function()
{
canvasNow  = document.getElementById("strokeAndChomp");
contextNow = canvasNow.getContext('2d');
contextNow.fillStyle = '#ACCFCC';
contextNow.fillRect(5,20,100,100);
contextNow.fill();
}
CanvasMaster.ajouterTraits=function()
{
contextNow.strokeStyle='#595241';
contextNow.strokeRect(7,22,91,76);
}
CanvasMaster.raz=function()
{
contextNow.clearRect(5,20,100,100);
}
CanvasMaster.perforer=function()
{
contextNow.clearRect(40,45,30,30);
}
</script>
<style type="text/css">
body {
font-family:Verdana;
color:#8A0917;
background-color:#B8AE9C;
}
a {
text-decoration:none;
color:#595241;
margin-left:16px;
}
</style>
<title>Trait et suppression</title>
```

```
</head>
<body onLoad="CanvasMaster.showCanvas()">
<article>
  <figure>
    <canvas id="strokeAndChomp" width="100" height="100" > Passez à un
navigateur HTML5 ! </canvas>
    <figcaption> <br/>
      Carré </figcaption>
  </figure>
  <section>
    <p><a href="#" onClick="CanvasMaster.ajouterTraits()">Ajouter des
traits</a></p>
  </section>
  <section>
    <p><a href="#" onClick="CanvasMaster.raz()">Effacer le carré</a></p>
  </section>
  <section>
    <p><a href="#" onClick="CanvasMaster.perforer()">Perforer le carré</a></p>
  </section>
  <section>
    <p><a href="#" onClick="CanvasMaster.showCanvas()">Redessiner le
carré</a></p>
  </section>
</article>
</body>
</html>
```

Cette page est mise en forme pour un terminal mobile. Elle a été testée avec le navigateur Opera Mini sur un iPhone, comme cela est illustré à la figure 13.7.

Un carré bleu apparaît lors du chargement initial. Quand on clique sur le lien Ajouter des traits, un cadre apparaît à l'intérieur de l'image originale. Si vous ajoutez plus de traits, vous constaterez que le trait s'épaissit. Quand on clique sur le lien Perforer le carré, un petit carré apparaît au centre du carré bleu. Le lien Effacer le carré supprime à la fois l'image et les traits. Si vous cliquez sur le lien Ajouter des traits après avoir supprimé le carré bleu, vous ne verrez que le cadre sans le rectangle bleu.

13.1.2 Images et ombres dans les canvas

L'une des fonctionnalités les plus simples et les plus amusantes des `canvas` consiste à les utiliser avec des images que l'on a chargées. La figure 13.3 illustre un exemple typique d'image que l'on peut charger sur une page Web avec la balise `<img>`. L'usage de cette balise est parfait, mais on peut la rendre encore plus intéressante avec la balise `<canvas>`.

Chargement d'une image dans un canvas

Pour charger une image, qu'il s'agisse d'un fichier GIF, PNG ou JPEG, il faut un objet `Image` qui soit créé avec JavaScript. Au sein de la méthode utilisée pour créer un contexte de rendu, on risque que l'utilisateur voit une page vide à l'emplacement de l'image chargée, à moins que l'on ne dispose d'un événement qui indique que le

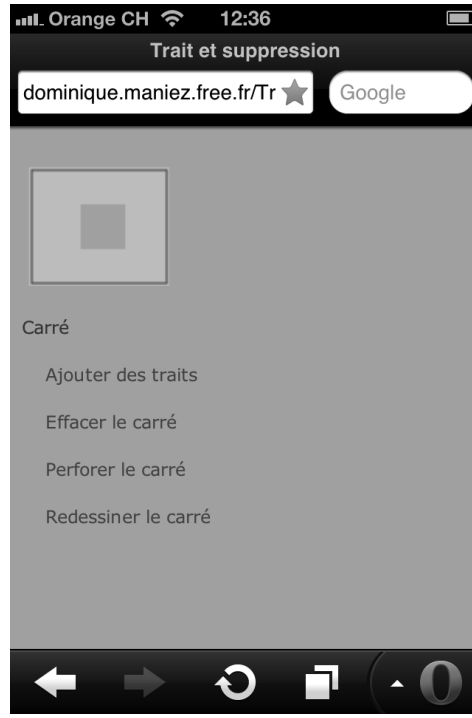


Figure 13.7 — Ajout de traits et suppression de tout ou partie d'un rectangle.

fichier a été chargé. Heureusement, c'est plutôt simple à faire avec le gestionnaire d'événements `onLoad`, comme le montre l'extrait de code suivant :

```
...
pic = new Image();
pic.onload = function()
{
     contextNow.drawImage(pic,10,10);
}
 pic.src = 'imageName.jpg';
...
```

La méthode de contexte de rendu `drawImage()` possède trois paramètres :

- **La référence au fichier qui est chargé :** dans cet exemple, l'étiquette `pic` est la référence du fichier qui est utilisé.
- **La position x et y :** c'est un peu plus compliqué que d'utiliser la balise `<img>`, mais pas tant que cela, et cette méthode permet de placer l'image où l'on veut à l'intérieur des paramètres du canvas.
- **La source de l'image :** on ajoute la source de l'image au sein de la méthode qui crée le contexte de rendu, ce qui ne diffère pas de l'identification employée par l'élément `img`.

Ajout d'une ombre portée

L'ajout d'une ombre portée à une image donne un aspect tridimensionnel à la page. Le contexte de rendu a quatre paramètres pour définir les propriétés de l'ombre :

- `shadowColor="couleur";`
- `shadowOffsetX=valeur horizontale;`
- `shadowOffsetY=valeur verticale;`
- `shadowBlur=valeur de flou;`

La couleur peut être assignée en utilisant l'une des méthodes étudiées au chapitre 4. Le décalage de l'ombre par rapport à l'image dépend de la grandeur que vous souhaitez pour l'ombre. Testez différentes valeurs en commençant par une valeur proche de 5. Dans l'exemple suivant, chaque paramètre de décalage est initialisé à 10 afin de fournir suffisamment d'ombre pour que l'image se détache de l'écran sans pour autant écraser l'image. Enfin, la valeur de flou peut être supérieure ou inférieure en fonction des valeurs de décalage et de la quantité de flou que vous souhaitez. Avec de grandes valeurs de décalage, il faut une grande valeur de flou.

Pour que l'ombre produise un effet sur l'image, toutes les propriétés de l'ombre doivent être saisies avant l'écriture de la méthode `drawImage()`. C'est tout ce qu'il y a lieu de faire. L'autre partie JavaScript nécessaire pour définir le contexte de rendu du canvas est très similaire aux dessins de la section précédente. Le code suivant (`OmbrePhoto.html` disponible dans les fichiers du chapitre) charge une image et l'encadre avec une ombre portée :

```
<!DOCTYPE html>
<html>
<head>
<script language="javascript">
//couleurs: F4F1BC,736F36,BFB95A
CanvasMaster=new Object();
CanvasMaster.showCanvas=function()
{
canvasNow  = document.getElementById("picFrame");
contextNow = canvasNow.getContext('2d');
pic = new Image();
pic.onload = function()
{
contextNow.shadowColor ='#BFB95A';
contextNow.shadowOffsetX=10;
contextNow.shadowOffsetY=10;
contextNow.shadowBlur=4;
    contextNow.drawImage(pic,10,10);
}
    pic.src = 'fisherkid.jpg';
}
</script>
<style type="text/css">
body {
font-family:Verdana;
color:#736F36;
```

```
background-color:#F4F1BC;
}
</style>
<title>Photo encadrée</title>
</head>
<body onLoad="CanvasMaster.showCanvas()">
<article>
  <figure>
    <canvas id="picFrame" width="340" height="300" > Vous avez manqué une
photo parce que vous n'avez pas de navigateur HTML5. </canvas>
    <figcaption> <br/>
      Photo avec une ombre portée</figcaption>
  </figure>
</article>
</body>
</html>
```

Avant de placer vos propres images, vérifiez leur taille et la taille que la balise `<canvas>` a réservée. Dans cet exemple, il y avait suffisamment de place pour l'image (une photographie) et l'ombre portée. La figure 13.8 illustre le résultat dans le navigateur Opera.

Les combinaisons de couleurs utilisées avec l'image sont importantes. Vous constaterez que certaines couleurs fonctionnent mieux que d'autres. Celles utilisées à la figure 13.8 sont un ensemble monochromatique basé sur les couleurs de l'image. Comme vous pouvez le voir, l'ombre donne l'impression que la photo se soulève de l'écran.

Comparez l'image de la figure 13.3 à celle de la figure 13.8. À la figure 13.3, on voit ce qui se passe avec les navigateurs qui ne sont pas compatibles HTML5 ; à la figure 13.8, on voit ce dont est capable HTML5. Pour les navigateurs qui ne sont pas compatibles HTML5, vous pouvez quand même ajouter la même image et le modèle de couleurs sans l'ombre portée.

Travail avec les filtres

Avant de passer à des formes complexes, jetons un coup d'œil à l'utilisation des filtres pour ajouter des teintes aux images. Internet est une immense bibliothèque de photos et de dessins libres de droits et vous pouvez utiliser votre moteur de recherche favori pour trouver des images (rappelez-vous cependant que toutes les images que l'on trouve en ligne ne sont pas libres de droits et assurez-vous que vous avez la permission d'utiliser les images que vous trouvez). De nombreux dessins sont en noir et blanc et peuvent créer un contraste austère avec les autres éléments d'une page. Pour intégrer ce type d'image, on peut ajouter un filtre en créant une forme colorée partiellement transparente et en la plaçant au-dessus de l'image. Avec `canvas`, ce processus est assez simple et le point important se trouve dans la ligne suivante :

```
context.fillStyle = 'rgba(rn, gn, bn, alpha)';
```

Au lieu d'employer une valeur hexadécimale, le code utilise un codage RGB avec un canal « alpha » (`rgba()`) qui contrôle la transparence. Le dernier paramètre est

Figure 13.8 — Image et ombre portée avec la balise `<canvas>`. © David Sanders

une valeur comprise entre 0 et 1. Plus la valeur est élevée, plus l'image sera opaque. En utilisant une valeur inférieure à 1, vous pouvez contrôler le degré d'opacité. Le reste de la forme correspond aux dimensions de l'image et le filtre est positionné sur le même espace.

Pour intégrer une image au reste de la page, il est souhaitable d'ajouter une teinte en utilisant la couleur de fond. Le programme suivant (`ImageFiltree.html` disponible dans les fichiers du chapitre) ajoute d'abord l'image puis dessine un objet rectangulaire par-dessus doté d'une couleur de remplissage transparente.

```
<!DOCTYPE html>
<html>
<head>
<script language="javascript">
//couleurs: F26A4B,F2D091=rgb(242,208,145)
CanvasMaster=new Object();
CanvasMaster.showCanvas=function()
{
canvasNow  = document.getElementById("filterFrame");
contextNow = canvasNow.getContext('2d');
pic = new Image();
pic.onload = function()
{
    contextNow.drawImage(pic,0,0);
contextNow.fillStyle = 'rgba(242, 208, 145, .6)';
contextNow.fillRect(0,0,472,306);
contextNow.fill();
```

```
}
    pic.src = 'dance.gif';
}
</script>
<style type="text/css">
body {
font-family:Verdana;
color:#F26A4B;
background-color:#F2D091;
}
</style>
<title>Filtre d'images</title>
</head>
<body onLoad="CanvasMaster.showCanvas()">
<article>
  <figure>
    <canvas id="filterFrame" width="472" height="306" > Vous avez raté la
danse ! Installez un navigateur HTML5 ! </canvas>
    <figcaption> <br/>
      Image filtrée</figcaption>
  </figure>
</article>
</body>
</html>
```

Vous noterez que la séquence charge d'abord l'image puis place le dessin au-dessus en utilisant l'extrait suivant :

```
contextNow.drawImage(pic,0,0);
contextNow.fillStyle = 'rgba(242, 208, 145, .6)';
contextNow.fillRect(0,0,472,306);
contextNow.fill();
```

Si le dessin est ajouté en premier, l'image repose dessus et il n'y a pas d'effet de filtre. Grâce à l'ajout d'un filtre, l'image s'intègre mieux à la page, comme l'illustre la figure 13.9.

Avec Adobe Photoshop ou tout logiciel de retouche graphique similaire, vous pourriez ajouter le filtre à l'image et charger l'image filtrée avec une balise standard `<img>`, mais l'utilisation de `canvas` et de HTML5 permet d'effectuer les modifications sans logiciel supplémentaire.

13.2 CRÉATION DE DESSINS COMPLEXES AVEC DES CANVAS

Les formes les plus simples sont des rectangles, ce qui va très bien pour tracer des carrés et des rectangles, et vous pouvez déjà faire beaucoup de choses avec des boîtes avant d'avoir besoin de lignes et de courbes. Cette section étudie les éléments de dessin complexe suivants qui font partie de l'objet canvas (le terme `context` fait référence au nom de l'objet du contexte de rendu).

Figure 13.9 — Image filtrée se fondant dans l'arrière-plan.

- `context.beginPath()`
- `context.moveTo(x, y)`
- `context.closePath()`
- `context.lineTo(x, y)`
- `context.quadraticCurveTo(cpx, cpy, x, y)`
- `context.bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)`
- `context.arcTo(x1, y1, x2, y2, radius)`
- `context.arc(x, y, radius, startAngle, endAngle, anticlockwise)`
- `context.rect(x, y, w, h)`
- `context.fill()`
- `context.stroke()`
- `context.clip()`
- `context.isPointInPath(x, y)`

Le fait de savoir comment utiliser ces méthodes avec une balise `<canvas>` ne vous garantit pas un résultat honorable. Le reste de ce chapitre détaille la plupart de ces méthodes. Vous devriez être capable de créer de nombreuses formes différentes après avoir achevé la lecture de ce chapitre.

13.2.1 Lignes et mouvement

Le meilleur moyen de commencer à réfléchir à l'utilisation des outils de canvas pour dessiner est de virtualiser tous les dessins sur un quadrillage, comme nous l'avons fait avec les rectangles. Cependant, étant donné la relative complexité du dessin libre, même avec des lignes droites, il faut partir des images sur un quadrillage. La figure 13.10 illustre deux dessins qui peuvent être créés à l'aide de lignes droites.

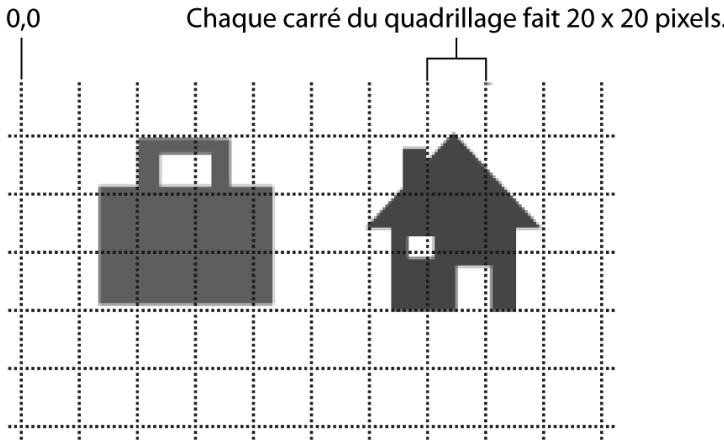


Figure 13.10 — Images sur un quadrillage.

Ce quadrillage contient des carrés de 20 pixels sur 20 pixels. Si vous prenez un crayon et une feuille de papier quadrillé (ou si vous activez le quadrillage sur un programme de dessin), vous pouvez répliquer les images de la figure 13.10. Si l'on commence par l'image de gauche de la figure 13.10, on peut recréer le dessin en accomplissant les étapes suivantes :

- **Placer le crayon à la position 40,20 sur le quadrillage**.
  Pour réaliser cela avec un `canvas`, utiliser `context.beginPath()` et `context.moveTo(40,20)`. Il s'agit du point de départ.
- **Tracer une ligne à partir du point de départ jusqu'à approximativement 72, 20 pour le haut de la poignée de la valise.**
  Utiliser `context.lineTo(72,20)` pour l'équivalent `canvas`.
- **Déplacer le crayon vers le bas jusqu'à 72, 38.**
  Utiliser `context.lineTo(72,38)` pour un dessin `canvas`.
- **Continuer de cette manière jusqu'à ce que le contour de la valise soit terminé.**
- Pour dessiner l'intérieur de la poignée, sélectionner le crayon, se déplacer à l'emplacement où l'on veut commencer à dessiner l'intérieur de la poignée.
  Avec `canvas`, utiliser `context.moveTo(x,y)` pour commencer à une nouvelle position puis utiliser `context.lineTo(x,y)` pour terminer. On n'a cependant pas à réutiliser `context.beginPath()`.

- **Dans un dessin au crayon, dès que le dessin est terminé, on dispose du contour de la valise. Avec canvas, il faut inclure context.stroke() pour ajouter les lignes.**

Quand on arrive à l'avant-dernier point du dessin, on peut utiliser la méthode `context.closePath()` pour aller au point de départ, et c'est la technique qui a été utilisée dans le programme. Le script suivant (`SimpleDessinTrait.html` disponible dans les fichiers du chapitre) fournit toutes les étapes.

```
<!DOCTYPE html>
<html>
<head>
<script language="javascript">
//couleurs: 8C6E37,BFA380
CanvasMaster=new Object();
CanvasMaster.showCanvas=function()
{
canvasNow  = document.getElementById("simpleDraw");
contextNow = canvasNow.getContext('2d');
contextNow.beginPath();
contextNow.moveTo(40,20);
contextNow.lineTo(72,20);
contextNow.lineTo(72,38);
contextNow.lineTo(88,38);
contextNow.lineTo(88,78);
contextNow.lineTo(28,78);
contextNow.lineTo(28,38);
contextNow.lineTo(40,38);
contextNow.lineTo(40,20);
contextNow.closePath();
contextNow.moveTo(46,26);
contextNow.lineTo(66,26);
contextNow.lineTo(66,38);
contextNow.lineTo(46,38);
contextNow.closePath();
contextNow.stroke();
}
</script>
<style type="text/css">
body {
font-family:Verdana;
color:#000000;
}
</style>
<title>Image tracée avec des lignes</title>
</head>
<body onLoad="CanvasMaster.showCanvas()">
<article>
  <figure>
    <canvas id="simpleDraw" width="90" height="80" > Vous avez raté un super
dessin car vous n'avez pas de navigateur HTML5.</canvas>
    <figcaption> <br/>
      Picasso est passé par là</figcaption>
  </figure>
```

```
</article>
</body>
</html>
```

La figure 13.11 illustre le résultat attendu (si vous avez travaillé avec vos propres coordonnées, votre dessin est probablement meilleur).

Figure 13.11 — Image dessinée à l'aide d'un canvas.

Jusqu'ici tout va bien, mais la valise originale est de couleur marron et il va donc nous falloir de la couleur. Pour ajouter de la couleur, on utilise la même technique que pour les rectangles : `context.fillStyle ="couleur"`. Les méthodes de dessin complexe incluent `context.fill()` pour remplir un contour, si bien qu'en remplaçant `context.stroke()` par `context.fill()`, et en ajoutant une méthode `fillStyle` le tour est joué. La figure 13.12 illustre le résultat.

Figure 13.12 — L'image est remplie en couvrant la poignée.

En examinant la figure 13.12, on constate que le contour et la couleur sont corrects, et qu'il n'y a plus de poignée, mais un bloc à la place. Quand une série de méthodes de dessin sont utilisées sans commencer un nouveau chemin et qu'ensuite la méthode `context.fill()` est appelée, elle remplit jusqu'au début du chemin. Cela a pour résultat que tout est rempli et non pas seulement les parties désirées.

Pour régler ce problème, deux méthodes `context.fill()` sont employées : une à la fin du premier tracé de la valise et l'autre à la fin du tracé de la poignée. Le premier tracé est rempli en marron et le second tracé est rempli en blanc. En outre, un second `context.beginPath()` est ajouté à la fin du tracé de la poignée. Le programme suivant (`SimpleDessinTraitRempli.html` disponible dans les fichiers du chapitre) contient tout le code nécessaire révisé pour générer l'image correctement remplie.

```
<!DOCTYPE html>
<html>
<head>
<script language="javascript">
//couleurs: 960, fff, 000
CanvasMaster=new Object();
CanvasMaster.showCanvas=function()
{
canvasNow  = document.getElementById("briefCase");
contextNow = canvasNow.getContext('2d');
contextNow.beginPath();
contextNow.moveTo(40,20);
contextNow.lineTo(72,20);
contextNow.lineTo(72,38);
contextNow.lineTo(88,38);
contextNow.lineTo(88,78);
contextNow.lineTo(28,78);
contextNow.lineTo(28,38);
contextNow.lineTo(40,38);
contextNow.lineTo(40,20);
contextNow.closePath();
contextNow.fillStyle ="#960";
contextNow.fill();
contextNow.beginPath();
contextNow.moveTo(46,26);
contextNow.lineTo(66,26);
contextNow.lineTo(66,38);
contextNow.lineTo(46,38);
contextNow.closePath();
contextNow.fillStyle ="#fff";
contextNow.fill();
}
</script>
<style type="text/css">
body {
font-family:Verdana;
color:#000;
}
</style>
<title>Dessin rempli</title>
</head>
<body onLoad="CanvasMaster.showCanvas()">
<article>
  <figure>
    <canvas id="briefCase" width="90" height="80" > Vous avez raté un super
dessin car vous n'avez pas de navigateur HTML5.</canvas>
    <figcaption> <br/>
      Picasso est passé par là</figcaption>
```

```
  </figure>
</article>
</body>
</html>
```

Quand on teste cette version modifiée, les résultats sont très proches du dessin original. Comparez la figure 13.10 avec la figure 13.13 pour voir comme l'image générée par programme est proche de l'originale.

Vous pouvez utiliser les lignes pour dessiner tout ce qui ne contient pas des courbes. Dans la section suivante, nous allons voir comment ajouter des courbes à vos outils artistiques `canvas`.

Figure 13.13 — Version finale du dessin de la valise.

13.2.2 Courbes

La réalisation de courbes, même avec des outils de dessin, est plus délicate que le tracé de lignes droites. Pour comprendre comment réaliser des courbes, je débuterai cette section par une discussion sur les arcs et les méthodes canvas du DOM qui permettent de les créer. Nous ferons un peu de géométrie, mais pas trop (il est vraiment nécessaire de comprendre un peu de géométrie, mais rassurez-vous, cela reste très basique).

La première chose qu'il faut comprendre, c'est la différence entre les degrés et les radians. La plupart des gens savent qu'un cercle fait 360 degrés. Sur une rose des vents, 360 ou 0 degré (midi) représente le nord. Quand on se déplace dans le sens des aiguilles d'une montre de 90 degrés (3 heures), la boussole pointe vers l'est ; à 180 degrés (6 heures), elle pointe vers le sud, et à 270 degrés (9 heures), elle pointe vers l'ouest.

Vous devez cependant utiliser des radians à la place des degrés, si bien que tous les degrés doivent être convertis en radians. Pour ce faire, utilisez la formule suivante :

$$\text{Radians} = (\text{PI} \div 180) \cdot \text{degrés}$$

Admettons que vous vouliez connaître les radians pour le plein ouest (9 heures), 270 degrés :

Radians = (3.14159265 ÷ 180) = 0.01745329251994

Radians = 0.01745329251994 · 270

Radians = 4.71238898

On peut réaliser la même chose en multipliant simplement les degrés par 0,01745329251994 ou bien en écrivant en JavaScript :

```
radians = (Math.PI/180)* degrés;
```

Vous trouverez aussi beaucoup de calculatrices en ligne qui feront les conversions à votre place.

Arcs

La méthode DOM canvas pour dessiner des arcs s'appelle `context.arc()`. Cette méthode comporte plusieurs paramètres qui ont besoin d'être compris individuellement, mais aussi les uns par rapport aux autres :

- `x,y` : centre du cercle
- `radius` : rayon du cercle
- `startAngle` : point de départ de l'arc exprimé en radians
- `endAngle` : point d'arrivée de l'arc exprimé en radians
- `anticlockwise` : booléen (`true` représente le sens contraire des aiguilles d'une montre et `false` le sens des aiguilles d'une montre)

Je trouve utile d'imaginer soit une rose des vents, soit une horloge avec les quatre points cardinaux et les heures et degrés : nord (midi ou 0 degré), est (3 heures ou 90 degrés), sud (6 heures ou 180 degrés) et ouest (9 heures ou 270 degrés). Voici un exemple d'instruction d'un arc complet :

```
contextNow.arc(150,100,50,six,0,true);
```

Cet arc a son centre à x = 150 et y = 100, et un rayon de 50. L'angle de départ est défini à 6, qui est une variable que nous avons créée pour représenter la position de 6 heures ou 180 degrés. La valeur de la variable a été convertie en radians. Les degrés et les radians ont la même valeur à la position de midi (0), qui est dans notre exemple l'angle d'arrivée. Pour finir, l'arc est défini à true, ce qui signifie qu'il est tracé dans le sens contraire des aiguilles d'une montre.

Le programme suivant est utilisé pour expérimenter différents arcs. Quatre variables (12, 3, 6 et 9) sont définies en radians et correspondent aux positions sur une horloge. Certaines instructions sont mises en commentaires, mais elles seront utilisées ultérieurement.

```
<!DOCTYPE HTML>
<html>
<head>
```

```
<script type="text/javascript">
CanvasMaster=new Object();
CanvasMaster.showCanvas=function()
{
      canvasNow  = document.getElementById("beHappy");
      contextNow = canvasNow.getContext('2d');
      contextNow.beginPath();
      contextNow.moveTo(0,0);
      contextNow.lineTo(300,0);
      contextNow.lineTo(300,200);
      contextNow.lineTo(0,200);
      contextNow.closePath();
      contextNow.stroke();
      // RADCON = (Math.PI/180) ;
      RADCON=0.01745329251994;
      twelve=0;
      three = RADCON * 90;
      six = RADCON * 180;
      nine = RADCON * 270;
      contextNow.beginPath();
      contextNow.arc(125,100,50,six,twelve,true);
      //contextNow.closePath();
      //contextNow.fill()
      contextNow.stroke();
}
</script>
<style type="text/css">
body {
      font-family:Verdana, Geneva, sans-serif;
      color:#cc0000;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Sourire</title>
</head>
<body onLoad="CanvasMaster.showCanvas()">
<figure>
  <canvas id="beHappy" width="300" height="200" > Vous ne voyez pas de sourire
car vous n'avez pas de navigateur HTML5 !</canvas>
  <figcaption>
    <p>Le rectangle représente les limites du canvas</p>
  </figcaption>
</figure>
</body>
</html>
```

La variable `RADCON` est une constante ($\pi \div 180$), si bien que tous les degrés ont été définis en radians en multipliant leur valeur par `RADCON`. Comme nous l'avons signalé, les noms des variables représentent les positions sur une horloge. De plus, un rectangle autour de la zone où l'arc est dessiné représente les limites de la largeur et de la hauteur de la balise `<canvas>`. La figure 13.14 illustre le résultat.

Le point de départ de l'arc est à gauche et il se déplace dans le sens contraire des aiguilles d'une montre jusqu'au point d'arrivée sur la droite. Modifiez la ligne suivante :

```
contextNow.arc(125,100,50,six,twelve,true);
```

en :

```
contextNow.arc(125,100,50,six,twelve,false);
```

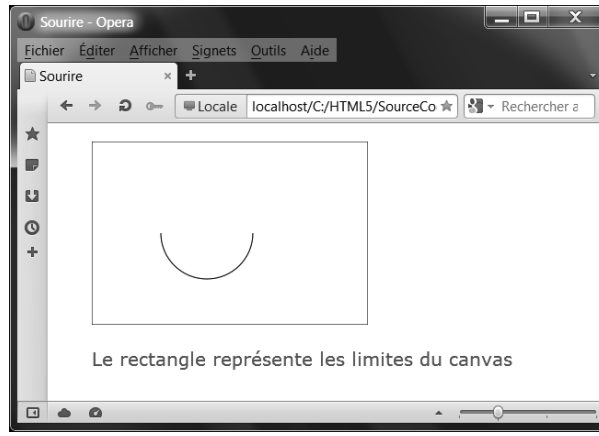


Figure 13.14 — Arc réalisé avec un `canvas`.

Cela modifie le sens du tracé qui passe du sens contraire des aiguilles d'une montre au sens des aiguilles d'une montre ; cela constitue une belle différence, comme vous le constaterez si vous testez le programme.

Ensuite, en utilisant le même programme, revenez à sa version originale en modifiant à nouveau cette ligne :

```
contextNow.arc(125,100,50,six,twelve,true);
```

Supprimez ensuite les lignes de commentaire (`//`) de la ligne suivante :

```
//contextNow.closePath();
```

Et testez à nouveau le programme. Cette dernière modification remplit l'arc. Supprimez le commentaire de la ligne suivante :

```
//contextNow.fill()
```

pour qu'elle devienne :

```
contextNow.fill()
```

Et mettez en commentaire la ligne suivante pour qu'elle apparaisse comme ceci :

```
//contextNow.stroke()
```

Quand toutes les modifications sont terminées, votre arc ressemble à présent à une bouilloire, comme cela est illustré à la figure 13.15.

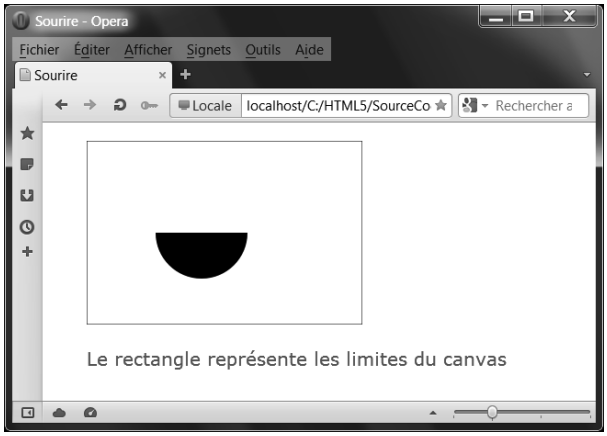


Figure 13.15 — Arc avec un chemin fermé et rempli.

Le seul moyen d'apprendre réellement à travailler avec des arcs est de s'entraîner. Utilisez le script de cette section pour tester différentes choses.

Cercles et dégradés

Jusqu'à présent, nous n'avons utilisé qu'un seul type de remplissage, le remplissage plein. Dans cette section, vous allez voir comment créer un cercle en utilisant un arc et en le remplissant avec un dégradé.

La création des cercles est facile avec la méthode `context.arc()`. Les paramètres du radian sont 0 et `Math.PI*2`. Le paramètre du sens des aiguilles d'une montre est égal à `false` (c'est là toute l'astuce). L'exemple suivant crée un grand cercle qui est rempli par un dégradé, afin qu'il ressemble à un coucher de soleil :

```
contextNow.arc(200,200,150,0,Math.PI*2,false);
```

La création d'un remplissage en dégradé, à la fois linéaire et radial est très simple. La première étape consiste à utiliser la méthode DOM canvas `context.createLinearGradient()`. La méthode attend quatre paramètres : `x0`, `y0`, `x1`, `y1`. Le remplissage en dégradé va de `x0` à `x1` et de `y0` à `y1`. Un dégradé strictement linéaire de la gauche vers la droite aurait une valeur unique de `x1`, et le reste serait égal à 0. Un dégradé de haut en bas aurait une valeur en `y0` ou `y1`, avec le reste égal à 0.

Pour définir les couleurs du dégradé, on utilise la méthode `gradient.addColorStop()` qui attend deux paramètres. Le premier est un nombre

compris entre 0 et 1 et le deuxième est la couleur. Quand les paramètres sont définis, on assigne le dégradé à `context.fillStyle`. L'extrait de code suivant illustre les étapes à accomplir pour ajouter un remplissage en dégradé :

```
sunsetGradient=contextNow.createLinearGradient(0, 0, 0,379);
sunsetGradient.addColorStop(0, "yellow");
sunsetGradient.addColorStop(1, "#cc0000")
contextNow.fillStyle = sunsetGradient;
```

Dans cet exemple particulier, le dégradé est vertical. La première couleur, le jaune, est au sommet et la deuxième couleur, le rouge, est en bas. Le script suivant (`CoucherSoleil.html` disponible dans les fichiers du chapitre) rassemble tous ces éléments et crée un coucher de soleil.

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
CanvasMaster=new Object();
CanvasMaster.showCanvas=function()
{
canvasNow  = document.getElementById("sunset");
contextNow = canvasNow.getContext('2d');
sunsetGradient=contextNow.createLinearGradient(0, 0, 0,379);
sunsetGradient.addColorStop(0, "yellow");
sunsetGradient.addColorStop(1, "#cc0000")
contextNow.fillStyle = sunsetGradient;
contextNow.beginPath();
contextNow.arc(200,200,150,0,Math.PI*2,false);
contextNow.closePath();
contextNow.fill()
}
</script>
<style type="text/css">
body {
font-family:Verdana, Geneva, sans-serif;
color:#cc0000;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Coucher de soleil</title>
</head>
<body onLoad="CanvasMaster.showCanvas()">
<figure>
  <canvas id="sunset" width="400" height="400" > Vous ne pourrez pas admirer
ce magnifique coucher de soleil car vous n'avez pas de navigateur
HTML5</canvas>
  <figcaption>
    <p>Coucher de soleil</p>
  </figcaption>
</figure>
</body>
</html>
```

Quand on teste la page, on voit un grand cercle avec un dégradé qui vire du jaune au rouge. Vous pouvez aussi utiliser la même technique de dégradé avec d'autres formes. La figure 13.16 illustre le résultat.

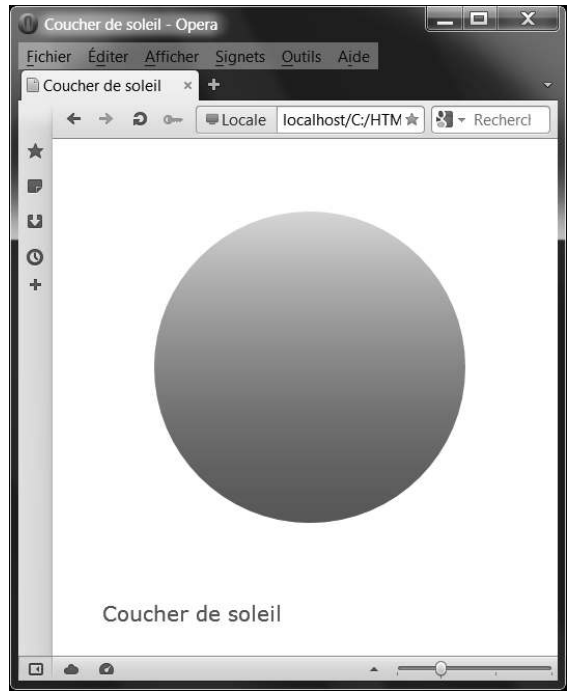


Figure 13.16 — Cercle avec un remplissage en dégradé.

On peut encore réaliser plein d'autres choses avec des canvas, et l'une des meilleures caractéristiques de ces images créées avec des objets du DOM par rapport aux images bitmap, c'est qu'elles consomment très peu de bande passante. Nous n'avons cependant que survolé l'étude de ce nouvel élément HTML5 qui est extrêmement puissant.

13.3 À VOUS DE JOUER !

Le travail avec `canvas` est tellement exaltant et tellement varié qu'il est difficile de savoir par où commencer. Je vous propose donc de tester les mini-projets suivants pour expérimenter ce nouvel élément en HTML5 :

- À la figure 13.13, on voit deux objets : une valise et une maison. Essayez de dessiner la maison en employant les méthodes utilisées pour la création de la valise.
- Prenez une image encadrée et superposez une autre image pour qu'elle apparaisse dans le cadre (ce projet nécessite de fixer les tailles du cadre et de l'image de telle sorte que l'image puisse rentrer dans le cadre).

- Trouvez ou créez une photo numérique et superposez un coucher de soleil au-dessus (vous pouvez aussi créer une image avec un autre type de dégradé et la superposer sur une photo numérique ou une autre image. Que diriez-vous d'un filtre en dégradé ?).

14

Ajout de formulaires

Objectif

L'une des fonctionnalités les plus importantes de toute page Web est sa capacité d'interaction avec une personne. Dans le jargon informatique, il y a une sous-catégorie appelée interface homme-machine qui traite les humains comme un autre type d'interface tel qu'une imprimante, une clé USB ou une webcam. Cela ne déshumanise pas les gens qui utilisent un ordinateur, mais au contraire cela traite les gens comme quelque chose qu'ils ne sont pas et ceci vous posera des problèmes à un moment ou un autre. Ce chapitre montre comment ajouter des formulaires et traiter les personnes comme des personnes.

14.1 AJOUT D'UN FORMULAIRE

Les formulaires se décomposent véritablement en deux parties (et parfois même plus dans certains cas). La première partie est la balise `<form>` qui définit un conteneur pour différentes sortes de saisies. Voici la forme que peut prendre un formulaire typique :

Début du formulaire

Champ de saisie n° 1

Champ de saisie n° 2

Champ de saisie n° 3

Champ de saisie n° 4

Fin du formulaire

Quand on étudie les formulaires, il faut vraiment distinguer le formulaire et ses attributs des champs de saisie et de leurs attributs. Avec les formulaires HTML5, vous allez trouver une quantité de nouveaux attributs et d'éléments.

Le programme suivant (`DegresEnRadians.html` disponible dans les fichiers du chapitre) est un exemple d'un simple convertisseur de degrés en radians (voir le chapitre 13 pour des exemples pratiques d'utilisation du convertisseur). Il suffit de saisir les degrés que vous voulez convertir et l'équivalent en radians s'affichera.

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
FormMaster=new Object();
FormMaster.resolveForm=function()
{
const RADCON=Math.PI/180;
degreesNow=document.converter.degrees.value;
radiansNow=degreesNow * RADCON;
document.converter.radians.value=radiansNow;
}
</script>
<style type="text/css">
/*048ABF,049DBF,F2F2F2,595959,0D0D0D */
h3 {
font-family:"Arial Black", Gadget, sans-serif;
color:#595959;
}
body {
font-family:Verdana, Geneva, sans-serif;
color:#049DBF;
background-color:#0D0D0D;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Convertisseur de degrés en radians</title>
</head>
<body >
<article>
  <header>
    <h3>Convertisseur de degrés en radians</h3>
  </header>
  <section>
    <form name=converter>
      Saisissez des degrés :<br>
      <input type=number name=degrees required >
      <br>
      Radians :<br>
      <input type=number name=radians>
      <br>
      <input type=submit name=submit value="Convertir en radians"
onClick="FormMaster.resolveForm()">
    </form>
  </section>
```

```
</article>
</body>
</html>
```

Si vous êtes déjà familier des formulaires HTML, vous savez que ce formulaire est différent car il possède une saisie numérique qui traite les entrées comme des nombres réels et non pas comme du texte qui doit être converti en nombres par JavaScript. Cette fonctionnalité n'était pas disponible dans les précédentes versions de HTML. La figure 14.1 illustre les contrôles « spinner » qui apparaissent dans Opera quand les pages Web utilisent une saisie numérique.

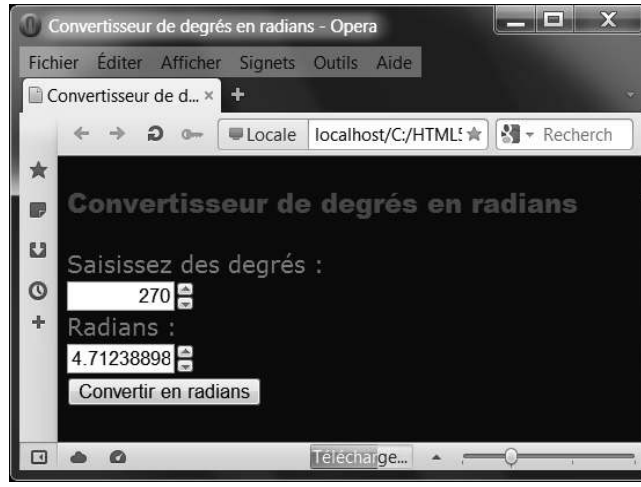


Figure 14.1 — Saisie de nombres pour des calculs et des conversions.

Comme vous pouvez le voir dans ce chapitre, il y a beaucoup de nouveautés et l'utilisation de JavaScript va vous permettre de tirer le meilleur parti des formulaires HTML5. Cela devrait vous inciter à mettre à jour tous vos navigateurs en HTML5.

14.1.1 Attributs généraux d'un formulaire

Un formulaire comporte plusieurs attributs qui ont un impact sur chacun des éléments de saisie du conteneur `form`, mais nous allons d'abord nous concentrer sur les attributs suivants du formulaire :

- `accept-charset`
- `action`
- `autocomplete`
- `enctype`
- `method`
- `name`
- `novalidate`

- `target`

La plupart de ces attributs sont rarement utilisés et certains n'ont de sens que lorsqu'on travaille avec des programmes comme PHP et ASP.NET où l'on communique avec une base de données. Nous allons cependant tous les examiner.

Accept-charset, enctype et novalidate

L'attribut `accept-charset`, s'il est spécifié, assigne en général la valeur `utf-8` comme encodage des caractères des données du formulaire. Cela signifie qu'il traite toutes les données saisies comme étant encodées en `utf-8`. Une instruction simple comme la suivante est suffisante :

```
<form name=monformulaire accept-charset=utf-8>
```

Si aucun encodage de caractères n'est spécifié, on suppose alors qu'il est inconnu et on utilise l'encodage par défaut. Quand on utilise plusieurs encodages, chaque encodage est séparé par un espace en HTML5, alors que dans les versions précédentes de HTML le séparateur était une virgule ou un point-virgule.

La plupart du temps, l'attribut `enctype` est laissé vide et il utilise le paramètre par défaut. L'attribut `enctype` possède trois mots-clés et trois états (mot-clé/état) :

- `application/x-www-form-urlencoded` (par défaut)
- `multipart/form-data`
- `text/plain`

Un formulaire peut être défini pour accepter le texte clair, ce qui donne l'assignation suivante :

```
<form enctype="text/plain">
```

Pour la plupart cependant, il s'agit d'un autre attribut qui n'est pas inclus dans la balise `<form>`. Cela est dû au fait que la valeur par défaut (`urlencoded`) est celle qui est privilégiée.

L'attribut `novalidate` est un booléen utilisé dans la validation du formulaire ; il bloque la validation des données saisies par l'utilisateur pendant leur soumission. Cela peut économiser du temps, mais cela peut aussi conduire à des cafouillages. Parfois un simple formulaire ou un formulaire ouvert (les données soumises sont inconnues) ne se valide pas parce qu'il y a des problèmes de validation qui sont inconnus. Si cet attribut est présent dans la balise `form`, les éléments soumis ne seront pas validés :

```
<form nonvalidate>
```

Cela bloque effectivement la validation des éléments soumis.

Les attributs booléens `formnovalidate` et `required` représentent une meilleure solution et ils peuvent être placés dans les éléments individuels de saisie. Par exemple,

le formulaire suivant n'a pas de validation pour un bouton d'annulation et le prénom n'est pas obligatoire bien que le nom le soit.

```
<form name=monformulaire accept-charset=utf-8>
  Prénom :
  <input type=text name=prenom >
  <br>
  Nom :
  <input type=text name=nom required>
  <br>
  <input type=submit name=submit value="Envoyer les informations ! ">
  <input type=submit formnovalidate name=cancel value="Annuler">
</form>
```

Vous n'utiliserez sans doute pas très souvent les attributs `accept-charset`, `enctype` et `novalidate`, mais les attributs de l'élément `input` pour gérer la non validation et la saisie des données obligatoires peuvent se révéler pratiques.

Autocomplete

`autocomplete` est un attribut de formulaire assez simple, mais qui est important. Il possède deux états, `on` et `off`, et sa valeur par défaut est `on`. Fondamentalement, si vous ne voulez pas activer la saisie semi-automatique, il suffit simplement de lui attribuer la valeur `off`. La saisie semi-automatique peut parfois être gênante et si tel est le cas, il suffit d'ajouter la ligne suivante :

```
<form autocomplete="off">
```

Quand cet attribut est à off, les mots qui sont réutilisés ne s'afficheront pas. Par exemple, si vous avez changé d'adresse électronique, votre ancienne adresse peut s'afficher automatiquement dans les champs de saisie de l'adresse électronique si l'attribut `autocomplete` n'est pas défini à `off`.

Name et target

L'attribut `name` est l'un des attributs les plus importants du formulaire car il est utilisé par le DOM pour l'identifier. En tant que propriété de l'objet document, il peut être référencé soit comme élément de tableau (par exemple `forms[0]`), soit par son nom. D'un point de vue pratique, il est plus facile de faire référence à un formulaire et à ses sous-formulaires par leur nom.

En plus de l'attribut `name`, les formulaires ont un attribut global, `id`. Les deux attributs ont des noms. Dans le DOM, la référence se fait sur l'attribut `name`, mais à l'intérieur d'un document Web unique (une page), d'autres éléments peuvent identifier le formulaire grâce à une référence à l'ID du formulaire. Il y a de plus une nouvelle fonctionnalité de HTML5 qui permet au formulaire enfant d'exister en dehors du conteneur `<form>` et d'avoir un attribut de formulaire qui le relie à n'importe quel formulaire de la page. Par exemple, l'élément de saisie de texte suivant fait partie du formulaire dont l'`id` est `ralph`.

```
<input type=text form=ralph  name=maison>
```

L'élément de saisie de texte peut se situer n'importe où sur la page, ce qui signifie que les concepteurs n'ont pas à mettre tous les champs de saisie en un même lieu. Testez le script suivant (`FormID.html` disponible dans les fichiers du chapitre) avec le navigateur Opera (qui a implémenté cette nouvelle fonctionnalité).

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
FormMaster=new Object();
FormMaster.resolveForm=function()
{
favorite = document.formName.favURL.value;
personName=document.formName.person.value;
message= "Le site Web favori de " + personName + " est " +favorite;
document.formName.output.value=message;
}
</script>
<style type="text/css">
h3 {
font-family:"Arial Black", Gadget, sans-serif;
color:#97CCA6;
}
body {
font-family:Verdana, Geneva, sans-serif;
color:#EFF09E;
background-color:#AB1F33;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Champs de saisie distants</title>
</head>
<body >
<article>
  <header>
    <h3>ID à connecter</h3>
  </header>
  <section> Quel est votre site Web favori ?<br>
    <label>Site favori :
      <input type=url form=formID name=favURL>
    </label>
  </section>
  <section>
    <blockquote> Cette section représente une coupure entre le premier champ
de saisie (qui nécessite une URL) et le reste du formulaire auquel le
formulaire d'URL appartient. Cela laisse plus de latitude aux concepteurs dans
la gestion d'un site interactif. </blockquote>
  </section>
  <section>
    <form name=formName id=formID>
      <label>Quel est votre nom ?
        <input type=text name=person>
      </label>
```

```
      <br>
      Résultat :<br>
      <textarea name=output cols=50 rows=5></textarea>
      <br>
      <input type=submit name=submit value="On rassemble les morceaux"
onClick="FormMaster.resolveForm()">
    </form>
  </section>
</article>
</body>
</html>
```

Vous noterez qu'à l'intérieur du conteneur `<form>` avec les attributs `name=formName` et `id=formID` il y a un seul élément de saisie, une balise `<textarea>` et un bouton de soumission. Plus important encore, vous remarquerez que l'élément de saisie ayant l'attribut `name=favURL` se situe en dehors du conteneur `<form>`, mais il s'assigne à lui-même l'ID du formulaire sur la page, `formID`. En HTML5, il est traité comme s'il était à l'intérieur du conteneur `<form>`. La figure 14.2 montre que les données saisies dans l'élément de saisie de type `url` (`name=favURL`) sont sélectionnées par le DOM du code JavaScript comme faisant partie du même formulaire que le reste des éléments de saisie du formulaire appartenant au formulaire nommé `formName`.

Figure 14.2 — La saisie peut être placée en dehors du conteneur du formulaire.

Vous n'avez donc plus à présent à vous préoccuper de l'emplacement où vous mettez vos formulaires de saisie. Tant que l'on assigne aux éléments de saisie l'ID du formulaire, ils sont traités comme s'ils étaient à l'intérieur du conteneur du formulaire.

L'attribut `target` fait référence au contexte de navigation du formulaire après la soumission du formulaire. Si aucune valeur n'est assignée à l'attribut `target`, le contexte de navigation est le même que si la valeur `_self` est assignée à l'attribut `target`. Les autres contextes de navigation sont `_blank`, `_parent` ou `_top`. Le contexte de navigation `_blank` est assez utile quand on récupère des informations d'un script côté serveur qui remplacent le contenu de la page appelante par son propre contenu. Le fait d'utiliser `_blank` permet aux utilisateurs de voir à la fois la page appelante et les informations de la page appelée.

14.1.2 Le formulaire comme partie du DOM

Bien que le DOM soit généralement abordé comme une organisation de nœuds, il peut aussi être décrit en termes d'objets (après tout, il y a bien le mot objet en plein au milieu du DOM !). Afin de voir la manière dont les formulaires et les champs de saisie sont organisés dans le DOM, vous pouvez utiliser les références JavaScript aux différentes parties d'un formulaire. Le DOM référence les éléments du formulaire sous la forme d'un tableau au sein d'un document. Les éléments de saisie liés au formulaire sont les éléments de tableau du formulaire, le premier nœud s'appelant `elements[0]` (on est dans un système de numérotation où le premier élément commence à zéro). De la même manière, la numérotation des formulaires commence à zéro et le premier formulaire du document se nomme `forms[0]` (Note : les éléments et les formulaires sont au pluriel, même si le nom des balises `<element>` et `<form>` est au singulier).

Pour vous aider à distinguer les parties d'une organisation DOM, le simple script suivant (`IDNom.html` disponible dans les fichiers du chapitre) montre les différentes manières de référencer le même objet dans un document contenant des formulaires. Il est préférable de référencer un formulaire par son nom d'objet et de propriété et les différentes combinaisons ne sont présentes ici qu'à des fins de démonstration. Ce programme utilise également plusieurs types de saisie.

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
FormMaster=new Object();
FormMaster.resolveForm=function()
{
      alpha = document.motherShip.elements[0].value;
      beta = document.forms[0].secondInput.value;
      gamma = document.motherShip.thirdInput.value;
      delta = document.forms[0].elements[3].value;
      epsilon = document.motherShip.fifthInput.value;
      const cr="\n";
      message=alpha+cr+beta+cr+gamma+cr+delta+cr+epsilon;
      document.motherShip.output.value=message;
```

```
}
</script>
<style type="text/css">
h3 {
     font-family:"Arial Black", Gadget, sans-serif;
     color:#677E52;
}
body {
     font-family:Verdana, Geneva, sans-serif;
     color:#89725B;
     background-color:#B0CC99;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>DOM et formulaires</title>
</head>
<body >
<article>
  <header>
    <h3>DOM, le formulaire et les noeuds</h3>
  </header>
  <form name=motherShip>
    <input type=number name=firstInput>
    Numéro<br>
    <input type=email name=secondInput>
    Courriel<br>
    <input type=text name=thirdInput>
    Texte<br>
    <input type=text name=fourthInput>
    Texte<br>
    <input type=url name=fifthInput>
    URL<br>
    <textarea cols="15" rows="6" name=output></textarea>
    <input type=button value="Envoyer au DOM"
onClick="FormMaster.resolveForm()">
  </form>
  </section>
</article>
</body>
</html>
```

Quand on teste le programme, on saisit le texte et les nombres appropriés, puis on clique sur le bouton Envoyer au DOM. Dans le programme JavaScript, vous noterez que tant que les noms des éléments ou les noms des nœuds sont utilisés, les informations saisies sont envoyées dans la zone de texte qui est utilisée comme fenêtre de sortie. La figure 14.3 illustre le résultat attendu.

Les contenus sont récupérés via les chemins du DOM et placés dans des variables puis envoyés à l'élément `<textarea>` pour être affichés. Une constante (`const cr="\n"`) est insérée entre les cinq éléments pour placer un caractère de contrôle qui force un retour à la ligne.

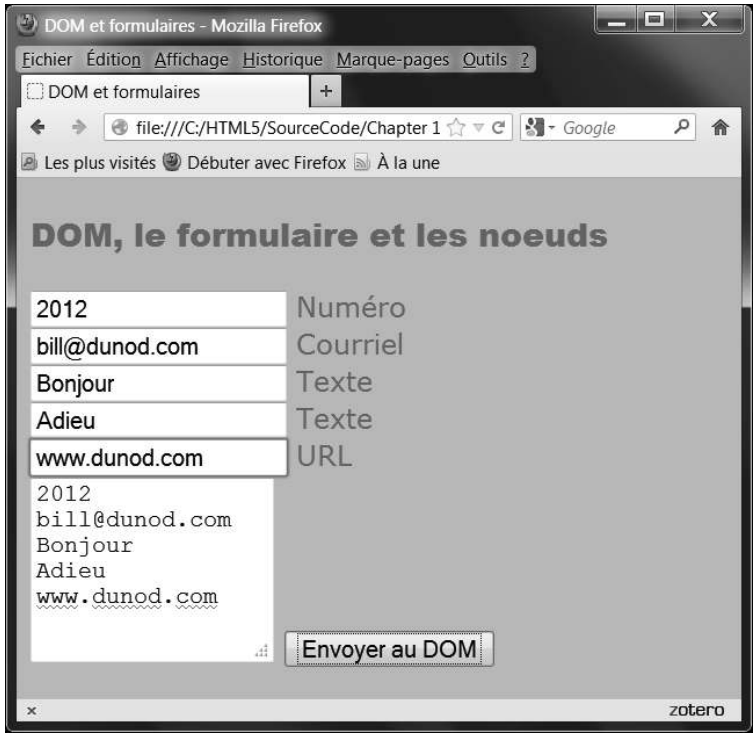


Figure 14.3 — Saisies de l'utilisateur affichées sur la page.

14.2 LES DIFFÉRENTES SORTES DE SAISIE

L'une des principales nouvelles fonctionnalités de HTML5 est l'ajout de différents types d'attributs de saisie. De plus, ces différents attributs de saisie fonctionnent avec les périphériques mobiles. Par exemple, si l'on utilise un type de saisie `email` ou `url`, un clavier spécial avec un point (`.`) et le suffixe `.com` apparaît quand on commence à saisir des données dans un formulaire sur certains terminaux mobiles.

Avec les nouveaux types de saisie, il y a des attributs supplémentaires qui modifient la manière dont la page interagit avec les utilisateurs. Sur les 29 attributs de saisie, 11 sont nouveaux en HTML5. Comme pour les nouveaux types de saisie, il est nécessaire d'apprendre à utiliser ces attributs. Comme il y a de nombreux types de saisie et d'attributs, je les ai rassemblés en deux tableaux. Le tableau 14.1 liste les valeurs des différents types que l'on peut utiliser et le tableau 14.2 liste tous les attributs.

Au cours de l'écriture de ce livre, tous ces types n'avaient pas été implémentés par les principaux navigateurs, mais comme il est prévu à terme une implémentation complète de la nouvelle norme HTML5, n'ayez pas peur de tester de votre côté les différents types. Le tableau 14.2 liste les attributs généraux des différents éléments de saisie.

Tableau 14.1 — Types des éléments de saisie HTML5. *Nouveau en HTML5.

Type	Fonctionnalité	Type	Fonctionnalité
`button`	Bouton de commande	`checkbox`	Sélection
`color`	Palette de couleurs	`date`*	Sélectionneur de date
`datetime`*	Sélectionneur de date	`datetime-local`*	Sélectionneur de date
`email`	Adresse électronique	`file`	Envoi de fichier
`hidden`	Masqué	`image`	Coordonnées d'image
`month`*	Sélectionneur de date	`number`*	Valeur numérique
`password`	Masque le mot de passe	`radio`	Sélection
`range`*	Plage de nombres	`reset`	Efface les entrées
`search`*	Mots de recherche	`submit`	Envoie les données du formulaire
`tel`	Numéro de téléphone	`text`	Valeur string
`time`*	Sélectionneur de date	`url`*	Adresse Web
`week`*	Sélectionneur de date		

Les prochaines sections étudient les différents regroupements d'éléments et d'attributs de formulaires étant donné le grand nombre de combinaisons possibles. La première section couvre l'utilisation de l'élément `datalist` avec `list` et les attributs de `form`. Comme pour toutes les sections suivantes, celle-ci rassemble un grand nombre de fonctionnalités tout en se concentrant sur les fonctionnalités importantes qui sont l'objet de la discussion.

14.2.1 L'attribut list, le type URL et les datalist

L'attribut `list` est l'un des nouveaux attributs qui peut être utilisé avec des formulaires. Au début de ce chapitre, j'ai insisté sur le fait que les applications Web devaient être

Tableau 14.2 — Attributs des éléments de saisie. *Nouveau en HTML5.

Attribut	Fonctionnalité	Attribut	Fonctionnalité
`accept`	Type de fichier accepté	`alt`	Indice de chargement de fichier
`autocomplete`*	Saisie semi-automatique	`autofocus`*	Place le focus sur un champ
`checked`	État sélectionné	`disabled`	Inutilisable
`form`*	Définit l'ID du formulaire	`formaction`*	Substitution
`formenctype`*	Substitution	`formmethod`*	Substitution
`formnovalidate`*	Substitution	`formtarget`*	Substitution
`height`*	Hauteur en pixels	`list`*	Suggestion de liste de données
`max`*	Valeur maximale	`maxlength`	Longueur maximale
`min`*	Valeur minimale	`multiple`	Valeurs multiples
`name`	Nom DOM	`pattern`*	Expression régulière
`placeholder`*	Disparaît à la saisie	`readonly`	Saisie impossible
`required`*	Champ obligatoire	`size`	Nombre de caractères visibles
`src`	Source	`step`*	Nombre d'étapes
`type`	Type de saisie	`value`	Valeurs assignées
`width`*	Largeur en pixels		

conviviales et interactives. Il est donc préférable de minimiser le travail de saisie de l'utilisateur dans un formulaire. L'attribut `list` fournit une liste de suggestions dans un élément de saisie et les utilisateurs peuvent sélectionner à partir de la liste ou bien saisir une réponse. L'attribut `list` est en fait une référence à une balise `<datalist>` qui est située ailleurs sur la page Web. De plus, si l'on place le conteneur `<datalist>` au sein d'un conteneur `<form>`, les éléments `<input>` après la liste de données ne s'affichent pas sur la page. Dans ces conditions, il faut fournir un attribut ID dans la balise `<datalist>` et l'assigner à l'attribut `list` de la balise `<input>`. La liste de données

est stockée en dehors du conteneur `<form>`, mais elle est connectée via l'ID de la liste de données.

Affichage d'un choix dans une fenêtre de message

Dans cet exemple simple, on définit un champ de saisie où l'utilisateur tape ou sélectionne une URL. Une fois que l'URL est entrée, l'utilisateur clique sur le bouton Afficher votre URL et une boîte de dialogue affiche l'adresse. Le script suivant (`DataList.html` disponible dans les fichiers du chapitre) montre comment assembler toutes les pièces.

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
FormMaster=new Object();
FormMaster.resolveForm=function()
{
place=document.traveler.getURL.value;
alert(place);
}
</script>
<style type="text/css">
h3 {
font-family:"Arial Black", Gadget, sans-serif;
color:#B9121B;
}
body {
font-family:Verdana, Geneva, sans-serif;
color:#4C1B1B;
background-color:#FCFAE1;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>List et Datalist</title>
</head>
<body >
<article>
  <header>
    <h3>List et Datalist</h3>
  </header>
  <section>
    <datalist id=favoriteSites>
      <option value="http://www.smashingmagazine.com/" label="Smashing">
      <option value="http://www.sandlight.com/" label="Sandlight">
    </datalist>
  </section>
  <section>
    <form name=traveler>
      <label>Saisissez l'un de vos sites favoris :</label>
      <br>
      <input type=url  list=favoriteSites name=getURL>
      <br>
      <input type=submit value="Afficher votre URL"
onClick="FormMaster.resolveForm()">
```

```
    </form>
  </section>
</article>
</body>
</html>
```

En examinant le script, vous vous demandez peut-être ce que l'attribut `label` fait dans la balise `<option>` du conteneur `<datalist>`. Il n'y a pas d'attribut `label` dans les éléments `form` ou `input` (voir les tableaux 14.1 et 14.2) car cet attribut se trouve dans la balise `<option>`. Bien que cela paraisse évident, quand on ouvre la page, on voit non seulement les URL, mais aussi le libellé dans la fenêtre de saisie des URL. Cela s'explique car la balise `<input type=url>` contient une référence aux options de la liste de données via l'attribut `list` du balisage de l'élément `input`.

Pendant la rédaction de ce livre, la liste de données ne s'affichait que sous Opera (Windows 7 ou Mac OS X).

Dans le champ de saisie, on peut voir les sélections disponibles dans la liste de données (il y a aussi un libellé pour chaque adresse). Quand l'utilisateur fait une sélection, elle apparaît dans le champ de saisie. Enfin, le résultat de la saisie est passé à la fonction JavaScript qui l'affiche dans une boîte de dialogue (vous noterez que la boîte de dialogue du navigateur Opera affiche aussi le domaine).

Le point important de ce processus est que les utilisateurs n'ont pas à saisir les URL. Toute personne qui a déjà saisi une adresse de site Web a commis une faute de frappe à un moment donné. La liste de données oriente le choix de l'utilisateur et facilite sa saisie.

Élément de Datalist sur périphérique mobile et clavier pour URL

Mes tests de l'application avec le navigateur Mini Opera sur iPhone ont révélé que la liste de données n'apparaissait pas. D'autres tests avec la version mobile de Safari ont montré que cela ne marchait pas non plus sous Safari.

Cependant, au cours de ces tests, un clavier unique pour les nouveaux types d'éléments de saisie `url` et `email` est apparu. Le navigateur mobile Safari reconnaît les champs de saisie typés `url` et `email` et, quand ils sont utilisés, il affiche un clavier qui inclut un point (`.`) et le suffixe `.com`, plus quelques autres touches qui sont couramment employées pour la saisie des URL et des adresses électroniques. La figure 14.4 illustre les navigateurs Safari mobile (gauche) et Mini Opera (droite) côte à côte affichant le programme de liste de données sur le même iPhone. Si vous regardez soigneusement, vous pourrez voir les différences.

Un navigateur mobile qui reconnaît que la saisie attend une URL ou une adresse électronique est important car cela signifie qu'il prend en considération l'utilisateur. Avec ce clavier spécial, les utilisateurs n'ont pas à jongler autant entre les claviers alphabétique et numérique.

Figure 14.4 — Clavier spécifique pour la saisie d'une URL sur un périphérique mobile (gauche) et clavier mobile standard (droite).

14.2.2 Boutons radio et cases à cocher : éléments de saisie faciles à sélectionner

Si vous utilisez des boutons radio et des cases à cocher avec des programmes externes qui accèdent à des bases de données ou qui réalisent d'autres types d'opération côté serveur, cela est très simple du côté HTML5. Il suffit d'utiliser un bouton qui envoie les données et tout est transféré au serveur pour que ces programmes gèrent les données.

En raison du fait que le script suivant renvoie les données saisies à un objet `<textarea>` de la page, ces données doivent être contrôlées en utilisant JavaScript avec une petite boucle pour voir d'abord si l'attribut `checked` est égal à `true` ou `false`. Si l'élément est sélectionné, le script ajoute ensuite la valeur à une propriété `FormMaster` nommée `this.countVal` (c'est comme une variable, mais on garde le style de programmation DOM et on assigne la valeur à un objet). Quand c'est terminé, seules les valeurs sélectionnées sont envoyées dans la fenêtre de résultat. Le script suivant (qui est un peu long) (`BoutonsCases.html` disponible dans les fichiers du chapitre) réalise toutes ces tâches.

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
FormMaster=new Object();
FormMaster.resolveForm=function()
{
this.countVal="";
this.topCount=document.checkRadio.length-2;
```

```
for(var count=0;count < this.topCount;count++)
if(document.checkRadio.elements[count].checked)
{
this.countVal+=document.checkRadio.elements[count].value+"\n";
}
document.checkRadio.outNow.value=this.countVal;
}
</script>
<style type="text/css">
/* 735840,733119,BF5D39,352D1B,C0B787 */
body {
background-color:#C0B787;
color:#733119;
font-family:Verdana, Geneva, sans-serif;
font-size:12px;
margin-left:20px;
}
h1 {
font-family:"Arial Black", Gadget, sans-serif;
color:#733119;
}
h2 {
color:#BF5D39;
}
h3 {
color:#BF5D39;
}
#dataEntry {
display:table;
}
#lang {
display:table-cell;
width:200px;
}
#out {
display:table-cell;
width:300px;
}
aside {
display:table-cell;
width:250px;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Cliquez pour choisir</title>
</head>
<body>
<article>
  <header>
    <h1>Sélections</h1>
  </header>
  <div id="dataEntry">
  <div id="lang">
  <section>
  <h2>Langages Web</h2>
  <form name=checkRadio>
    <label>
```

```
      <input type=checkbox name=html value="HTML5" checked>
      HTML5</label><br>
    <label>
      <input type=checkbox name=css value="CSS3">
      CSS3</label><br>
    <label>
      <input type=checkbox name=js value="JavaScript">
      JavaScript</label><br>
    <label>
      <input type=checkbox name=php value="PHP">
      PHP5</label><br>
    <label>
      <input type=checkbox name=asp value="ASP.NET">
      ASP.NET</label><br>
    <label>
      <input type=checkbox name=action value="ActionScript 3.0">
      ActionScript 3.0</label>
    </section>
    </div>
    <section>
      <aside>
        <h2>Spécialités</h2>
        <fieldset>
          <legend> Web </legend>
          <label>
            <input type=radio name=work value="Conception graphique">
            Conception </label><br>
          <label>
            <input type=radio name=work value="Conception d'interface">
            Conception d'interface </label><br>
          <label>
            <input type=radio name=work value="Frontal">
            Développement de frontal </label><br>
          <label>
            <input type=radio name=work value="Back-end">
            Développement de back-end</label><br>
        </fieldset>
      </aside>
    </section>
    </div>
    <section>
      <div id="out">
        <fieldset>
          <legend>Fenêtre de résultat</legend>
          <textarea name=outNow rows=10 cols=40 ></textarea>
        </fieldset>
      </div>
    </section>
    <section>
      <div>
        <p>
          <input type=button name=getEm value="Transmission des sélections"
onClick="FormMaster.resolveForm()">
        </p>
      </div>
    </section>
  </form>
```

```
  </div>
</article>
</body>
</html>
```

Bien que ce programme soit un peu long, la majeure partie est du code de mise en forme si bien qu'il est de taille convenable et facilite la tâche des utilisateurs. La balise `<fieldset>` a été utilisée pour mettre en évidence un groupe de boutons et pour encapsuler la fenêtre de résultat. C'est une balise essentielle quand on veut regrouper des éléments. La balise `<legend>` permet de placer une étiquette dans le rectangle qui encadre l'ensemble des champs. La figure 14.5 illustre le résultat attendu.

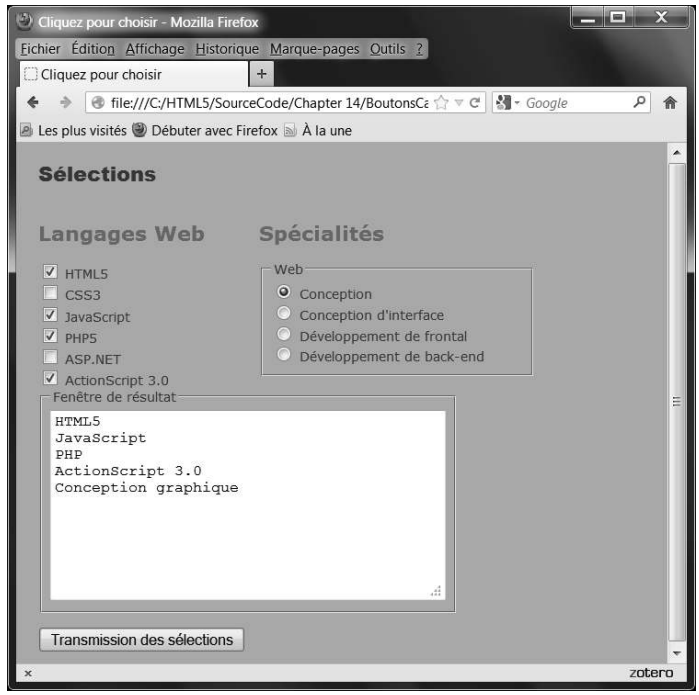


Figure 14.5 — Cases à cocher et bouton radio.

Quand on exécute le programme pour la première fois, on voit que la case à cocher HTML5 a déjà été sélectionnée. Cela est dû au fait que l'attribut `checked` a été ajouté à la balise. Il s'agit d'un booléen, mais on n'a pas à lui assigner la valeur `true` ou `false`. Après le chargement de la page, examinez ce qui se passe quand vous cliquez dessus.

La fenêtre de résultat abrite toutes les valeurs qui ont été assignées aux cases à cocher et aux boutons radio. Dans une véritable application, ces données seraient envoyées dans une base de données.

14.2.3 Sélectionneur de date

Nous terminerons ce chapitre par un dernier attribut de saisie qui est simple à implémenter, mais qui procure des résultats impressionnants. Le nouvel attribut `date` de l'élément `input` est puissant et facile à inclure dans un formulaire. Plusieurs nouveaux attributs de date et d'heure ont été ajoutés à l'élément input, mais seul l'attribut date est étudié ici. Le programme suivant (`Calendrier.html` disponible dans les fichiers du chapitre) montre comment utiliser cet attribut et récupérer l'information.

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
FormMaster=new Object();
FormMaster.resolveForm=function()
{
alert(document.calendar.dateNow.value);
}
</script>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Date</title>
</head>
<body >
<form name=calendar>
  <input name=dateNow type="date" onChange="FormMaster.resolveForm()">
</form>
</body>
</html>
```

Avec simplement cette petite balise dans le conteneur `form`, vous êtes capable de créer un programme de calendrier complet. Vous pouvez utiliser le gestionnaire d'événements `onChange` pour capturer la date sélectionnée dans le calendrier. La figure 14.6 illustre l'application dans le navigateur Opera (le seul à présent qui fonctionne avec ce nouvel attribut de saisie) sous Windows 7.

Dans cette implémentation particulière, dès que l'utilisateur fait son choix, une boîte de dialogue s'ouvre et affiche la date sélectionnée, comme cela est illustré à la figure 14.7.

Le but est ici de montrer comme il est facile de passer la valeur de la date sélectionnée. De telles données pourraient être stockées dans une base de données pour un système de réservation en ligne.

La petite fenêtre derrière la boîte de dialogue montre la date sélectionnée sans qu'il y ait besoin de beaucoup de programmation. Ce nouvel attribut va vraiment faciliter la vie des utilisateurs quand ils veulent choisir une date. Si vous avez déjà expérimenté ce genre de système pour réserver un avion ou un hôtel, vous savez comme cet outil est appréciable. Le problème est que pour le moment seul le navigateur Opera l'a implémenté.

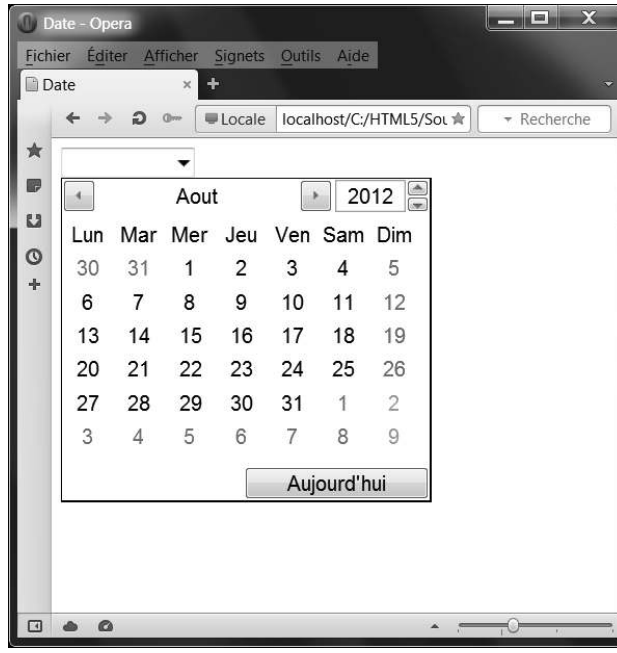


Figure 14.6 — Une simple balise fournit un calendrier en ligne.

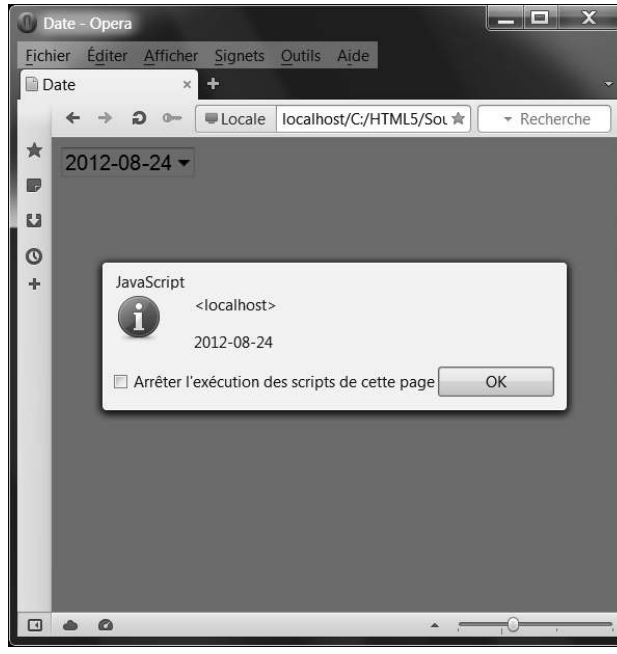


Figure 14.7 — Passage de la valeur de la date à JavaScript.

14.3 À VOUS DE JOUER !

Le plat de résistance de ce chapitre a été la manière d'utiliser le DOM pour accéder aux informations d'un formulaire, grâce à la syntaxe de base suivante :

```
document.formulaire.élément.valeur
```

Vous devez utiliser JavaScript pour accéder aux données qui sont en général passées à un programme côté serveur comme PHP, ColdFusion ou ASP.NET. Pour simuler ce comportement, les exemples de ce chapitre ont utilisé un bouton de commande qui déclenchait un programme JavaScript afin d'envoyer les résultats dans une balise `<textarea>` où l'on pouvait voir ce qui aurait été normalement envoyé à une base de données. Voici les défis que vous aurez à relever :

- Imaginez un magasin en ligne qui vend une gamme de produits (au moins cinq) ou fournit des services (au moins cinq), par exemple un magasin d'informatique ou bien un service de conception de sites Web.
- Concevez une interface où les utilisateurs entrent leur nom, leur adresse électronique, une URL, leur adresse complète, un nom d'utilisateur et un mot de passe. La saisie doit être la plus simple possible pour l'utilisateur. Faites tester le formulaire par quelqu'un qui ne l'a jamais vu auparavant.
- Les utilisateurs sélectionnent ensuite plusieurs produits ou services.
- Les sélections de l'utilisateur sont ensuite affichées dans une balise `<textarea>` avec leur prix individuel HT et TTC.
- Le programme génère aussi une étiquette d'expédition qui se contentera d'être affichée dans un objet `<textarea>` et qui ne sera donc pas imprimée.

Essayez d'utiliser le maximum d'éléments de saisie et d'attributs qui ont été étudiés dans ce chapitre.

15

Incorporation d'objets et stockage d'informations

Objectif

Pendant des années, les utilisateurs ont été capables de réaliser des choses remarquables sur le Web grâce à différentes sortes de plug-ins chargés dans le navigateur. D'une manière générale, deux plug-ins principaux sont installés avec la plupart des navigateurs : Adobe Flash Player et Java.

Certaines des nouvelles fonctionnalités de HTML5 marchent mieux avec des plug-ins spéciaux ou via une URL qui exploite la nouvelle fonctionnalité. Parmi ces nouveaux objets HTML5, l'objet `geolocation` permet de réaliser des choses très intéressantes et c'est celui que nous étudierons en premier. Ensuite, nous examinerons comment Flash fonctionne avec HTML5.

15.1 GEOLOCATION

L'objet `geolocation` fait partie de l'objet `navigation` du DOM HTML5. Il permet notamment de trouver plus ou moins précisément l'emplacement où vous vous trouvez. Les attributs les plus importants de l'objet `geolocation` sont `latitude` et `longitude`. Grâce à ces valeurs, vous pouvez charger une carte d'un emplacement si vous connaissez ses coordonnées.

La création d'une page HTML qui montre aux utilisateurs leur latitude et leur longitude est parfaite, mais les navigateurs HTML5 sont aussi capables de charger une carte sur un site Web avec Google Maps. Voici l'URL qui permet de réaliser cela :

```
" http://maps.google.com/maps?hl=en&ie=UTF8&ll= " + latitude + ", " +
longitude + "&spn=0.054166,0.110378&z=13&output=embed"
```

Les variables `latitude` et `longitude` contiennent les valeurs des coordonnées. L'astuce consiste donc à trouver les valeurs de latitude et de longitude pour les insérer au bon endroit.

15.1.1 Trouver la latitude et la longitude

Que ce soit sur votre téléphone mobile ou votre ordinateur, l'obtention de ces valeurs nécessite du code JavaScript dont voici la syntaxe de base :

```
navigator.geolocation.getCurrentPosition(uneFonction);
```

Pour filtrer les navigateurs qui ne reconnaissent pas l'objet `geolocation`, il suffit d'utiliser la technique suivante :

```
if (navigator.geolocation)
{
navigator.geolocation.getCurrentPosition(uneFonction);
}
else
{
        alert("Geolocation n'est pas reconnu !")
}
```

Cela permet d'avertir les utilisateurs que leur navigateur ne prend pas en charge l'objet `geolocation`.

La fonction appelée pour obtenir les informations de positionnement doit inclure un paramètre qui stocke ces informations. En utilisant des objets et des méthodes, l'appel de fonction est réalisé de la manière suivante :

```
...
navigator.geolocation.getCurrentPosition(LocationMaster.lookUpPosition);
...
```

Ceci récupère la méthode qui retourne les valeurs demandées :

```
LocationMaster=new Object();
LocationMaster.lookUpPosition=function(position)
{
     this.latNow=position.coords.latitude;
     this.longNow=position.coords.longitude;
...
```

Vous remarquerez que le paramètre `position` est similaire à une variable qui stocke les valeurs de latitude et de longitude. Il ne s'agit pas d'une propriété de l'objet geolocation (`cords.latitude` et `cords.longitude` sont les propriétés ; c'est moi qui ai

choisi le nom « position » car je le trouve plus descriptif que « NainTracassin », par exemple, mais vous pouvez utiliser le nom que vous voulez).

Une fois que les valeurs sont assignées au paramètre de l'objet, elles font partie de l'objet `LocationMaster` grâce à l'emploi du mot-clé `this`. Les propriétés `latNow` et `longNow` stockent les valeurs comme s'il s'agissait de variables, la seule différence étant qu'elles font partie de l'objet.

15.1.2 Récupération de la carte

La page HTML5 qui travaille avec JavaScript ne réalise qu'une seule chose : récupérer les coordonnées. L'obtention de la carte par la suite n'est qu'une simple affaire d'insertion de ces valeurs dans la requête de carte. Ainsi, pour terminer la méthode, le programme utilise la ligne suivante :

```
document.getElementById("mapHolder").src =
"http://maps.google.com/maps?hl=en&ie=UTF8&ll=" + this.latNow + "," +
this.longNow + "&spn=0.054166,0.110378&z=13&output=embed";
```

Vous trouverez une nouvelle méthode dans le cœur du DOM HTML5 : `getElementById`. Dans cet exemple, l'ID est celui d'un élément `iFrame`, la carte étant l'objet source, tout comme une image est chargée grâce à l'identification de sa source :

```
<img src="monImage.jpg">
```

La seule différence est que l'endroit où la carte est chargée est spécifié par l'ID de l'`iFrame` et non pas par la page par défaut.

Placement de la carte sur la page Web

Tout autre chargement après que la page ait été chargée ne peut pas être déposé n'importe où sur la page. La balise `<iframe>` peut être un emplacement cible situé en dehors du document principal. Le fait d'utiliser `<iframe>` sans attribut produit une fenêtre d'affichage relativement petite, mais l'idée est de montrer qu'avec quelques balises et un peu de code JavaScript j'arrive à afficher une carte sur la page.

On rassemble le tout sur une page simple

J'ai testé tous les grands navigateurs sous Windows 7 et Macintosh OS X, et le programme suivant (`MiniGeolocalisation.html` disponible dans les fichiers du chapitre) représente un bon point de départ d'une page qui affiche une carte proche de l'endroit où se trouve l'utilisateur.

```
<!DOCTYPE html >
<html>
    <head>
<style type="text/css">
/* BF7F6C,FFDDAE,B59D7B,40372B,E6C79C */
```

```
body {
font-family:Verdana, Geneva, sans-serif;
color:#40372B;
background-color:#FFDDAE;
}
h3 {
font-family:Tahoma, Geneva, sans-serif;
color:#BF7F6C;
}
</style>
<script>
LocationMaster=new Object();
LocationMaster.lookUpPosition=function(position)
{
this.latNow=position.coords.latitude;
this.longNow=position.coords.longitude;
document.getElementById("mapHolder").src =
"http://maps.google.com/maps?hl=en&ie=UTF8&ll=" + this.latNow + "," +
this.longNow + "&spn=0.054166,0.110378&z=13&output=embed";
}
        if (navigator.geolocation)
{

navigator.geolocation.getCurrentPosition(LocationMaster.lookUpPosition);
        }
else
{
            alert("Essayez un navigateur HTML5 différent car celui-ci ne
fonctionne pas avec l'objet geolocation.");
        }
    </script>
<title>Carte minimale</title>
    </head>
    <body>
    <article>
      <header>
        <h3>Votre emplacement</h3>
      </header>
      <section>
        <iframe id="mapHolder"> </iframe>
      </section>
      <section>
        <p> Cet exemple d'utilisation de la géolocalisation et de Google Maps
est très simple. Il a été testé avec les principaux navigateurs et les
navigateurs mobiles. </p>
      </section>
    </article>
</body>
</html>
```

Quand vous testerez cette page Web, prenez d'abord la dernière version du navigateur Firefox. Essayez ensuite avec Google Chrome et Opera. Avec Safari, qui reconnaît l'objet `geolocation`, j'ai été incapable de charger la carte dans l'objet `iframe`. Ironiquement, quand j'ai testé ce code sur le navigateur Safari mobile de l'iPhone, cela fonctionnait parfaitement. La figure 15.1 illustre le programme avec Internet Explorer sous Windows 7.

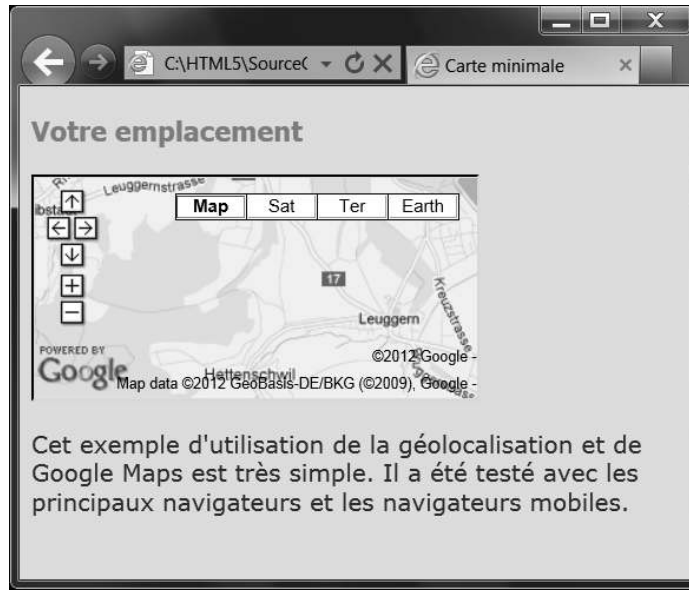


Figure 15.1 — Outil geolocation utilisé pour trouver la longitude et la latitude avec Google Maps.

Vous pouvez déplacer la carte dans l'`iframe` avec la souris et, sur les navigateurs Safari et Perfect sur un iPhone, avec les doigts. Cependant, sur les navigateurs mobiles, l'`iframe` et l'image sont agrandies par un glissement vers le bas.

Adaptation de la page pour un terminal mobile

Pour rendre la page plus pratique pour les utilisateurs mobiles, j'ai procédé à quelques ajustements afin de changer l'orientation de la carte en modifiant l'`<iframe>` de la manière suivante :

```
<section>
<iframe id="mapHolder" width="240" height="320"> </iframe>
</section>
```

On a à présent une orientation verticale et la carte est plus facile à lire. La figure 15.2 illustre le programme sur un iPhone avec les navigateurs Perfect (gauche) et Safari (droite). En bas de la page, il y a des instructions pour agrandir l'image sans déplacer la carte en dehors de l'`iframe`.

La figure 15.2 illustre cela en tirant la page vers l'extérieur et loin de la carte (copie d'écran de gauche) ; les utilisateurs mobiles peuvent agrandir la carte de manière à pouvoir la lire facilement (copie d'écran de droite).

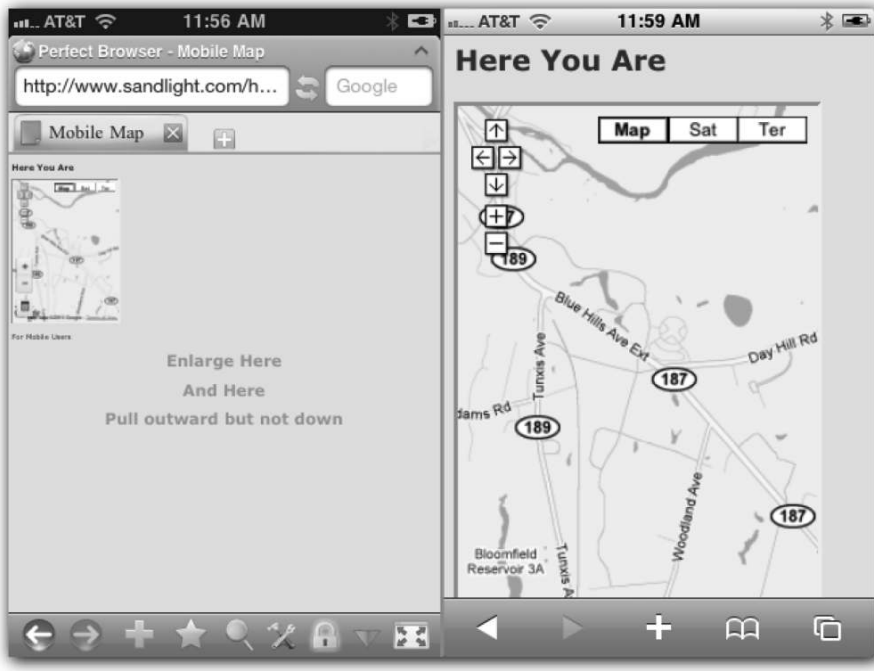


Figure 15.2 — Carte dans un environnement mobile.

15.1.3 Exploitation des propriétés de l'objet geolocation et du plug-in Google Earth

Voici une liste complète des propriétés de l'objet geolocation qui peuvent fournir de nombreuses informations :

- `latitude` : coordonnées géographiques en degrés
- `longitude` : coordonnées géographiques en degrés
- `altitude` : hauteur en mètres
- `accuracy` : niveau de précision en mètres des coordonnées de latitude et longitude
- `altitudeaccuracy` : niveau de précision en mètres de l'altitude
- `heading` : direction du déplacement du périphérique hôte en degrés (pertinent sur un périphérique mobile)
- `speed` : vitesse du périphérique hôte en mètres par seconde (pertinent sur un périphérique mobile)

Si vous disposez d'un téléphone mobile, vous pouvez, par exemple, tester l'orientation et la vitesse en voiture, si ce n'est pas vous qui conduisez ! Toutes les propriétés de `geolocation` peuvent être envoyées à un formulaire pour être affichées si vous le souhaitez. Si vous utilisez ces informations sur un périphérique mobile, vous aurez besoin d'un serveur open-socket ou d'un fréquent rafraîchissement de page.

Il est également possible d'employer `geolocation` avec le plug-in Google Earth. La figure 15.3 illustre une version modifiée de la page Web de base avec le plug-in qui permet de générer un affichage en 3D de la zone cartographiée.

Vous pouvez mettre à jour l'exemple de page Web avec les mêmes dimensions en donnant à la balise `<iframe>` les attributs suivants : `width=500 height=400`. Il suffit ensuite de cliquer sur l'option Earth au sommet de la carte ; si votre navigateur a le plug-in, il affichera une vue en 3D. Dans le cas contraire, on vous proposera de télécharger le plug-in et de l'installer sur le navigateur.

Figure 15.3 — Vue en 3D de la zone cartographiée avec le plug-in Google Earth.

15.2 STOCKAGE EN HTML5

Quand on évoque le stockage de données dans le navigateur de l'utilisateur, on pense d'abord aux cookies, puis viennent en général à l'esprit les bases de données et les programmes comme PHP et ASP.NET. Le DOM HTML5 possède cependant un objet de stockage qui peut être utilisé dans quatre contextes différents :

- Stockage de session,
- Stockage global,
- Stockage local,
- Stockage de base de données.

Tous les navigateurs ne supportent pas tous ces contextes de stockage, mais au fur et à mesure que les navigateurs sont mis à jour, ils incluent de plus en plus de contextes. Au moment de la rédaction de cet ouvrage, Safari, Chrome et Opera prenaient en charge tous les contextes sauf le stockage global ; Firefox prenait en charge tous les contextes, sauf le stockage de base de données.

Le stockage est réalisé à l'aide de paires clé/valeur, la clé étant un identifiant d'une valeur donnée (la clé ressemble à une variable avec un libellé et une valeur assignée). Les deux sections suivantes expliquent comment fonctionnent le stockage de session et le stockage local. Je laisse de côté le stockage global et le stockage de base de données car pour le moment ils sont moins bien implémentés.

15.2.1 Stockage de session

Le stockage de session permet aux utilisateurs de stocker des données d'une seule page Web tant que cette page Web est visualisée. Dès que l'utilisateur quitte la page, toutes les données stockées sont perdues. Pour les jeux interactifs, les programmes de calculatrice ou tout autre type de page qui a besoin d'un stockage temporaire, vous pouvez utiliser le stockage de session.

Pour commencer, vous devez examiner les méthodes qui permettent d'assigner des valeurs aux propriétés du stockage de session puis de les récupérer. Voici la syntaxe de base pour stocker une valeur :

```
sessionStorage.setItem("nomClé", valeur);
```

La `clé` doit être une chaîne de caractères (type string), alors que la `valeur` peut être de n'importe quel type de données acceptable (numérique, texte, booléen, objet, fonction). Le code suivant fournit quelques assignations valides :

```
this.maCle="deuxièmeCle"; //Nom de clé assigné à une propriété
function bonjour()          //Fonction avec une valeur de retour
{
     return "Salut !";
}
jill="Mon nom est Jill"; //Variable
//Assignation de valeurs aux clés
sessionStorage.setItem("premiereCle",88);     //Nombre (numérique littéral)
sessionStorage.setItem(this.maCle,true );  //Booléen
sessionStorage.setItem("troisiemeCle",bonjour() ); //Fonction
sessionStorage.setItem("quatriemeCle","Mon nom est Jack" ); //String littérale
sessionStorage.setItem("cinquiemeCle",jill );  //Variable
```

Comme vous pouvez le voir, on peut utiliser des variables pour les clés et leurs valeurs. Tant que la variable (ou la propriété) est une donnée de type string, elle peut être utilisée comme `clé` (vous pouvez même utiliser une fonction qui retourne une valeur string comme clé).

Une fois que vous avez stocké les données, vous avez besoin d'un moyen de les récupérer grâce à une méthode. Voici la syntaxe générale pour obtenir les données

stockées (il faut connaître le nom de la `clé` de chaque valeur que l'on souhaite récupérer).

```
sessionStorage.getItem(("nomClé");
```

Vous pouvez vous représenter le nom de la `clé` comme s'il s'agissait d'un nom de variable : si vous connaissez le nom de la variable, vous pouvez retrouver sa valeur. Les noms de clés fonctionnent de la même manière.

Le programme suivant (`StockageSession.html` disponible dans les fichiers du chapitre) fournit une illustration simple de la manière de travailler avec le stockage de session. Cela vous rappellera probablement le travail avec les variables car les valeurs survivent tant que l'on ne change pas de page.

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
StorageMaster=new Object();
//Set values
StorageMaster.setPositions=function()
{
sessionStorage.setItem("firstBase",document.players.firstBase.value );
sessionStorage.setItem("secondBase",document.players.secondBase.value );
sessionStorage.setItem("thirdBase",document.players.thirdBase.value );
}
//On récupère les valeurs
StorageMaster.getFirst=function()
{
playerName=sessionStorage.getItem("firstBase");
alert(playerName + " joue première base");
}
StorageMaster.getSecond=function()
{
playerName=sessionStorage.getItem("secondBase");
alert(playerName +" joue seconde base");
}
StorageMaster.getThird=function()
{
playerName=sessionStorage.getItem("thirdBase");
alert(playerName+ " est assigné à la troisième base");
}
</script>
<style type="text/css">
/*323F14,EBD4B2,273A4B,D49756,790007 */
body {
background-color:#EBD4B2;
color:#273A4B;
font-family:Verdana, Geneva, sans-serif;
}
h2 {
background-color:#273A4B;
color:#D49756;
text-align:center;
```

```
}
h3 {
color:#323F14;
}
fieldset {
color:#790007
}
#playerTable {
display:table;
}
#getPlayer {
display:table-cell;
width:250px;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Stockage</title>
</head>
<body>
<article>
<header>
  <hgroup>
    <h2>Entraîneur de baseball</h2>
    <h3>Assignation des joueurs :</h3>
  </hgroup>
</header>
<section>
<form name=players>
  <input type=text name=firstBase placeholder="Première base">
   Première base<br>
  <input type=text name=secondBase placeholder="Deuxième base">
   Deuxième base<br>
  <input type=text name=thirdBase placeholder="Troisième base">
   Troisième base<br>
  <input type=button onClick="StorageMaster.setPositions()" value="Assignation
des positions">
  </section>
  <br>
  <div ID="playerTable">
  <section ID="getPlayer">
  <fieldset>
    <legend>Qui joue quoi ?</legend>
    <input type=button onClick="StorageMaster.getFirst()" value="Qui est en
première base ?">
    <br>
    <input type=button onClick="StorageMaster.getSecond()" value="Qui est en
deuxième base ?">
    <br>
    <input type=button onClick="StorageMaster.getThird()" value="Qui est en
troisième base ?">
    <br>
  </fieldset>
</form>
</section>
</div>
</body>
</html>
```

Quand on charge la page pour la première fois, on voit un nouvel attribut HTML5 dans les champs de saisie de texte. Il s'agit de textes de substitution qui sont codés de la manière suivante :

```
< input type=text name=thirdBase placeholder="Troisième base">
```

Dès que l'utilisateur commence à saisir une valeur, ils disparaissent immédiatement. Testez ce programme et saisissez trois valeurs dans les champs, puis cliquez sur le bouton Assignation des positions. Ceci a pour effet de définir les valeurs de stockage de session.

Pour retrouver les valeurs stockées, il suffit de cliquer sur l'un des boutons de la zone « Qui joue quoi ? ». La figure 15.4 illustre le résultat attendu.

Si vous tentez de récupérer les données avant d'avoir cliqué sur le bouton Assignation des positions, vous obtiendrez une valeur `null` dans la boîte de dialogue. Si vous quittez la page et y revenez, vous obtiendrez aussi des valeurs null tant que vous n'aurez pas réassigné les positions des joueurs.

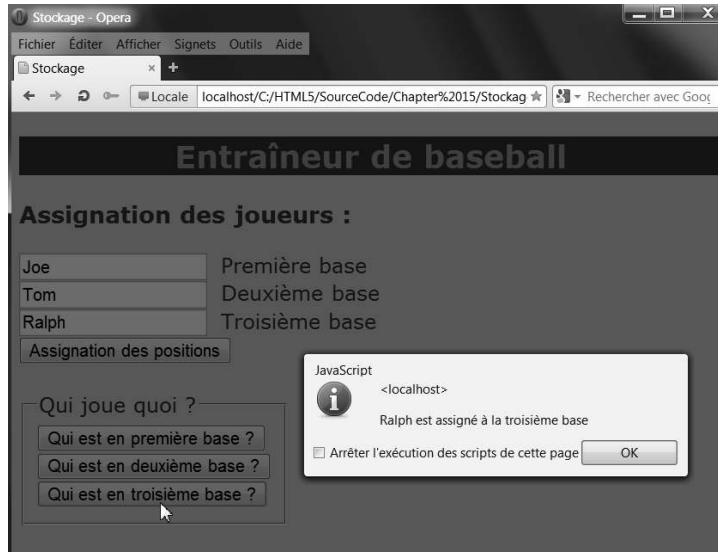


Figure 15.4 — Données stockées et affichées dans une boîte de dialogue.

15.2.2 Stockage local

La principale différence entre le stockage de session et le stockage local est que le stockage local conserve les données. Les utilisateurs peuvent quitter le site, éteindre leur ordinateur, revenir le lendemain, et les données seront toujours là. Le stockage local fonctionne comme les cookies, mais certaines différences sont importantes :

- Les cookies disposent d'un très faible espace de stockage, alors que le stockage local a bien plus d'espace.

- Les cookies sont retransmis automatiquement à chaque requête du serveur, ce qui n'est pas le cas du stockage local ; cela signifie que le stockage local demande beaucoup moins de travail au serveur et au navigateur. Le stockage local n'est transmis que si l'on en fait la demande.

Vous verrez que le stockage local et le stockage de session utilisent les mêmes méthodes pour définir les valeurs et les récupérer, si bien que vous n'avez à apprendre la syntaxe qu'une seule fois. L'exemple suivant (`StockageLocal.html` disponible dans les fichiers du chapitre) montre comment stocker, retrouver et effacer les données du stockage local.

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
StorageMaster=new Object();
//Définition des valeurs
StorageMaster.setRegistration=function()
{
this.hobbyNow="";
this.topCount=document.loisirs.elements.length;
for(var count=0;count < this.topCount;count++)
{
if(document.loisirs.elements[count].checked)
{
this.hobbyNow=document.loisirs.hobby[count-2].value;
}
}
localStorage.setItem("uNom",document.loisirs.userName.value);
localStorage.setItem("uHobby",this.hobbyNow);
localStorage.setItem("uVille",document.loisirs.ville.value);
}
//Récupération des valeurs
StorageMaster.getReg=function()
{
userProfile="Profil Utilisateur :\n";
nomNow=localStorage.getItem("uNom")+"\n";
hobbyNow=localStorage.getItem("uHobby")+"\n";
villeNow=localStorage.getItem("uVille")+"\n";
fileLength=localStorage.length + " éléments dans le profil";
this.profile=userProfile+nomNow+hobbyNow+villeNow+fileLength;
document.getElementById("profile").innerHTML = this.profile;
}
StorageMaster.clearReg=function()
{
localStorage.clear();
alert("Stockage local effacé");
}
</script>
<style type="text/css">
/*962D3E,343642,979C9C,F2EBC7,348899 */
body {
background-color:#F2EBC7;
color:#962D3E;
```

```
font-family:Verdana, Geneva, sans-serif;
}
h2 {
color:#979C9C;
}
fieldset {
color:#348899;
}
#hobbyTable {
display:table;
}
#getHobby {
display:table-cell;
width:275px;
}
#profile {
display:table-cell;
background-color: #979C9C;
padding: 3px;
width:150px;
font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;
font-size:14px;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Stockage</title>
</head>
<body>
<article>
<header>
  <h2>Enregistrement des loisirs</h2>
</header>
<section>
  <form name="loisirs">
    <input name=userName placeholder="Nom SVP">
     Nom<br>
    <div id="hobbyTable">
      <section id="getHobby">
        <fieldset>
          <legend>Quel est votre loisir favori ?</legend>
          <label>
            <input type=radio name=hobby value="voyage">
            Voyage</label>
          <br>
          <label>
            <input type=radio name=hobby  value="lecture">
            Lecture</label>
          <br>
          <label>
            <input type=radio name=hobby value="théâtre">
            Théâtre</label>
          <br>
          <label>
            <input type=radio name=hobby value="danse">
            Danse</label>
          <br>
          <label>
```

```
            <input type=radio name=hobby value="frisbee">
            Frisbee</label>
          <br>
        </fieldset>
      </section>
    </div>
    <input type=text name=ville placeholder="Ville">
     Ville<br>
    <input type=button onClick="StorageMaster.setRegistration()"
value="Enregistrer">
    <input type=button onClick="StorageMaster.getReg()" value="Trouver infos">
    <input type=button onClick="StorageMaster.clearReg()" value="Effacer les
données">
  </form>
</section>
<br>
<pre id="profile"></pre>
</body>
</html>
```

On a ajouté à cet exemple l'utilisation de boutons radio pour transmettre les données à stocker. Les boutons radio sont importants car ils facilitent le choix des utilisateurs. Cela demande un peu plus de travail pour obtenir les informations correctes des boutons radio et des cases à cocher, mais cela illustre la maxime du Web qui prétend que plus le développeur travaille, plus l'utilisateur a la tâche facile.

Vous noterez aussi que le stockage local est lié au navigateur, ce qui signifie que chaque navigateur a son propre stockage. Dans ces conditions, si vous stockez des données en utilisant le navigateur Safari, le navigateur Chrome ne peut donc pas y accéder. La figure 15.5 illustre une page chargée dans le navigateur Firefox qui a stocké des données grâce au stockage local. Quand le même programme est chargé dans le navigateur Opera et tente de récupérer les données stockées, il obtient pour toute réponse la valeur null.

Figure 15.5 — Accès aux données du stockage local.

Vous remarquerez aussi que quand on charge le programme la première fois, on ne voit pas la fenêtre de résultat, mais juste une ligne grisée sous les boutons. Dès que l'on clique sur le bouton Trouver infos, les informations apparaissent à l'emplacement de la ligne grisée. Ceci est l'œuvre d'un peu de code CSS3 et du DOM HTML5. Commencez par définir l'ID suivant dans le code CSS3 :

```
#profile {
     display:table-cell;
     background-color: #979C9C;
     padding: 3px;
     width:150px;
     font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;
     font-size:14px;
}
```

En utilisant la ligne JavaScript suivante :

```
document.getElementById("profile").innerHTML = this.profile;
```

les informations stockées dans `this.profile` sont envoyées à la page Web où la balise suivante a été placée :

```
<pre id="profile"></pre>
```

Avant HTML5, envoyer à une page Web des données de manière dynamique sans recharger la page était bien plus complexe. Cependant, pour certains programmes comme Adobe Flash CS6, c'est assez facile, comme nous allons le voir dans la prochaine section.

15.3 AJOUT ET AJUSTEMENT D'OBJET SUR DES PAGES WEB HTML5

Quand la première version de HTML est sortie, elle ne pouvait pas faire grand-chose si bien que les développeurs ont commencé à utiliser des programmes comme Java et Flash qui offraient des fonctionnalités absentes de HTML. La plupart de ces restrictions ne sont plus d'actualité aujourd'hui grâce à HTML5, mais même si HTML5 en fait beaucoup plus que les précédentes versions de HTML, les dernières versions de Flash et de Java sont toujours bien supérieures.

Il est prévisible qu'Adobe Flash CS6 et HTML5 vont travailler de concert, en dépit du fait que l'iPhone et l'iPad ne prennent pas en charge le lecteur Flash. L'un des grands mérites de Flash, c'est qu'il assure de la cohérence entre les différentes plateformes et les différents navigateurs. Même si les éditeurs de navigateurs ont des versions différentes du DOM HTML et des idées différentes sur la meilleure façon d'implémenter CSS et JavaScript, le plug-in Flash a longtemps assuré une cohérence

de présentation entre les plateformes et les navigateurs. C'est la raison pour laquelle les concepteurs et les développeurs ont utilisé Flash.

15.3.1 Ajout d'un objet

Pour vous donner une idée de la manière dont on incorpore un objet en HTML5, j'ai créé une animation simple d'étoile filante en Flash. La figure 15.6 illustre cette petite animation dans la fenêtre de conception.

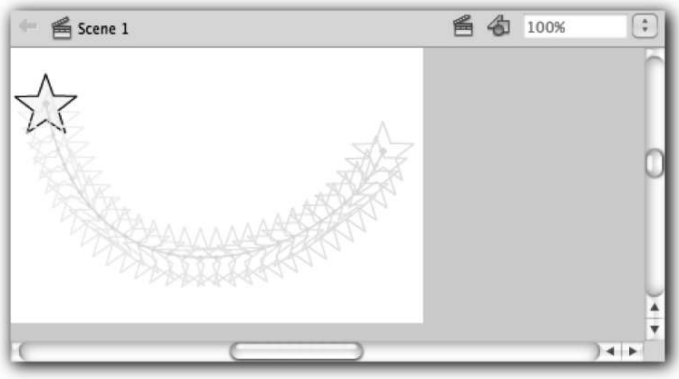


Figure 15.6 — Animation Flash.

Vous pouvez placer l'animation sur une page Web de plusieurs manières différentes, mais le plus simple est de la publier en Flash, ce qui génère automatiquement une page Web avec une référence au fichier binaire qui est stocké dans le format `.swf`. Dans les navigateurs équipés du plug-in Flash (ce qui est le cas de pratiquement tous les navigateurs), le code suivant (`EtoileFilante.html` disponible dans les fichiers du chapitre) illustre l'objet empaqueté dans une page HTML5.

```
<!DOCTYPE HTML>
<html>
<head>
<title>Etoile filante</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<style type="text/css" media="screen">
html, body { height:100%; background-color: #ffffff;}
body { margin:0; padding:0; overflow:hidden; }
#flashContent { width:100%; height:100%; }
</style>
</head>
<body>
<div id="flashContent">
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" width="300"
height="200" id="ShootingStar" align="middle">
<param name="movie" value="ShootingStar.swf" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
```

```
<param name="play" value="true" />
<param name="loop" value="true" />
<param name="wmode" value="window" />
<param name="scale" value="showall" />
<param name="menu" value="true" />
<param name="devicefont" value="false" />
<param name="salign" value="" />
<param name="allowScriptAccess" value="sameDomain" />
<!--[if !IE]>-->
<object type="application/x-shockwave-flash" data="ShootingStar.swf"
width="300" height="200">
<param name="movie" value="ShootingStar.swf" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<param name="play" value="true" />
<param name="loop" value="true" />
<param name="wmode" value="window" />
<param name="scale" value="showall" />
<param name="menu" value="true" />
<param name="devicefont" value="false" />
<param name="salign" value="" />
<param name="allowScriptAccess" value="sameDomain" />
<!--<![endif]-->
<a href="http://www.adobe.com/go/getflash">
<img
src="http://www.adobe.com/images/shared/download_buttons/get_flash_player.gif"
alt="Get Adobe Flash player" />
</a>
<!--[if !IE]>-->
</object>
<!--<![endif]-->
</object>
</div>
</body>
</html>
```

15.3.2 Ajustement des objets

La balise `<object>` est l'élément clé HTML5. Plusieurs paramètres ont été inclus, mais ils peuvent tous être modifiés pour mieux correspondre à votre site. Par exemple, c'est le blanc (`#ffffff`) qui a été défini comme couleur de fond, mais vous pouvez en changer pour qu'elle s'accommode à votre site. De la même manière, vous pouvez modifier tous les styles CSS que vous souhaitez.

Un autre programme, appelé Flex ou Flash Builder, permet aussi de générer des fichiers `.swf`. Pour ajouter un fichier `.swf` à son propre code, le développeur HTML5 n'a qu'à insérer le code généré automatiquement par Flash et Flash Builder. En utilisant un programme très puissant nommé ActionScript 3.0, les développeurs sont également capables de créer des programmes aussi performants que ceux qui sont écrits dans des langages aussi réputés que Java et C++.

15.4 À VOUS DE JOUER !

J'espère que vous allez apprécier ce défi qui met en scène à la fois le nouvel objet `geolocation` et le stockage local en HTML5. Comme vous l'avez vu dans ce chapitre, tout ce dont vous avez besoin pour placer une carte Google Map sur votre page Web ce sont la latitude et la longitude de l'emplacement. L'objet `geolocation` génère ces valeurs pour vous en HTML5 en fonction de votre position. Si vous avez un périphérique mobile, vous pouvez générer cette information dans plusieurs emplacements différents. Sinon, vous pouvez aller sur un programme de cartographie en ligne, saisir une adresse et obtenir ainsi les coordonnées de l'emplacement souhaité. Voici le défi que vous avez à relever :

- Récupérez la longitude et la latitude de cinq emplacements différents.
- Saisissez la longitude et la latitude dans un objet de stockage local (`localStorage`).
- Définissez cinq boutons qui appelleront un programme JavaScript qui chargera cinq cartes à la demande.

Fondamentalement, vous allez créer une page Web qui charge des cartes de n'importe quel emplacement que vous aurez choisi. Vous ne devriez pas avoir besoin de plus de JavaScript que le peu qui a été étudié dans ce chapitre.

Index

B

C

D

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X