Jean Engels

XHTML et CSS

Cours et exercices

Cours et exercices

EYROLLES

XHTML et CSS

Cours et exercices

CHEZ LE MÊME ÉDITEUR

Du même auteur

J. ENGELS. – **PHP 5.** *Cours et exercices* (collection Noire).
N°11407, 2005, 518 pages.

CSS 2 et design web

D. SHEA, M. HOLZSCHLAG. – **Le Zen des CSS.**
N°11699, 2005, 296 pages.

R. GOETTER. – **CSS 2: pratique du design web**(collection Blanche).
N°11570, 2005, 324 pages.

R. GOETTER. – **Mémento CSS**.
N°11726, 2005, 14 pages.

J. ZELDMAN. – **Design web: utiliser les standards.**
N°11548, 2005, 440 pages.

K. GOTO, E. KOTLER. – **[Re]design web 2.0.***Conduite de projet*.
N°11579, 2005, 294 pages.

S. BORDAGE. – **Conduite de projets web.**
N°11599, 2e édition 2005, 384 pages.

Autres ouvrages

E. DASPET etC. P IERRE deG EYER. – **PHP 5 avancé**(collection Blanche).
N°11669, 2e édition 2005, 804 pages.

G. PONÇON. – **Best practices PHP**(Architecte logiciel).
N°11676, 2005, 490 pages.

S. MARIEL. – **PHP 5**(Les Cahiers du programmeur).
N°11234, 2004, 290 pages.

V. CARON, Y. FORGERIT *et al*. – **SPIP 1.8** (Les Cahiers du programmeur).
N°11428, 2005, 450 pages.

A. CAOUISSIN. – **Dreamweaver MX 2004.**
N°25501, 2004, 1032 pages.

J.-M. DEFRANCE. – **PHP/MySQL avec Dreamweaver MX 2004** (Best of Eyrolles).
N°11709, 2004, 550 pages (semi-poche).

C. DELANNOY. – **Programmer en Java** (collection Noire).
N°11748, 4e édition, 2006, 740 pages + CD-Rom.

A. TASSO, S. ERMACORE. – **Initiation à JSP** (collection Noire).
N°11532, 2004, 354 pages + CD-Rom.

J. WEAVER, K. MUKHAR, J. CRUME. – **J2EE 1.4** (collection Blanche).
N°11484, 2004, 662 pages.

J. MOLIÈRE. – **J2EE**(Les Cahiers du programmeur).
N°11574, 2005, 234 pages.

A. PATRICIO. – **Hibernate 3.0**(collection Blanche).
N°11644, 2005, 336 pages.

K. DJAFAAR. – **Eclipse et JBoss**.
N°11406, 2005, 656 pages + CD-Rom.

Jean Engels

XHTML et CSS

Cours et exercices

EYROLLES

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris CEDEX 05
www.editions-eyrolles.com

Table des matières

PARTIE II

Les styles CSS

CHAPITRE 9

CHAPITRE 10

Avant-propos

HTML est un langage aujourd'hui dépassé ! Il est désormais remplacé par l'association du langage XHTML (eXtensible HyperText Markup Language) qui permet de structurer précisément les contenus des sites Internet et de CSS (Cascading Style Sheets) pour créer la mise en page et les différentes présentations de ces contenus pour les terminaux les plus divers, allant du navigateur classique au téléphone portable. La conception des sites devant se faire en deux phases (contenu et présentation), cet ouvrage est lui-même divisé en deux grandes parties.

La première traite du langage XHTML en tant qu'outil de structuration des documents. Elle vous permettra d'acquérir une bonne connaissance de tous les éléments disponibles dans ce but. À ce stade, et même si nous indiquons la présentation par défaut liée à chacun d'eux, vous ne devriez pas vous préoccuper outre mesure de cet aspect et ne jamais choisir tel élément en fonction de son aspect prédéterminé, mais en fonction de son rôle logique dans la structuration du contenu. C'est cette différence fondamentale de conception qui crée la véritable révolution de XHTML par rapport à HTML.

Cette première partie comprend les chapitres suivants :

- Le chapitre 1 est une introduction générale destinée à mettre en évidence les différences entre XHTML et HTML. Il définit les règles qui vous permettront de créer un document XHTML bien formé et conforme aux standards du W3C. Il indique également les outils logiciels nécessaires à la création puis à la mise en place sur un serveur distant des pages Web que vous allez créer.
- Le chapitre 2 vous permet de créer la structure générale d'une page conforme à XHTML selon la version que vous choisirez d'utiliser, sachant que nous recommandons d'utiliser la dernière version disponible, en l'occurrence XHTML 1.1. Cette structure fait apparaître les éléments essentiels qui sont communs à toutes les pages, comme la déclaration DOCTYPE. Vous y trouverez tous ceux qui constituent l'en-tête d'une page et qui, s'ils ne créent généralement pas de parties visibles dans un navigateur, ont un rôle important souvent négligé. En particulier, ils permettent par exemple de lier la page à des ressources externes comme une feuille de style ou des scripts JavaScript. Certains autres éléments jouent aussi un rôle essentiel dans le référencement de votre site, en permettant d'inclure des informations sur le document, dites

méta-informations. Vous trouverez enfin dans ce chapitre une présentation de l'ensemble des éléments incorporables dans le corps d'un document.

- Le chapitre 3 permet de faire un tour d'horizon complet de tous les éléments susceptibles de contenir du texte, les diverses formes de structuration d'une page en divisions et en paragraphes ainsi que tous les éléments sémantiques en ligne. Enfin nous découvrirons comment structurer l'information au moyen des différentes formes de listes dont l'utilisation est courante dans la création de menus par exemple.
- Le chapitre 4 nous fait découvrir comment insérer les différents types d'images admis par les navigateurs et les objets multimédias les plus divers (photos de tous types, sons, animations Flash…) en complément du texte dans une page. Nous verrons également comment diviser une image en plusieurs zones réactives aux clics du visiteur.
- Le chapitre 5 présente ce qui est la base des documents hypertextes, à savoir la création de liens déclenchés à partir d'un texte, d'un bouton ou d'une image, soit entre les différents éléments d'une même page, soit entre les pages d'un même site pour créer un système de navigation complet. Ces liens peuvent permettre également de déclencher le téléchargement de documents externes non affichables dans une page Web, l'envoi d'un e-mail ou encore un script JavaScript.
- Le chapitre 6 nous permet d'utiliser les différents éléments qui interviennent dans la création de tableaux. Il aborde et définit tout d'abord la structure générale commune à tous les tableaux. Les éléments permettant la création de lignes, de cellules et leurs regroupements sémantiques éventuels en groupes de ligne ou de colonnes sont ensuite traités. Les tableaux sont envisagés ici aussi bien pour permettre la structuration de données mais également comme moyen d'organisation d'une page. Nous précisons cependant les limites des tableaux en tant que technique de mise en page, l'usage des styles CSS étant une bonne alternative à ce type d'organisation.
- Le chapitre 7 présente le seul moyen de rendre une page Web interactive, à savoir en y incorporant des formulaires. Après avoir défini la structure globale de ces derniers, nous décrirons l'ensemble des composants qui permettent de saisir du texte ou des mots de passe, et d'effectuer des choix à l'aide de boutons radio, de cases à cocher ou de listes de sélection d'options. Nous verrons enfin comment réaliser le transfert de fichiers du poste client vers un serveur. Nous montrons également dans ce chapitre comment utiliser des tableaux pour améliorer la structure d'un formulaire.
- Le chapitre 8 présente la création de pages avec des cadres (*frames*). Cette technique est aujourd'hui considérée comme obsolète en raison de ses nombreux inconvénients, mais nous la présentons par souci d'exhaustivité et parce qu'elle est encore admise dans la version 1.0 de XHTML. La création de cadres horizontaux, verticaux ainsi que les structures complexes y sont étudiées en utilisant l'imbrication des cadres. Il est cependant fait référence aux techniques CSS qui permettent d'obtenir des résultats similaires sans créer de cadres.

La seconde partie de cet ouvrage traite de la création des feuilles de style au moyen de CSS. Elle est le complément indispensable de XHTML puisque la première partie a bien mis l'accent sur le fait de la séparation indispensable du contenu et de la présentation.

- Le chapitre 9 vous permet d'apprendre le fonctionnement des CSS et la syntaxe utilisée dans la création des styles applicables à un document XHTML. Ce chapitre constitue une étape essentielle dans l'apprentissage des CSS car il aborde les nombreux sélecteurs qui permettent, par exemple, d'appliquer un même style à toutes les occurrences d'un élément ou d'appliquer des styles différents à un même élément en fonction de son contexte. C'est du bon usage de ces sélecteurs que dépendra toute la puissance et la diversité d'utilisation des styles que vous allez créer par la suite. Nous y étudions enfin les règles d'héritage et de cascade des propriétés CSS.

- Le chapitre 10 permet de découvrir les propriétés de gestion des couleurs aussi bien pour le texte que pour le fond d'un élément, puis celles qui permettent de créer des images de fond et leurs différents types de positionnement dans tous les éléments XHTML.

- Le chapitre 11 présente tout d'abord le modèle de « boîte » applicable à tous les éléments CSS, puis traite de la création des bordures (style, épaisseur, couleur) applicables à chaque élément individuellement. Nous y abordons également la création des marges entre la boîte d'un élément et son environnement, ainsi que l'espacement entre son contenu et ses bordures. Toutes les propriétés abordées ici sont de nature à affiner la présentation à l'intérieur d'un document. La notion récente de contour y est également étudiée.

- Le chapitre 12 fait un tour d'horizon des propriétés applicables aux textes, qu'il s'agisse du choix d'une police de caractères, des différentes façons de définir sa taille de manière absolue ou relative par rapport au contexte, du choix de sa couleur, de son style, de sa casse, de sa graisse, sans oublier les nombreuses autres possibilités décoratives. Nous décrivons aussi la gestion des interlignes, de l'alignement et de l'espacement du texte. Ce chapitre présente enfin les propriétés spécifiques aux liens hypertextes et les sélecteurs particuliers qui permettent de réaliser des effets dynamiques.

- Le chapitre 13 apporte les éléments essentiels qui permettent de gérer la présentation et la mise en page globale d'un document. Nous y étudions les méthodes de dimensionnement des éléments ainsi que les méthodes de positionnement qui sont des avancées essentielles de l'association CSS/XHTML par rapport à HTML. Elles permettent de remplacer les méthodes de mise en page habituelles comme celles qui usent et abusent des tableaux ou encore celles qui utilisent des cadres. La richesse de ces propriétés permet également d'agir sur la visibilité et l'ordre d'empilement des éléments. Toutes ces propriétés permettent de créer les mises en page les plus diverses.

- Le chapitre 14 est spécialement dédié aux tableaux qui possèdent un modèle de gestion particulier. Nous montrons comment gérer la couleur des cellules en fonction, par exemple, de leur appartenance à un groupe de lignes ou de colonnes. Nous étudions également la gestion des bordures des cellules, de la détermination de la largeur des

colonnes ou de l'ensemble d'un tableau, des alignements spécifiques à l'intérieur des cellules ou des groupes, etc.

- Le chapitre 15 est destiné spécifiquement aux listes qui constituent un moyen de structuration efficace, très approprié aux menus par exemple. Nous y abordons les multiples styles de numérotation possibles pour les listes ordonnées ou à puces graphiques et leur position par rapport aux items. La numérotation automatique utilisant la création de compteurs est une autre façon de numéroter des listes générées dynamiquement. La modification du rendu habituel des éléments nous permet de créer des listes en ligne constituant par exemple des menus horizontaux, ou encore le rendu sous forme de liste d'un ensemble d'éléments dont ce n'est pas la vocation initiale.
- Le chapitre 16 est consacré aux moyens disponibles pour obtenir un rendu correct du contenu d'un document Web à l'impression, et plus généralement sur tout support constitué de pages calibrées (fichiers .pdf, présentations de style diaporama, écrans de petite taille…). Nous signalons cependant que seul un nombre limité des propriétés CSS spécifiques à ce type de support sont aujourd'hui supportés par l'ensemble des navigateurs.

Enfin, les annexes A, B, C, et D proposent des références complètes des éléments XHTML et de leurs attributs, des propriétés CSS 2.1 et de leurs valeurs, des codes de couleurs conseillées sur le Web et des entités de caractères.

Les exercices proposés à la fin de chaque chapitre vous permettront une mise en œuvre immédiate des points étudiés et de tester l'ensemble des connaissances acquises.

Les corrigés de ces exercices ne figurent pas dans cet ouvrage pour ne pas l'alourdir inutilement, mais ils sont téléchargeables librement sur le site des éditions Eyrolles (www.editions-eyrolles.com) et sur mon site dédié à ce sujet (www.funxhtml.com). Ils vous permettront de mesurer votre compréhension des notions abordées.

Partie I

Le langage XHTML

1

Introduction à XHTML

Il n'y aura pas de HTML 5. C'est ce qu'a confirmé le W3C (World Wide Web Consortium), l'organisme qui édite les recommandations des langages du Web : le HTML est mort en tant que tel. Certes, pendant des années, il a permis à tous de « bricoler » des pages web plus ou moins bien ficelées, mais il était devenu trop permissif, et surtout se caractérisait par un manque de rigueur assez flagrant. La rivalité entre Netscape et Microsoft a entraîné la création de balises propriétaires utilisables uniquement dans l'un des navigateurs, chacun s'ingéniant à créer le gadget qui lui attirerait le plus d'utilisateurs. Cette situation ne faisait que gêner les créateurs de pages web qui étaient obligés de prévoir des solutions alternatives aux balises manquantes en fonction du navigateur client. Il est évident que c'est la pression du e-commerce qui a provoqué la disparition de HTML, et que la naissance de XML (eXtensible Markup Language) l'a précipitée. Le XML aurait pu l'emporter tout de suite (car à terme c'est lui qui restera), mais c'était sans compter avec sa complexité et les problèmes de lecture qu'il pose aux utilisateurs avec des navigateurs d'anciennes générations. Il a donc fallu définir une alternative au tout XML.

Généalogie du XHTML

XHTML (eXtensible HyperText Markup Language) est un langage de balisage (dit aussi langage de marquage) qui permet de structurer le contenu des pages web dans différents éléments. Voilà une définition bien abstraite, reconnaissons-le, mais nous y reviendrons en détail dans la section suivante en présentant la notion de balisage.

Historiquement, les langages de balisage sont issus du langage SGML (Standard Generalized Markup Language) créé en 1986 pour structurer des contenus très divers.

Ce langage s'est révélé trop complexe pour être appliqué tel quel au Web, d'où la nécessité d'en créer une version allégée respectant les mêmes principes essentiels.

L'inventeur du HTML (1992), Tim Berners-Lee, l'avait conçu à l'origine comme un outil de structuration des contenus, principalement textuels, et non pas pour créer des présentations diversifiées. Ce sont les développements successifs, l'essor populaire du Web et la concurrence acharnée entre Netscape et Microsoft pour s'emparer du marché des navigateurs, qui ont détourné HTML de sa vocation première avec l'ajout d'éléments de design qui n'avaient rien à y faire. Voulant faire mieux que l'autre, chacun des deux grands a empilé des couches superflues sur HTML. Il est vrai que l'entrée du Web dans le grand public nécessitait de répondre à une demande d'interfaces graphiques plus esthétiques.

L'absence d'un langage particulier dédié uniquement à la présentation poussait effectivement les webmestres à utiliser tous les moyens pour créer des présentations visuelles agréables. L'apparition de CSS (Cascading Styles Sheets) en 1996 aurait dû résoudre le problème du détournement de HTML de sa destination première. Les mauvaises habitudes étaient prises et la facilité faisait le reste.

L'apparition de HTML 4, et particulièrement de sa version « strict » associée à l'emploi systématique de CSS 2 (publié en 1998), pouvait apporter une solution efficace à ce problème. La création de XML (eXtensible Markup Language) en 1998 et son succès dans de multiples domaines d'application ont conduit le W3C (World Wide Web Consortium) à créer le langage XHTML, non plus comme une nouvelle version de HTML, mais comme une reformulation de HTML en tant qu'application XML. Au niveau des éléments et des attributs disponibles, il existe à vrai dire très peu de différences entre HTML 4 strict et XHTML 1.1.

Les éléments, balises et attributs

Mais au juste comment fonctionne XHTML et qu'est-ce qu'un langage de balisage ?

Vous avez sûrement déjà utilisé un traitement de texte tel que Word. Votre texte peut comprendre des titres, des paragraphes, des images, des tableaux, et vous pouvez utiliser différentes polices de caractères et différentes tailles de caractères dans le même document. Le document final que vous avez réalisé ne laisse apparaître que le résultat de votre mise en page, mais en arrière-plan, votre traitement de texte a enregistré tous les paramètres de mise en page que vous avez utilisés en plus du texte lui-même.

Dans un langage de balisage, tout contenu, qu'il s'agisse de texte, d'image ou d'éléments multimédias les plus divers, doit être renfermé dans un élément. En XHTML, comme dans HTML, chaque élément a un nom déterminé et la liste des éléments utilisables est limitative et clairement définie dans la DTD (Document Type Definition) liée à la version utilisée du langage. C'est la grande différence entre XHTML et XML, langage dans

lequel c'est le programmeur qui crée ses propres éléments selon ses besoins. À quelques exceptions près, un élément a la structure suivante :

```
<nom_element> Contenu </nom_element>
```

Son contenu est précédé par une balise d'ouverture <nom_element> et suivi par une balise de fermeture </nom_element>. Toutes les balises d'ouverture (ou marqueur) commence par le signe < et se terminent par le signe >. La balise de fermeture fait de même mais le nom de l'élément est précédé d'un slash (/). Les navigateurs interprètent donc les contenus en fonction du nom de l'élément et attribuent un style par défaut à chacun de ses contenus.

Les caractéristiques de chaque élément peuvent être précisées par des informations complémentaires que l'on désigne en tant qu'attributs de l'élément. Il peut s'agir par exemple de la définition de la largeur, de la hauteur ou de l'alignement du contenu. Comme nous le verrons, un certain nombre d'attributs sont communs à quasiment tous les éléments de base dans les sections suivantes.

Les attributs d'un élément sont toujours définis dans la balise d'ouverture et doivent être séparés les uns des autres par au moins une espace typographique. Chaque attribut doit avoir une valeur, contrairement à ce qui se pratiquait dans HTML 4, même s'il ne peut prendre qu'une valeur unique. Aucune valeur n'est donc implicite du moment que l'attribut figure dans la balise d'ouverture. La présence de certains attributs est obligatoire dans quelques éléments particuliers, ce que nous préciserons systématiquement quand ce sera le cas. La plupart du temps, les attributs d'un élément sont facultatifs et c'est au programmeur de déterminer leur définition par rapport au cas qu'il doit traiter. Nombre d'attributs ont une valeur par défaut. Cela signifie que même si on ne les définit pas dans l'élément, celui-ci se comporte comme si nous avions défini explicitement cette valeur. Il est donc important de connaître ce type d'attribut et de ne pas négliger de les définir avec une autre valeur si ce comportement par défaut n'est pas désiré. La valeur de tous les attributs doit être définie entre des guillemets doubles. La syntaxe conforme d'un élément ayant des attributs est donc la suivante :

```
<nom_element attribut1="valeur1" attribut2="valeur2" > Contenu de l'élément
➥ </nom_element>
```

Le contenu d'un élément peut être constitué de texte ou d'autres éléments qui, eux-mêmes, peuvent en contenir d'autres, et ainsi de suite. Cet ensemble d'inclusion constitue la hiérarchie du document XHTML.

Les attributs de base de XHTML

Dans leur quasi-totalité, les éléments disponibles en XHTML ont en commun un ensemble d'attributs ayant chacun le même rôle. Ces attributs se répartissent en trois catégories. Chaque élément peut avoir par ailleurs d'autres attributs particuliers. Quand nous définirons par la suite les différents éléments, nous signalerons s'ils possèdent ces attributs sans rappeler leur définition.

Les attributs courants (noyau)

Ils s'appliquent à quasiment tous les éléments. On compte trois principaux attributs courants et un quatrième encore accepté mais dont l'emploi est déconseillé :

- L'attribut `id`. Il sert à identifier un élément de manière unique en lui donnant un nom, soit pour lui attribuer un style, soit pour y faire référence sans ambiguïté dans un script JavaScript.
- L'attribut `class`. Il contient le nom d'une classe CSS qui contient des définitions de styles. Comme nous le verrons dans la deuxième partie, son usage est très répandu pour affecter des styles ponctuellement à un élément.
- L'attribut `title`. Il contient un texte qui apparaît dans une bulle quand l'utilisateur positionne le curseur quelques instants (ce n'est pas instantané) sur un élément. Le texte qu'il contient peut servir à fournir une information ou une explication sur le rôle de l'élément.
- L'attribut `style`. Il permet de définir un style localement pour un élément donné. Il est encore toléré en XHTML 1.1 mais déconseillé.

Les attributs d'internationalisation

- L'attribut `xml:lang`. Il qui remplace l'attribut `lang` de HTML 4.
- L'attribut `dir`. Il indique le sens de lecture du contenu textuel d'un élément ; il peut prendre les valeurs `ltr` (lecture de gauche à droite) ou `rtl` (de droite à gauche).

Les attributs de gestion d'événements

Ces attributs permettent de gérer les événements dont un élément peut être le siège et qui sont créés par l'utilisateur. Leur contenu est un script écrit en général en langage JavaScript. Les DTD HTML 4 définissent dix attributs de gestion d'événements, y compris pour des éléments qui ne peuvent pas être le siège de ces événements. Il appartient donc aux programmeurs d'effectuer des tests pour vérifier la réalité des événements pour un élément donné. Vous trouverez ci-après la liste des gestionnaires de base et la description de l'événement correspondant.

Tableau 1-1. Les attributs gestionnaires d'événements communs

attribut	Action de l'utilisateur
onclick	Clic sur le contenu de l'élément.
ondblclick	Double-clic sur le contenu de l'élément.
onkeydown	Maintien d'une touche enfoncée.
onkeypress	Frappe sur une touche.
onkeyup	Relâchement d'une touche enfoncée.

Tableau 1-1. Les attributs gestionnaires d'événements communs (suite)

attribut	Action de l'utilisateur
onmousedown	Enfoncement d'un bouton de la souris.
onmousemove	Le curseur de la souris bouge dans la zone de l'élément.
onmouseout	Le curseur de la souris quitte la zone de l'élément.
onmouseover	Le curseur de la souris est au-dessus de la zone de l'élément.
onmouseup	Relâchement d'un bouton de la souris au-dessus de la zone de l'élément.

D'autres attributs gestionnaires d'événements sont spécifiques à certains éléments, que nous citerons au fur et à mesure de notre étude de ceux-ci.

Intérêt des spécifications

La création du XHTML ne consiste pas seulement en la redéfinition de quelques règles syntaxiques pour aménager le langage HTML. S'il ne s'agissait que de cela, cet ouvrage n'aurait pas lieu d'être. Il s'agit bien plus selon moi d'un changement de pensée et d'organisation qui doit s'opérer dans la création des pages web.

Une page web créée avec XHTML doit être pensée en distinguant deux parties :

- Un contenu, structuré au moyen des éléments XHTML (grandes divisions, titres, paragraphes, tableaux, images et liens, etc.). À ce stade, et même s'il en a déjà une idée, le créateur ne doit pas nécessairement avoir une idée définitive de la présentation finale. Il lui faut maîtriser principalement l'organisation des informations à fournir à un utilisateur.
- Une feuille de style CSS, définissant la mise en page de ces éléments en fonction du média qui va opérer le rendu du contenu (polices et tailles de caractères, bordures, marges, couleurs, positionnement dans la page, etc.). Les médias se diversifient en effet de plus en plus en devenant des éléments portables dotés de petits écrans, et il n'est pas impossible que le traditionnel écran d'ordinateur ne soit plus à l'avenir le principal vecteur d'affichage d'une page web. La séparation du contenu et de la présentation étant réalisée, il est possible d'associer à chaque média une feuille de style adaptée au terminal.

L'utilisation de ces méthodes présente les avantages suivants :

- Une meilleure organisation du contenu.
- Une meilleure qualité du code et plus grande rapidité d'affichage sur les navigateurs récents (Firefox, Mozilla, Internet Explorer, Netscape…).
- Une réduction des coûts de développement et de maintenance des sites web ainsi qu'une réutilisabilité accrue et rapide du code. En effet, en ayant respecté les principes

précédents, il est très facile de modifier rapidement toute la présentation d'une page sans toucher au code XHTML.

Les standards XHTML et CSS sont aujourd'hui incontournables pour tous ceux qui veulent concevoir un site web de manière professionnelle, et tous les étudiants en informatique et les professionnels du Web se doivent d'acquérir ou de mettre à jour leurs connaissances sur ces techniques.

Règles de base XHTML

Un document bien formé

Un document XHTML doit respecter certaines règles simples :

- Les éléments et les attributs sont sensibles à la casse et doivent être écrits en minuscules. Par exemple, `<body>` et non plus `<BODY>` comme en HTML.
- Les éléments non vides doivent avoir obligatoirement une balise d'ouverture et une balise de fermeture. Par exemple, on ne doit plus écrire :

```
<ol>
  <li>Item 1
  <li>Item 2
```

mais le code suivant :

```
<ol>
  <li>Item1 </li>
  <li>Item2 </li>
</ol>
```

- Les éléments vides ne comportent qu'une seule balise et doivent se terminer par les caractères `/>` précédés d'une espace pour marquer la fin de l'élément. Par exemple, il ne faut plus écrire :

```
<img src= "monimage.gif"> <hr> <br>
```

mais le code suivant :

```
<img src= "monimage.gif" /> <hr /> <br />
```

- Les éléments ne doivent pas se chevaucher. Ils doivent donc obéir au principe premier-ouvert-dernier-fermé. Dans ce cas, le premier élément est le parent du second et celui-ci est enfant du premier. Par exemple, le code suivant est incorrect :

```
<div> Cette division contient un titre <h1> Important ! </div> </h1>
```

et doit être remplacé par :

```
<div> Cette division contient un titre <h1> Important ! </h1></div>
```

- Tous les attributs doivent avoir une valeur incluse entre des guillemets doubles ("). Les différents attributs du même élément doivent être séparés par au moins une espace. Par exemple, il ne faut plus écrire :

```
<p class=styleperso title=attention> Texte important</p>
```

mais le code suivant :

```
<p class="styleperso" title="attention" > Texte important</p>
```

- À tous les attributs utilisés doit être donnée une valeur, y compris ceux dont la valeur est unique. Par exemple, il ne faut plus écrire :

```
<input type= "checkbox" checked disabled />
```

mais le code suivant :

```
<input type= "checkbox" checked="checked" disabled="disabled" />
```

- L'attribut `name` qui servaient à identifier certains éléments (`<a>` `<form>` par exemple) est supprimé et doit être remplacé par l'attribut `id`.
- Les scripts et les feuilles de style qui contiennent les caractères `<` et `&` peuvent figurer dans des sections CDATA de la façon suivante :

```
<script type="text/javascript">
  < ![CDATA[
    Code du script…
  ]]>
</script>
```

Ces sections n'étant pas actuellement reconnues par les navigateurs, nous ne les utiliserons pas dans cet ouvrage, mais il faudra en tenir compte dans un avenir proche. Une autre solution efficace consiste à inclure les scripts ou les feuilles de style contenant ces caractères dans des fichiers séparés et à les inclure à l'aide de l'élément `<link>` sur lequel nous reviendrons en détail par la suite.

Un document conforme

Un document XHTML se doit également de respecter les règles d'inclusion des éléments les uns dans les autres, telles qu'elles sont définies dans la DTD choisie. En effet, une DTD définit la liste limitative de tous les éléments XHTML utilisables et énumère ceux qui peuvent y être inclus. Le respect de ces contraintes est impératif pour que le document soit déclaré conforme à la DTD. Vous trouverez à cet effet tout au long de cet ouvrage lors de la description des éléments, et dans l'annexe A, non pas le texte des DTD XHTML 1 qui est particulièrement difficile à lire, mais son interprétation, pour chaque élément, sous la forme de la liste de ses éléments enfants (ceux qu'il peut inclure) et de ses éléments parents (ceux dans lesquels il peut être inclus).

Validation d'un document : le label du W3C

Malgré toutes les vérifications auxquelles vous pouvez procéder personnellement, il peut rester des erreurs de conformité dans votre code. Comme peut le faire un compilateur qui signale les éventuelles erreurs de syntaxe, le validateur du W3C permet de vérifier si le code est bien formé et conforme à la DTD indiquée. Pour lancer cette validation automatique, vous devez soumettre l'URL ou le code de vos documents XHTML au validateur du W3C accessible à l'adresse suivante : http://validator.w3.org

La figure 1-1 montre la page de validation du site du W3C dans laquelle plusieurs moyens sont mis à disposition pour valider un document. Vous pouvez saisir l'URL de la page si le document est déjà transféré sur un serveur (repère ³), choisir le fichier sur votre ordinateur en cliquant sur le bouton Parcourir (repère ·), ou encore écrire le code à vérifier directement dans une zone de saisie (repère »).

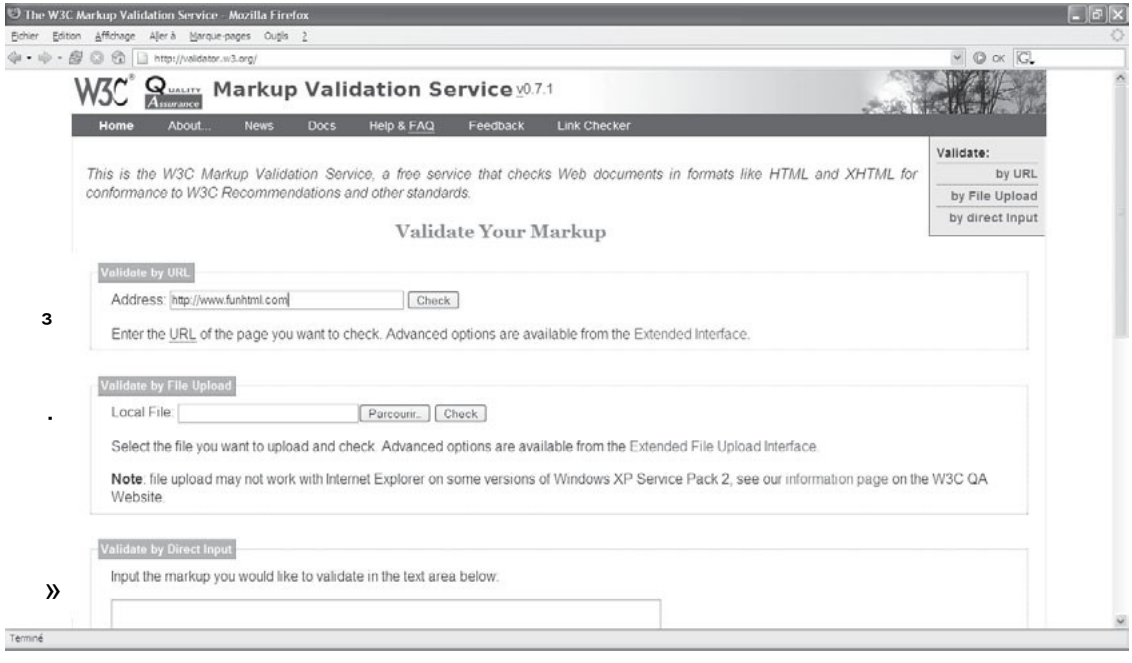


Figure 1-1
La page de validation du W3C

Le résultat de la validation est affiché dans une nouvelle page ; si le code est conforme, vous obtenez une page semblable à celle de la figure 1-2, sinon la liste des erreurs ainsi qu'un commentaire sont fournis.

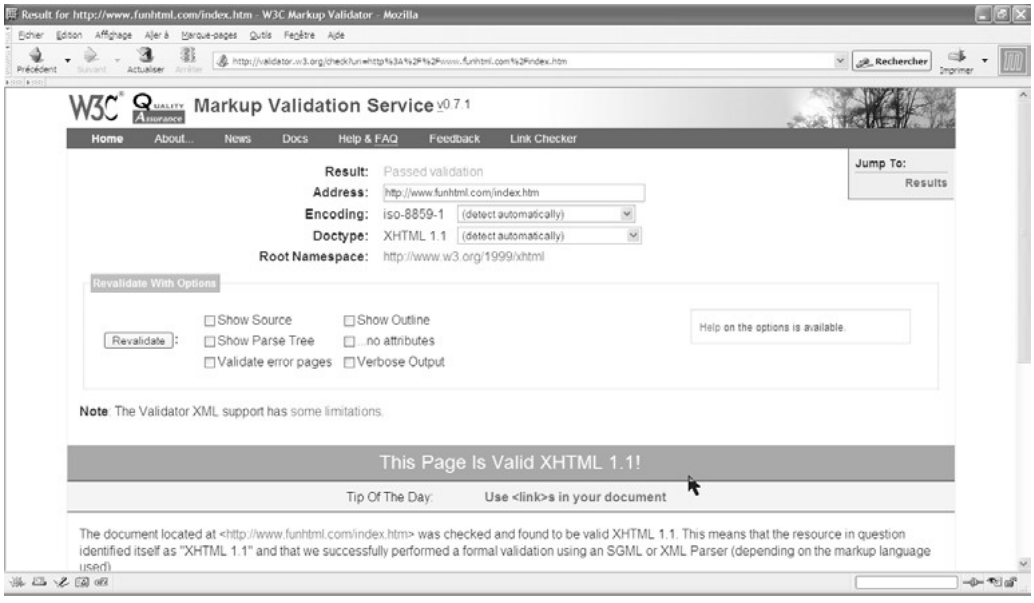


Figure 1-2

La page de résultat de validation du W3C

L'environnement de travail

Pour créer des pages web avec XHTML et CSS, il faut être doté d'un environnement de travail adapté. En théorie, un simple éditeur de texte tel que le Bloc Notes de Windows ou Emacs sous Linux par exemple peut suffire. Cependant, comme la saisie des noms de différents éléments XHTML dans ce type d'éditeur peut devenir à la longue plutôt rébarbative, nous mentionnons quelques outils susceptibles de faire gagner du temps.

Les éditeurs visuels

Dans ce type d'éditeurs, vous travaillez graphiquement sur une page en y incluant des éléments sans taper une ligne de code. Le plus connu est Dreamweaver, mais il en existe bien d'autres plus ou moins perfectionnés tel que Golive. L'inconvénient de ces éditeurs tient en fait à ce qui paraît être leur avantage : le code que l'on ne saisit pas est généré automatiquement et rien ne garantit qu'il convienne ou qu'il soit conforme aux dernières spécifications XHTML, car même Dreamweaver permet l'inclusion d'éléments obsolètes. De plus, toute génération automatique éloigne du travail de programmation et ne présente pas un avantage évident pour la création de pages, alors que ce peut être le cas dans un éditeur dédié à un langage de programmation, Java par exemple, et qui peut décharger le programmeur de tâches répétitives.

Outre le temps d'apprentissage et son prix élevé, je ne pense pas qu'il soit utile de faire appel à ce genre d'éditeur qui aurait tendance à vous rendre passif.

Signalons également dans cette catégorie, l'existence d'un nouvel éditeur qui se veut le concurrent libre et gratuit de Dreamweaver pour la création de pages web. Il s'agit de Nvu, téléchargeable sur le site http://www.nvu.com. Dans sa version actuelle, version 1.0, il est encore assez loin de son modèle mais ses débuts sont prometteurs car il offre des fonctionnalités avancées, comme un éditeur de styles CSS. C'est d'ores et déjà un bon produit dont il faudra suivre les évolutions futures. La figure 1-3 présente l'espace de travail de Nvu.

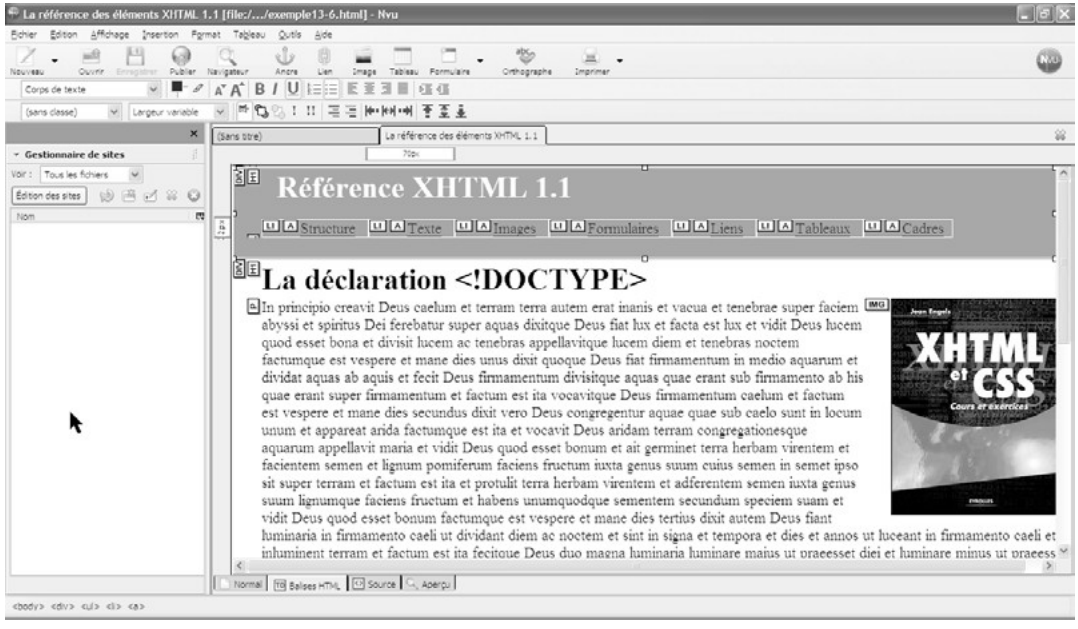


Figure 1-3
L'éditeur Nvu 1.0

Les éditeurs classiques

Entre le Bloc Notes Windows et Dreamweaver, certains éditeurs offrent un compromis, apportant à la fois une aide à la saisie en dispensant de taper soi-même le nom des éléments et en laissant la liberté du choix de ces éléments. Outre le fait qu'ils présentent souvent l'avantage d'être gratuits, ces éditeurs, très nombreux et disponibles en téléchargement, permettent de créer rapidement la structure d'un document en utilisant un squelette commun à toutes les pages XHTML. Nous donnerons la structure de base d'une telle page au chapitre 2. Il vous restera ensuite à inclure dans le corps de la page les différents éléments qui vont structurer son contenu. Si vous avez procédé à une analyse préalable sur le papier, comme il se doit avant tout codage, cette phase de travail d'inclusion sera très rapide. Parmi les nombreux éditeurs, j'ai choisi pour ma part EditPlus qui s'avère très pratique pour incorporer rapidement les différents éléments XHTML et CSS. La figure 1-4 montre l'aspect de l'environnement de travail qu'il fournit. La fenêtre de

l'éditeur est divisée en plusieurs zones. La zone ³ contient le code de la page. Comportant plusieurs onglets, la zone · permet de visualiser selon vos besoins du moment la liste des éléments XHTML 1.1, celle des propriétés CSS 2 ou un explorateur de fichiers. La zone » contient en plus des menus habituels un ensemble de boutons permettant l'inclusion instantanée du code correspondant à une icône (lien, titres, tableaux, etc.). Pour incorporer un élément XHTML complet, muni donc d'une balise d'ouverture, une balise de fermeture éventuelle ainsi que certains attributs importants, il vous suffit de placer le curseur à l'endroit désiré dans le code, puis d'effectuer un double-clic sur l'élément dans la zone de gauche (repère ·). Il en est de même pour créer une feuille de style : suffit d'un clic sur la propriété CSS voulue. La liste des éléments XHTML et CSS fournie par défaut avec EditPlus est actuellement plus large que celle des éléments conformes de XHTML 1.1, mais vous pouvez la personnaliser en enlevant des éléments ou en en ajoutant. Si vous utilisez par exemple très souvent tel attribut d'un élément, vous pouvez le rajouter, et il apparaîtra ensuite systématiquement quand vous choisirez cet élément, car la liste des éléments et de leurs attributs est enregistrée dans un fichier. Pour effectuer cette opération, effectuez un clic droit sur le nom d'un élément et choisissez Edit. Vous pouvez ensuite procéder à toutes les modifications désirées. Vous pourrez télécharger sur le site funxhtml.com/edit les fichiers xhtml11.ctl et css21.ctl et suivre la procédure indiquée sur le site pour obtenir dans EditPlus tous les éléments XHTML 1.1 et les propriétés CSS 2.1 qui sont actuellement conformes aux recommandations du W3C.

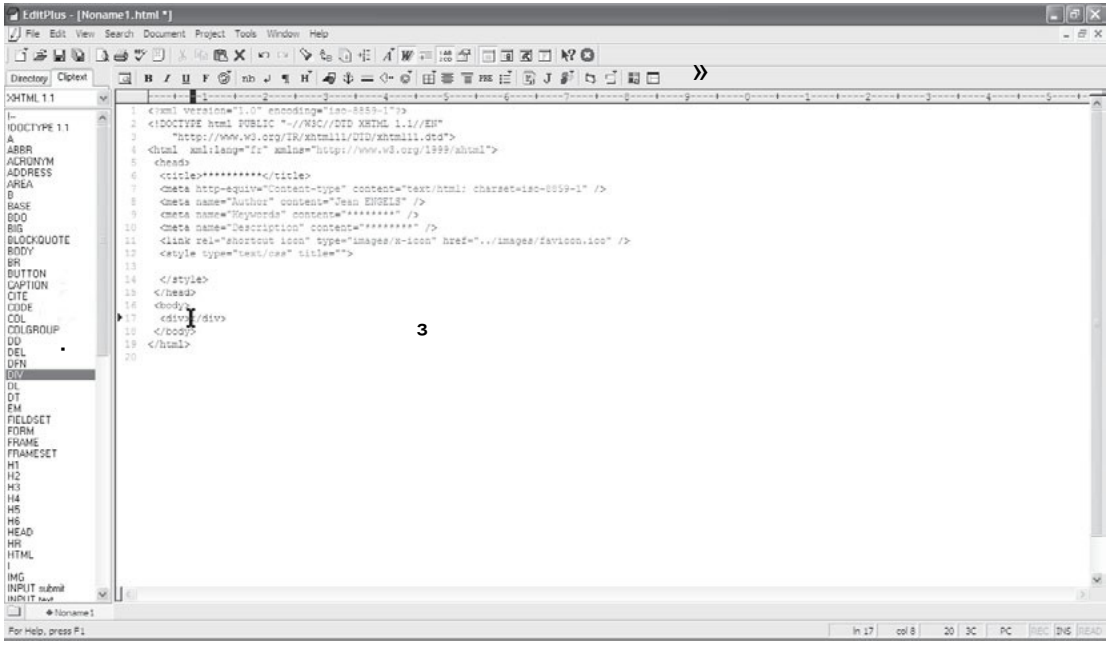


Figure 1-4
L'éditeur EditPlus

Tests et mise en place du site

Une fois que vous aurez créé l'ensemble des pages de votre site, il faudra vous assurer qu'elles ont un aspect identique ou du moins très semblable dans les différents navigateurs du marché et les résolutions d'écran afférentes. Ce dernier point doit d'ailleurs avoir fait l'objet d'un choix initial, en particulier si les dimensions choisies pour les différents éléments de la page sont définies de manière fixe (en pixels par exemple), et non en pourcentages, ce qui représente la meilleure solution. Choisir de créer des mises en page en ciblant le format le plus répandu (1024 par 768 pixels actuellement par exemple) peut produire des résultats désagréables si le poste client est en 800 par 600 pixels ou moins encore.

Si cette phase de test vous amène à constater des résultats divergents et indésirables, c'est sans doute que vous avez utilisés des fonctionnalités non encore prises en compte par un navigateur particulier, et le plus souvent il s'agira de propriétés CSS encore marginales dans Internet Explorer. Il vous appartiendra alors de renoncer à certaines d'entre elles au moins provisoirement, pour assurer l'universalité de vos pages si tel est votre objectif. Nous signalons à cet effet dans la deuxième partie de cet ouvrage les propriétés qui risquent de poser problème dans certains navigateurs.

Quand cette phase de test a été effectuée, il ne reste plus qu'à transférer l'ensemble des pages du site vers le serveur distant qui va les héberger pour les mettre à la disposition de tous. Vous devez pour cela utiliser un logiciel de transfert FTP (File Transfert Protocol).

Les hébergements gratuits

Les hébergeurs gratuits de sites personnels peuvent mettre à votre disposition des pages spécialisées permettant ce transfert via une interface web simple. Si vous disposez d'un nom de domaine sur un serveur dédié ou mutualisé, il vous faut généralement utiliser un logiciel FTP.

Il existe de nombreux logiciels de ce type, gratuits ou payants. Je recommande pour ma part d'utiliser FileZilla téléchargeable sur le site http://filezilla.sourceforge.net/, gratuit et très pratique. La figure 1-5 présente l'interface de FileZilla. Pour établir une connexion avec le serveur et acquérir les droits en vue d'effectuer les transferts de vos fichiers XHTML, de vos images ou des supports multimédias, vous devez connaître les paramètres de connexion qui sont fournis par l'hébergeur du site. Il s'agit des données suivantes :

- L'adresse du serveur ftp, par exemple ftp.funhtml.com (repère ³).
- Le nom d'utilisateur, par exemple funhtml (repère ·).
- Le mot de passe d'accès au site (repère »).

- Éventuellement, indiquez le port de communication (repère¿) (en général, il s'agit de la valeur 21).

Si ces paramètres sont correctement reconnus, vous avez accès à un gestionnaire de fichiers côté serveur similaire à celui que vous pouvez connaître dans Windows ou Linux (repère ´). Dans la partie gauche, vous trouvez le même type de gestionnaire de fichiers pour sélectionner les fichiers présents sur votre ordinateur (repère ²). Il vous suffit alors de réaliser une opération de glisser-déposer entre ces deux zones pour que le transfert commence. Sa durée dépend évidemment de la taille du fichier et de la vitesse de votre connexion. Le dossier principal côté serveur est nommé www. C'est dans ce dossier que le serveur ira chercher le fichier nommé index.html ou index.htm en réponse à la requête générale effectuée sur votre site sous la forme http://www.votresite.comC'est dans ce fichier que vous devez créer la page d'accueil du site (ou la page index.php si elle contient du code PHP). Dans ce dossier www, vous pouvez créer autant de sous-dossiers que vous le désirez. Si vous constituez par exemple un sous-dossier nommé xhtml, son contenu principal devra figurer dans un nouveau fichier nommé encore index.html, et il sera alors accessible directement par la requête http://www.votresite.com/xhtml . Dans les deux cas (répertoire principal ou sous-répertoire), si vous ne créez pas de fichier nommé index.html, l'utilisateur devra écrire une requête complète comprenant le nom du fichier auquel il veut accéder de la forme http://www.votresite.com/xhtml/page1.html·

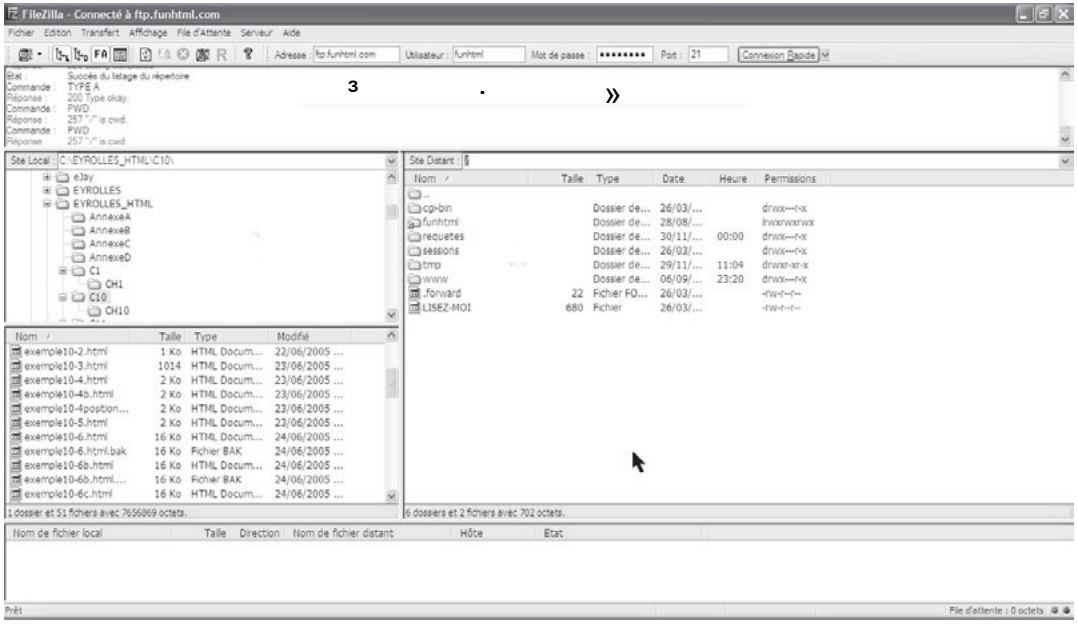


Figure 1-5

L'interface de FileZilla

Référencement du site

Une fois que vous avez créé votre superbe site conforme à XHTML et CSS, il reste encore une phase des plus importantes à accomplir, à savoir le faire connaître. À moins de disposer de moyens publicitaires conséquents, la fréquentation souhaitée ne se produira pas du jour au lendemain. Même si c'est l'adaptation du contenu à un besoin qui fait venir et revenir les internautes, il vous faut d'abord faire connaître l'existence de votre site.

Le moyen le plus simple, et généralement gratuit, est de le référencer dans les annuaires et moteurs de recherche les plus connus comme Google ou Yahoo afin qu'il soit bien placé en réponse à la recherche d'un internaute.

Des ouvrages entiers sont consacrés au référencement, apportant des conseils avisés pour augmenter la fréquentation d'un site. Je recommande le site http://www.abondance.com ou de se procurer l'ouvrage de son webmestre portant sur le référencement. Nous retiendrons cependant les quelques points suivants :

- Si vous achetez un nom de domaine – ce qui, lié à un hébergement mutualisé, est aujourd'hui possible à bon marché et « présente mieux » que l'hébergement gratuit offert par votre fournisseur d'accès dont les adresses sont très longues –, choisissez de préférence un nom court et facile à mémoriser. Les noms longs, et particulièrement s'ils sont composés de plusieurs mots, posent soucis à l'utilisateur qui se demande alors si les mots se suivent ou sont séparés par des tirets, d'où des interprétations divergentes et des erreurs. Certains moteurs de recherche affichent les sites répondant aux mêmes critères par ordre alphabétique et il vaut mieux que votre nom de domaine commence par « a » plutôt que par « z ».
- Les extensions .com (ou .fr en France) sont préférables à .org ou .net car ce sont les premières qui viennent à l'esprit des internautes s'ils ont oublié l'extension réelle. La notion de site commercial qui était liée à l'extension .com a désormais disparue et votre site ne peut avoir aucun aspect commercial et posséder cette extension.
- La définition d'un maximum de mots-clés rapportant objectivement le contenu de votre site est essentielle. Elle doit être réalisée à l'aide de l'élément :

```
<meta name="keywords" content="liste des mots clés" />
```

(voir le chapitre 2), qui est situé dans l'en-tête du document. Cette liste mérite toute votre attention car elle est utilisée par les moteurs de recherche pour indexer vos pages et mettre en adéquation la demande d'un internaute avec les sites qui lui correspondent. Les mots-clés se doivent de refléter fidèlement le contenu du site et les idées qui lui sont associées, et rien d'autre. Inutile par exemple de rechercher les mots les plus recherchés et de les inclure dans votre liste pour attirer du monde. Outre que vous risquez de tromper les internautes, vous pouvez surtout les décevoir si votre contenu ne correspond pas à leur attente. En revanche, n'hésitez pas à fournir une longue liste de mots-clés avec leurs variantes masculin/féminin et singulier/pluriel, ainsi que les mots dérivés qui peuvent correspondre à votre contenu. Vous augmenterez ainsi vos chances de figurer en bonne place dans les résultats des recherches effectuées par les internautes.

Les moteurs de recherche se basant aussi sur le contenu des pages, rien ne vous empêche de faire apparaître plusieurs fois dans le contenu les mots importants. Une astuce consistait à répéter ces mots dans la page et de les écrire de la même couleur que le fond de la page en créant des styles CSS appropriés et non plus les balises et les attributs utilisés pour définir les couleurs. Leur répétition est ainsi invisible dans la page, mais perçue par les robots qui analysent le texte.

- Créez la structure de vos pages en évitant l'emploi des cadres, autrefois très en vogue, mais dont les contenus sont mal indexés par les moteurs de recherche. L'emploi de XHTML et du positionnement CSS rend ces techniques caduques et des solutions de remplacement peuvent être mises à profit (voir le chapitre 13).
- Référencez votre site dans tous les principaux moteurs de recherche. La figure 1-6 montre par exemple la page de référencement de Yahoo qui permet de soumettre un site gratuitement.

Figure 1-6

La page de référencement de Yahoo

2

Structure d'un document XHTML

Avant de créer des pages web et de leur donner un contenu, nous allons déterminer une structure générale commune à toute page XHTML et qui soit en conformité avec les spécifications du W3C. En fonction des besoins, les codes des exemples 2-1, 2-2 et 2-3 serviront de base à la constitution de toutes nos pages. Il suffira donc de les copier dans votre éditeur préféré, puis de les compléter avec un contenu particulier pour chaque page.

Les éléments de base

Le langage XHTML consiste en une reformulation du langage HTML, conforme aux prescriptions XML. Tout document XHTML peut donc débuter de la même manière qu'un document XML par la déclaration suivante (exemple 2-1 repère ³) :

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

Elle sera suivie de la déclaration de la DTD suivante (exemple 2-1 repère ·) :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/
➥ OTR/xhtml11/DTD/xhtml11.dtd">
```

Vient ensuite l'élément racine <html> (exemple 2-1 repère ») qui inclut les éléments <head> (repère ¿) et <body> (repère ²). Chacun de ces éléments a un contenu et donc une balise d'ouverture et une balise de fermeture, <head> incluant au moins l'élément <title> (repère ´) et <body> ayant au moins un élément enfant ; ici, il s'agit de <h1> (repère ¶). La structure minimale d'un document conforme au standard XHTML 1.1 est

donc semblable à celle de l'exemple 2.1. Le fichier contenant ce code doit avoir une extension .html ou .htm.

L'extension .xml

Il est possible d'enregistrer un document XHTML sous l'extension .xml, tout comme pour un document purement XML. La page ainsi créée est affichée normalement dans les navigateurs modernes (Mozilla, FireFox, Netscape 7.2, Opera), mais pas dans Internet Explorer. Nous utiliserons par la suite l'extension .html.

Exemple 2-1. Structure minimale d'un document XHTML 1.1

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> XHTML et CSS </title>
 </head>
 <body>
  <!-- Tout le contenu de la page -->
  <h1>Le corps de la page minimale</h1>
 </body>
</html>
```

Nous retrouvons bien dans cet exemple la structure arborescente décrite au chapitre 1. L'élément racine, au sens XML du terme, est <html> et il inclut les éléments <head> et <body>. L'élément <head> contient l'élément <title>, qui est obligatoire, et l'élément <body> qui ne doit pas être vide (ce qui est évident), contient un titre de niveau 1 <h1> sur lequel nous reviendrons plus loin. Du point de vue hiérarchique, <html> est bien le parent ou l'ancêtre de tous les autres.

Les commentaires

Tout ce qui est contenu entre les symboles <!-- et --> est considéré par le navigateur comme des commentaires et n'est pas affiché dans la page, même s'ils se trouvent dans l'élément <body>. Comme pour tout langage de programmation, nous avons tout intérêt à commenter le code XHTML afin d'en faciliter la relecture a posteriori. Notez toutefois que les commentaires seront visibles par l'internaute si celui-ci choisit d'afficher le code source de la page dans son navigateur. Évitez donc d'y inclure des informations confidentielles et d'y faire figurer des informations privées.

La déclaration XML doit contenir le numéro de version XML dans l'attribut version. Sa valeur est actuellement 1.0 mais une version 1.1 de XML est en cours d'élaboration. Cette dernière version n'est actuellement pas reconnue par les navigateurs, mais devrait

l'être dans un avenir proche. L'attribut `encoding` de la déclaration XML contient le nom du jeu de caractères utilisé dans le document. Pour des pages rédigées en Europe occidentale, il a pour valeur `iso-8859-1`. La déclaration XML n'est pas obligatoire dans un document XHTML, mais son utilisation est conseillée, en particulier dans une page ne contenant que du code XHTML sans ajout de script de création de pages dynamiques. Dans le cas d'intégration de script PHP (comme c'est couramment le cas dans une page), nous ne pouvons plus utiliser cette déclaration car l'interpréteur PHP renvoie une erreur – cela peut varier selon sa configuration, mais si vous n'avez pas la haute main sur le serveur vous ne pouvez pas la modifier ; il faudra procéder à un test. En effet, ce type de script est inclus généralement entre les balises `<?php` et `?>`, donc selon une syntaxe similaire à celle de la déclaration XML, d'où la confusion du serveur PHP. Dans ce cas, nous n'utiliserons pas la déclaration XML et l'indication du jeu de caractères obligatoire est faite dans un élément `<meta />` inclus dans `<head>` sur lequel nous reviendrons. Pour éviter les problèmes d'interprétation divergente entre les différents navigateurs, nous utiliserons systématiquement la déclaration du jeu de caractères avec l'élément `<meta />` dans chaque document, que la déclaration XML y soit présente ou non. La structure minimale de ce type de page est donc celle de l'exemple 2-2. Notez que le fichier PHP a une extension propre, du type `.php` ou `.php5` par exemple, toujours en fonction de la configuration du serveur. Pour que le document XHTML, que le serveur va finalement envoyer au navigateur, soit conforme au standard, il faut que le premier script placé au début du document (repère ³) ne crée aucun code XHTML (il peut par exemple ne contenir que des fonctions) et que le second (repère ·) ne crée que du code XHTML conforme. En respectant ces conditions, il n'y a aucune limite à l'utilisation de scripts PHP à l'intérieur d'un document XHTML.

Exemple 2-2. Structure d'une page XHTML 1.1 contenant un script PHP

```
<?php³
//Ici du code PHP;
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html  xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> XHTML et CSS </title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
 </head>
 <body>
  <!-- Tout le contenu de la page -->
  <h1>Le corps de la page</h1>
  <?php ·
   echo "<h2>Ici du code PHP </h2>";
  ?>
 </body>
</html>
```

Comme indiqué au chapitre 1, nous avons choisi par principe d'utiliser systématiquement la DTD XHTML 1.1 qui représente la version la plus évoluée et la plus porteuse d'avenir. Cependant, ponctuellement, les navigateurs rencontrent encore quelques problèmes pratiques, par exemple pour afficher correctement le code créé avec cette DTD. Nous serons contraints en de rares occasions d'utiliser la DTD propre à la version XHTML 1.0 strict (exemple 2-3, repère ³) qui comporte peu de différences avec XHTML 1.1 et qui est conforme à l'esprit XHTML de séparation du contenu et de la présentation. La suite du document restera la même et obéira aux mêmes règles à de rares exceptions près. La structure de base d'un tel document sera alors construite sur le modèle de l'exemple 2-3.

Exemple 2-3. Structure d'un document XHTML 1.0 strict

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"> ³
<html  xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Page XHTML 1.0 strict</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" /> ·
 </head>
 <body>
  <h1>Page XHTML 1.0 strict</h1>
 </body>
</html>
```

Nous aborderons en détail au chapitre 8 une quatrième possibilité pour la création de la structure élémentaire d'une page. Cette dernière concerne la composition de pages divisées en cadres indépendants ayant chacun comme contenu un document XHTML différent. Ce type de conception est aujourd'hui un peu passé de mode et n'est plus possible avec la DTD XHTML 1.1, mais reste réalisable en XHTML 1.0 frameset pour les inconditionnels de cette méthode. Dans toute la suite de cet ouvrage et sauf indication contraire exceptionnelle, nous n'utiliserons que la DTD XHTML 1.1 avec la structure de base présentée à l'exemple 2-1, à laquelle nous ajoutons, comme indiqué plus haut, la déclaration du jeu de caractères dans l'élément `<meta>` lequel est présenté ci-après (repère ·) :

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
```

La déclaration DOCTYPE

Nous avons déjà indiqué que le code d'une page XHTML devait se conformer à des règles précises, définies dans une DTD spécifique. La déclaration `DOCTYPE` obligatoire dans tout document, précise le type de document qui va être créé et la DTD à laquelle il va se conformer. La première partie de la déclaration :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
```

indique le nom générique de la DTD. La partie `html` donne le nom de l'élément racine du document.

La seconde partie :

```
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"
```

précise l'adresse du fichier `xhtml11.dtd` qui contient la DTD par elle-même.

Pour utiliser les variantes de XHTML 1.0, nous avons le choix entre les déclarations DOCTYPE suivantes :

- ```
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/
  ➥xhtml1/DTD/xhtml1-strict.dtd"
  ```

qui correspond à la version XHTML 1.0 strict, antérieure à XHTML 1.1 et qui lui est pratiquement identique. C'est la plus rigoureuse des versions de XHTML 1.0.

- ```
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
  ➥TR/xhtml1/DTD/xhtml1-transitional.dtd"
  ```

Elle autorise l'utilisation d'éléments obsolètes présents dans les versions anciennes du langage HTML, tels que `<font>` ou `<center>` absolument déconseillés depuis des années. Cette DTD n'a été créé par le W3C que pour assurer la compatibilité avec des documents HTML élaborés avec HTML 4 Transitional, lui-même créé pour assurer la compatibilité avec HTML 3.2. Utiliser cette DTD aujourd'hui serait une aberration et une hypocrisie tendant à faire croire que l'on fait du XHTML. Elle est la négation même de l'esprit du XHTML, n'en déplaise à certains, et je ne la cite ici que pour mémoire en déconseillant formellement son usage.

- ```
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/
  ➥xhtml1/DTD/xhtml1-frameset.dtd">
  ```

C'est cette déclaration qui nous permettra de créer des pages comportant cadres, conception que nous abordons au chapitre 8.

L'élément racine <html>

C'est l'élément `<html>` qui est l'élément racine du document, au sens XML du terme. C'est donc lui qui est le parent de tous les autres, soit directement comme `<head>` et `<body>` soit indirectement par l'intermédiaire de ces derniers éléments. L'élément `<html>` est donc le conteneur de premier niveau placé en haut de la hiérarchie de tous les éléments du document. Il n'existe que deux éléments enfants de l'élément `<html>`. En XHTML 1.1 et 1.0, le contenu de cet élément est constitué de l'en-tête du document, introduit par la balise `<head>` et terminé par la balise `</head>` puis par le corps du document introduit par `<body>` et terminé par `</body>` comme nous pouvons le vérifier dans les exemples 2-1, 2-2 et 2-3.

L'élément racine possède trois attributs facultatifs :

- L'attribut `xml:lang` dont la valeur est un code de langue normalisée qui indique la langue utilisée par défaut dans la page. Cette valeur sera reconnue par les moteurs de recherche pour leur permettre d'indexer les pages du site en effectuant un tri par langue.
  ```

Elles n'apparaîtront dans Google par exemple que si l'utilisateur a choisi le bouton « France ».

- L'attribut `dir` qui indique le sens de lecture du texte de la page. Il peut prendre les valeurs `ltr` pour le texte qui se lit de gauche à droite (langues européennes) ou `rtl` pour le texte qui se lit de droite à gauche (langues orientales : hébreu, arabe).
- L'attribut `xmlns` qui précise l'adresse URL de l'espace de nom utilisé dans la page. Il prend la valeur fixe `http://www.w3.org/1999/xhtml` Un élément `<html>` complet tel que nous pouvons l'utiliser s'écrira donc :

```
<html xml:lang="fr" dir="ltr" xmlns="http://www.w3.org/1999/xhtml"
 <!--suite des éléments inclus --
</html>
```

En pratique pour des sites ayant un contenu dans une langue européenne, nous omettrons l'attribut `dir` .

L'en-tête d'un document : l'élément <head>

L'élément `<head>` englobe un certain nombre d'informations utiles au bon affichage de la page web. Ces informations sont contenues dans six éléments différents qui ont chacun un rôle bien déterminé. Il s'agit des éléments `<base>` `<link>`, `<meta>` `<script>`, `<style>` et `<title>` dont nous allons étudier les rôles respectifs.

Aucun d'eux n'a de répercussion directement visible dans la page et seul le contenu de l'élément `<title>` sera visible, non dans la page mais dans la zone de titre du navigateur. Le bloc d'en-tête a donc la structure suivante, dans laquelle seuls les éléments `<title>` et `<base />` ne doivent figurer qu'une seule et unique fois, les autres n'ayant pas de limites.

```
<head>
 <title>Titre de la page</title>
 <base href="http://www.monsite.com" />
 <link rel="shortcut icon" type="images/x-icon"  href="../images/favicon.ico" />
 <meta name="Author" content="Jean ENGELS" />
 <script  type="text/javascript">
  <!-- Scripts JavaScript -->
 </script>
 <style type="text/css" title="Styles de base">
  <!-- Styles CSS -->
 </style>
</head>
```

Remarquons d'emblée que seuls les éléments `<title>` , `<script>` et `<style>` ont un contenu, et donc une balise fermante.

L'élément `<head>` possède les attributs suivants :

- `xml:lang` qui a le même rôle que dans l'élément `<html>` ;
- `dir` qui indique le sens de lecture du contenu des sous-éléments de `<head>` ;

- profile dont la valeur est une URL qui pointe vers un fichier de définitions de méta-données telles que celles définies dans les éléments <meta />. À ce jour, aucun navigateur ne prend cet attribut en compte.

Nous allons maintenant détailler le rôle de chacun des éléments inclus dans <head>

L'adresse de base : l'élément <base />

Il s'agit d'un élément vide et n'a donc pas de balise de fermeture. L'information qu'il contient est donnée dans son unique attribut href dont l'utilisation est obligatoire. Le contenu de cet attribut est une URL qui fournit l'adresse de base de tous les fichiers utilisés dans la page quand leur adresse est transmise de manière relative. Si nous écrivons le code suivant :

```
<base href="http://www.funhtml.com/" />
```

le navigateur ira chercher une image dont l'URL est indiquée par /xhtml/images/monimage.gif à l'adresse :

```
http://www.funhtml.com/xhtml/images/monimage.gif
```

sur le serveur. L'élément <base /> possède également l'attribut commun id, qui ne peut servir qu'à modifier la valeur de l'attribut href au moyen d'un script JavaScript, selon la syntaxe suivante :

```
document.getElementById(id_element).href='valeur'
```

Les documents liés : l'élément <link />

Comme le précédent élément, il s'agit également d'un élément vide dont l'information est contenue dans ses attributs. Il permet d'établir un lien entre la page XHTML en cours et un autre document externe nécessaire à la page. Nous l'utiliserons particulièrement pour lier la page à une feuille de style CSS contenue dans un fichier ayant l'extension .css ou un script JavaScript contenu dans un fichier sous l'extension .js.

L'utilisation de l'élément <link /> crée l'incorporation virtuelle de ces documents dans le code de la page web. On parle d'incorporation virtuelle car la page se comportera comme si le code des fichiers externes faisait partie de la page, le contenu de ces fichiers n'étant pas visible, même en affichant le code source de la page.

La liaison avec les fichiers externes est déterminée par les attributs rel, rev, type, href, hreflang, media et charset.

- Les attributs rel et rev indiquent le nom de la relation à établir avec le fichier externe. Ils peuvent prendre les valeurs suivantes :
  - rel="stylesheet" si le fichier externe est une feuille de style ;
  - rel="alternate" si le fichier est une page alternative (de rechange, proposée aux visiteurs dans les navigateurs) ;
  - rel="alternate stylesheet" si le fichier est une feuille de style de remplacement, option proposée au visiteur dans certains navigateurs ;

- rel="shortcut icon" pour faire référence à l'icône identifiant le site et qui s'affiche devant l'adresse dans les navigateurs les plus conformes (mais pas Explorer 6 !) ;
- rel="start" si le fichier cible est la page d'accueil ;
- rel="previous" ou rel="prev" si le fichier désigné est la page précédente dans l'ordre normal de consultation du site ;
- rel="next" si le fichier est la page suivante dans l'ordre normal de consultation du site ;
- rel="index" si le fichier est la page contenant l'index du contenu du site (et non pas la page index.html) ;
- rel="end" si le fichier est la page finale dans l'ordre normal de consultation du site ;
- rel="help" si le fichier est la page d'aide ;
- rel="glossary" si le fichier cible contient un glossaire des termes utilisés dans le site ;
- rel="chapter" si le fichier contient un chapitre ;
- rel="section" si le fichier fait référence à une section d'un chapitre ;
- rel="appendix" si le fichier fait référence à l'appendice d'une page ;
- rel="contents" si le fichier contient la table des matières du site.

- L'attribut type précise le type du contenu du fichier externe. Il peut par exemple prendre les valeurs suivantes :

```
type = "text/css" pour une feuille de style CSS
type = "text/javascript" pour un script JavaScript
type = "text/html" ou "text/xml"
type = "images/x-icon" pour créer une icône
```

- L'attribut href contient l'adresse relative ou absolue de la ressource externe associée.
- L'attribut hreflang qui, comme xml:lang, prend pour valeur un code de langue, précise la langue utilisée dans le document cible.

L'attribut media précise le type de média concerné par le document externe. Nous l'utiliserons en particulier pour lier une feuille de style en précisant le type de média visé par les styles du document CSS. Les valeurs de l'attribut media sont : screen (écran d'ordinateur), print (imprimante), tty (télétype), tv (téléviseur), projections (rétro ou vidéo projecteur), handheld (PDA, téléphone), braille (terminal braille), aural (navigateurs oraux) et all (tous les médias). À titre d'exemple, nous pouvons écrire les codes suivants :

- Pour lier une feuille de style :

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

- Pour lier plusieurs feuilles de style en précisant le média concerné :

```
<link rel="stylesheet" type="text/css" href="styleWeb.css" media="screen"/>
<link rel="stylesheet" type="text/css" href="styleWeb.css" media="print"/>
```

- Pour lier un script JavaScript :

```
<link rel="script" type="text/javascript" href="code.js"/>
```

- Pour créer une icône dans la barre d'adresse :

```
<link rel="shorcut icon" type="images/x-icon" href="/fashion.icon"
```

- Pour créer un lien de dépendance vers un document :

```
<link rel="next" type="text/html" href="page3.html" />
<link rev="previous" type="text/html" href="page1.html" />
```

- Pour créer un lien vers la page d'accueil :

```
<link rel="start" type="text/html" href="index.html" />
```

Les navigateurs qui respectent les spécifications XHTML affichent une barre de navigation personnalisée qui peut devenir un véritable menu déroulant si on multiplie les liens qui permettent de passer rapidement d'une page à l'autre. La figure 2-1 montre le résultat obtenu par l'utilisation des éléments `<link />` de l'exemple 2-4 dans Mozilla.

Les commentaires

Il est toujours utile de commenter votre code XHTML, comme tout code informatique d'ailleurs, pour en permettre une meilleure compréhension, en particulier quand on souhaite le relire un certain temps après l'avoir écrit. Tout ce que vous écrirez comme commentaires sera ignoré par le navigateur, et vous pouvez donc vous exprimer librement. Pour écrire un commentaire, vous devez l'ouvrir avec les symboles : "`<!--` ", et le fermer avec les symboles " `-->`". N'oubliez pas de fermer vos commentaires, sinon tout ce qui suit sera encore interprété en tant que tel, provoquant une erreur. Par exemple, on aura :

```
<!-- Voici un commentaire HTML
qui peut comporter plusieurs lignes
```

Les méta-informations : l'élément <meta />

L'élément `<meta />` est également un élément vide pour lequel l'information est contenue dans ses attributs. Ces informations ne sont donc pas visibles dans la page mais elles sont destinées au serveur web, aux navigateurs et aux moteurs de recherche. Chaque information est identifiée par un nom et un contenu. Le nom de l'information est défini dans les attributs `name` ou `http-equiv` dont les rôles sont similaires, et la valeur associée est contenue dans l'attribut `content` sous la forme suivante :

```
<meta name="nom1" content="valeur1" />
<meta http-equiv="nom2" content="valeur2" />
```

Si nous utilisons l'attribut `http-equiv`, l'information indiquée dans l'attribut `content` sera présente dans les en-têtes HTTP envoyés par le serveur au navigateur sous la forme de couple nom/valeur.

La plupart des valeurs des attributs `name` et `http-equiv` sont des mots-clés. Nous allons nous arrêter sur la signification et l'utilité des plus courants d'entre eux.

- `name="author"` désigne le nom de l'auteur de la page sans pour autant créer un copyright.

```
<meta name= "author" content="Jean Engels" />
```

- `name="keywords"` Cette valeur est très importante pour le créateur de site car son incorporation dans un document sert à l'indexation des pages web par les moteurs de recherche et les annuaires. L'attribut `content` associé à `name` contient la liste des mots-clés que vous allez choisir comme les plus représentatifs du contenu du site. Chaque mot ou expression est séparé des autres par une virgule. Il n'est pas rare d'utiliser plusieurs lignes de mots-clés dans l'attribut `content`. L'utilisation de l'élément `<meta />` à cette fin est donc des plus utile car elle va permettre une mise en valeur de votre site qui apparaîtra dans les réponses fournies par les moteurs de recherche si vos mots-clés correspondent à la demande faite par un internaute. Il est important de bien choisir ses mots-clés pour qu'ils correspondent vraiment au contenu du site et d'en multiplier les variantes dans la liste de l'attribut `content`. On pourra retrouver le même mot au singulier et au pluriel, au masculin et au féminin. En revanche, il serait contre-productif d'utiliser les mots les plus demandés dans les moteurs de recherche sous prétexte d'attirer du public, alors qu'ils ne correspondent pas au contenu réel de votre site. Exemple de code :

```
<meta name="keywords" content="XHTML, HTML, XHTML 1.1, CSS 2, design web,
➥ création de sites />
```

- `name="description"` Dans le même ordre d'idée que la valeur précédente, il indique une brève description de l'information contenue dans le site. C'est cette description qui apparaît dans le moteur de recherche et il est donc essentiel qu'elle soit courte et correcte. Inutile de faire une description de 10 lignes alors que Google par exemple n'en affiche que deux. Il est également fortement recommandé d'utiliser cet élément `<meta />` car si vous ne donnez pas vous-même une description de la page, Google et les autres moteurs de recherche utilisent les premières lignes de votre page qui ne sont pas nécessairement les plus explicites. Exemple de code :

```
<meta name="Description" content="Le site du livre &raquo; XHTML et CSS &raquo;
➥ de Jean Engels Editions Eyrolles" />
```

- `name="robots"` Cette valeur permet de donner des directives aux robots des moteurs de recherche qui parcourent le Web automatiquement pour en indexer les pages. En fonction de la valeur de l'attribut `content`, vous pouvez choisir la manière dont vos pages seront indexées. Les valeurs possibles sont les suivantes :

  – `content="none"` ou `"noindex"`, si vous voulez empêcher l'indexation d'une page particulière.

- content="index", pour autoriser l'indexation de la page, ce qui est plus que recommandé.
- content="follow", pour autoriser l'indexation simultanée de la page concernée et des pages qui sont les cibles des liens contenus dans cette page. Le choix de cette valeur peut être risqué car il peut conduire à des indexations en chaîne si les pages cibles font de même. Si vos liens sont tous internes, n'hésitez pas à l'utiliser.
- content="nofollow", pour empêcher cette indexation des pages liées.

Exemple de code :

```
<meta name="robots" content="index" />
```

- name="revisit-after". Cette valeur précise aux robots des moteurs de recherche la périodicité des passages du robot sur le site. Nous l'utiliserons pour des sites dont le contenu évolue régulièrement. L'attribut content définit cet intervalle en nombres de jours. On écrira par exemple le code :

```
<meta name="revisit-after" content="15 days"/>
```

- http-equiv="refresh". Quand il est associé à l'attribut content qui a pour valeur un nombre de N secondes, son utilisation permet de forcer le navigateur à recharger la page toutes les N secondes. On procédera ainsi pour un site aux informations renouvelées très fréquemment, par exemple un site de cotation boursière, ou comme procédé pour afficher l'heure régulièrement par exemple à l'aide d'un script JavaScript ou PHP. Nous pouvons également utiliser cet élément pour rediriger automatiquement le visiteur vers une autre page du même site ou encore d'un autre site. On appliquera cette technique si le contenu du site a changé de nom de domaine. Pour cela, l'attribut content doit contenir le nombre N de secondes avant lequel la redirection sera effectuée, suivi d'un point-virgule (;) et de l'URL absolue de la nouvelle page. Par exemple, pour rediriger vers une autre page au bout de dix secondes nous écrivons le code suivant :

```
<meta http-equiv="refresh" content="10; http:/www.funphp.com/index/>
```

- http-equiv="Content-type". Cette valeur de l'attribut permet de définir simultanément le type du document et le jeu de caractères utilisés dans la page. Comme nous l'avons déjà signalé, il faut l'utiliser impérativement si le jeu de caractères n'a pas été précisé dans la déclaration XML ou si celle-ci est absente, comme dans le cas des pages PHP. Si cette déclaration n'est pas faite, l'utilisation de cet élément <meta /> est indispensable sinon le document n'est pas validé par le W3C. L'attribut content contient alors le type du document suivi d'un point-virgule puis du code du jeu de caractères. Nous avons par exemple le code suivant :

```
<meta http-equiv="Content-type" content="text/html;charset=iso-8859-1" />
```

- http-equiv="expires". Sauf spécification contraire effectuée dans le réglage des préférences de son navigateur, quand un utilisateur visite votre site, la page vue est mise en mémoire cache de son ordinateur. Elle s'affiche alors plus rapidement s'il y revient, même si le contenu du site a changé. Le visiteur peut bien sûr forcer le navigateur à

actualiser la page comme vous devez le faire pour tester les modifications apportées à une page lors des phases de tests. Pour forcer cette mise à jour, il faut définir une date dans l'attribut `content` associé à `http-equiv`. Quand la date du jour est postérieure à la date définie dans `content`, le navigateur recharge la page. Les dates sont données en anglais au format `"Mon, 11 Jul 2005 18:34:45 GMT"` pour le 11 juillet 2005 à 18 h 34 min 45 en temps GMT. Nous écrirons par exemple :

```
<meta http-equiv="expires" content="Mon, 11 Jul 2005 18:34:45 GMT"  />
```

Les scripts internes : l'élément <script>

Nous avons vu qu'il était possible de lier la page XHTML avec un fichier externe contenant du code JavaScript au moyen de l'élément `<link />`. Si cette solution correspond bien à l'aspect XHTML de séparation des fichiers ayant chacun un rôle différent, il est également possible de réaliser la même opération avec l'élément `<script>` qui sera alors vide. Le type de langage utilisé est précisé dans l'attribut `type` qui est obligatoire etcontient généralement la valeur `text/javascript` pour JavaScript mais qui peut être également `text/vbscript` pour le langage VBScript qui est cependant marginal vis-à-vis de JavaScript. Dans le cas d'un fichier externe, il faut utiliser l'attribut `src` qui donne l'URL du fichier externe du script qui possède l'extension `.js` pour JavaScript. Nous aurions par exemple le code suivant :

```
<script type="text/javascript" src="http://www.funhtml/xhtml/fichiercode.js">
</script>
```

Il est toujours possible d'incorporer du code JavaScript dans une page XHTML au moyen de l'élément `<script>` mais, à la différence de l'utilisation précédente, le code du script est inclut entre les balises `<script>` et `</script>`. Dans ce cas, seul l'attribut `type` est requis. Il est d'usage courant d'inclure tout le code JavaScript dans un commentaire situé à l'intérieur de l'élément `<script>` pour qu'il ne soit pas interprété par les navigateurs dépourvus d'interpréteur (si ça existe encore !) mais aussi et surtout si le visiteur a volontairement interdit l'exécution des scripts JavaScript.

Nous pouvons par exemple écrire le code suivant :

```
<script  type="text/javascript">
  <!--
  function debut()
  {alert('Bonjour XHTML');}
  -->
</script>
```

En plus de l'attribut `id` commun à la plupart des éléments, l'élément `<script>` possède également les attributs suivants dont le rôle est annexe par rapport aux précédents :

- `charset` pour préciser le jeu de caractères utilisés dans l'élément ;
- `defer` dont la seule valeur autorisée est `defer`. S'il est utilisé, l'interpréteur JavaScript du navigateur n'interprète pas le code contenu dans l'élément avant l'affichage de la page, ce qui rend l'affichage plus rapide. Cet attribut ne sera utilisé que si le code ne

contient pas d'instructions provoquant un affichage direct dans la page du type de celles réalisées au moyen de la fonction write() ;

- xml:space dont la seule valeur possible est preserve et qui a pour fonction de conserver intacts les différents espaces contenus dans le texte du code. Deux espaces consécutifs ou des tabulations sont donc conservés.

À la différence des autres éléments présents dans l'en-tête du document, l'élément <script> peut être utilisé également dans le corps du document, directement inclut entre les balises <body> et </body> ou indirectement dans de nombreux autres éléments inclus eux-mêmes dans <body> et dont la liste figure dans le tableau 2-1.

Tableau 2-1. Les éléments parents de l'élément <script>

a, abbr, acronym, address, area, b, bdo, big, blockquote, body, button, cite, code, dd, del, dfn, div, dt, em, fieldset, form, h1, h2, h3, h4, h5, h6, head, i, ins, kbd, label, legend, li, noscript, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var

Le code suivant utilise l'élément <script> et tous ses attributs :

```
<script type="texte/javascript" charset="iso-8859-1" defer="defer"
➥ xml:space="preserve">
<!-- Code JavaScript -- >
</script>
```

L'incorporation des styles : l'élément <style>

L'utilisation des styles CSS est étroitement liée à XHTML. Nous avons vu qu'ils pouvaient le plus souvent être écrits dans un fichier externe lié à la page avec l'élément <link /> . Cependant, comme pour les scripts, il est possible de les écrire directement dans le code XHTML en tant que contenu de l'élément <style>. Il n'est pas interdit d'écrire tous les styles dans cet élément mais cette possibilité sera réservée aux cas où les styles sont peu nombreux. Il est aussi envisageable de lier un fichier externe contenant les styles communs à toutes les pages d'un site avec <link /> et d'ajouter des styles particuliers à une page dans l'élément <style> de celle-ci.

En plus des attributs id, xml:lang, dir et title que nous avons déjà rencontrés, l'élément <style> possède les attributs suivants :

- L'attribut type, dont la présence est obligatoire, précise le type des feuilles de style utilisées. Pour les documents XHTML, il prend toujours la valeur text/css ;
- L'attribut media précise le type de média concerné par la feuille de style . Il est donc possible d'incorporer plusieurs éléments <style> dans l'en-tête de la page, chacun étant destiné à un média différent (voir l'exemple 2-4). Il prend les valeurs suivantes : screen (écran d'ordinateur), print (imprimante), tty (télétype), tv (téléviseur), projections (rétro ou vidéo projecteur), handheld (PDA, téléphone) braille (terminal braille), aural (navigateur oral) et all (tous les médias) ;

- L'attribut xml:preserve déjà rencontré, prend la valeur unique preserve si l'on désire que les espaces écrits dans le code CSS soient conservés ;
- Le code suivant définit une couleur de fond jaune pour la page et une couleur bleue pour le texte ;

```
<style type="text/css" media="screen"
  body{background-color:yellow;color:blue;}
</style
```

Nous reviendrons en détail sur l'écriture des styles dans la deuxième partie de cet ouvrage. Il nous faut simplement retenir ici la localisation des styles.

Le titre de la page : l'élément <title>

Chacun a pu remarquer dans son navigateur qu'avant l'affichage complet d'une page web, un titre apparaît dans la barre de titre située en haut de la fenêtre du navigateur. Ce texte est défini dans l'élément <title> qui est le seul dont la présence soit obligatoire dans l'élément <head>. Son contenu est un simple texte qui doit résumer le contenu de la page en une ligne maximum. Il est important de bien réfléchir à ce contenu car c'est aussi lui qui apparaîtra comme titre principal du site dans les moteurs de recherche. Il doit donc être accrocheur et bien correspondre à l'esprit de la page. Nous aurons par exemple le code suivant :

```
<title>Le site de XHTML 1.1 et CSS 2 </title>
```

L'élément <title> possède les attributs xml:lang, dir et id que nous avons déjà rencontrés.

L'exemple 2-4 crée une page en utilisant tous les éléments possibles de l'en-tête <head> dont <title> (repère ³), <base/> (repère ·), un grand nombre d'éléments <meta/> (repère ») et <link/> (repère ¿), les éléments <style> (repère ´) et <script> (repère ²). L'élément <body> (repère ¶) contient un minimum d'éléments pour obtenir un affichage. Nous reviendrons dans la section suivante sur cet élément et ce qu'il peut contenir.

Exemple 2-4. Les éléments de l'en-tête du document

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml" >
 <head>
   <!-- Élement <title> -->     ³
   <title> XHTML et CSS </title>
   <!-- Élément <base /> -->   ·
   <base href="http://www.funhtml.com/xhtml/" />
   <!-- Éléments <meta /> -->   »
```

```
<meta name="Author" content="Jean ENGELS"/>
<meta name="Keywords" content="XHTML, CSS 2, Web" />
<meta name="Description" content="Le site du livre &raquo; XHTML et CSS &raquo;
  ➥ de Jean Engels Editions Eyrolles" />
<meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
<meta http-equiv="Charset" content="iso-8859-1" />
<meta http-equiv="Content-Script-Type" content="text/javascript" />
<meta http-equiv="Content-Style-Type" content="text/css" />
<meta name="revisit-after" content="15 days" />
<meta name="robots" content="index,follow" />
<meta http-equiv="refresh" content="1250;URL=http://www.funhtml.com/xhtml" />
<!-- ***************  -->
<!-- Éléments <link /> -->     ¿
<link rel="shortcut icon" type="images/x-icon"
  ➥ href="/xhtml/images/favicon.ico" />
<link rel="stylesheet" type="text/css" href="/xhtml/C2/messtyles.css" />
<link rel="next" title="PHP 5" type="text/html"
  ➥ href="http://www.funhtml.com/php5" />
<link rel="start" title="Accueil" type="text/html"
  ➥ href="http://www.funhtml.com/xhtml" />
<link rel="previous" title="Document XML" type="text/xml"
  ➥ href="/xhtml/C2/exemple2-1.xml" />
<link rel="index" title="L'index" type="text/html"
  ➥ href="http://www.funphp.com/lindex.html" />
<link rel="chapter" title="Chapitre 1" type="text/html"
  ➥ href="http://www.funhtml.com/xhtml/ch1.html" />
<link rel="chapter" title="Chapitre 2" type="text/html"
  ➥ href="http://www.funhtml.com/xhtml/ch2.html" />
<link rel="chapter" title="Chapitre 3" type="text/html" h
  ➥ ref="http://www.funhtml.com/xhtml/ch3.html" />
<link rel="glossary" title="Glossaire" type="text/html"
  ➥ href="http://www.funphp.com/xhtml/glossaire.html" />
<link rel="section" title="Section 1" type="text/html"
  ➥ href="http://www.funphp.com/xhtml/sect1.html" />
<link rel="contents" title="Sommaire" type="text/html"
  ➥ href="http://www.funphp.com/xhtml/somm.html" />
<link rel="appendix" title="Appendice" type="text/html"
  ➥ href="http://www.funphp.com/xhtml/append.html" />
<!--  ************** -->
<!-- Élément <style> -->     ´
<!--  <style type="text/css" media="screen">
  body {background-color:yellow;color:blue;}
</style>-->
<!--  <style type="text/css" media="print">
  body {background-color:white;color:green;}
</style>-->
<!--  *************** -->
<!-- Élément <script> -->    ²
```

```
  <script  type="text/javascript">
   function debut()
   {alert('Bonjour XHTML');}
  </script>
  <script type="text/javascript" src="http://www.funhtml/xhtml/fichiercode.js">
    ➥ </script>
 </head>
 <body onload="debut()">¶
  <!-- Tout le contenu de la page -->
   <h1>Le corps de la page</h1>
  <p>
    <img src="/xhtml/images/php5_logo.gif" alt="PHP 5" />
  </p>
 </body>
</html>
```

La figure 2-1 montre le résultat obtenu, et en particulier la barre de navigation du site dans Mozilla, grâce à l'utilisation des différents éléments<link/> dans l'en-tête du document de l'exemple 2-4. Mozilla est le seul navigateur, avec Opera (dans une moindre mesure), à exploiter ces éléments pour créer une barre de navigation.

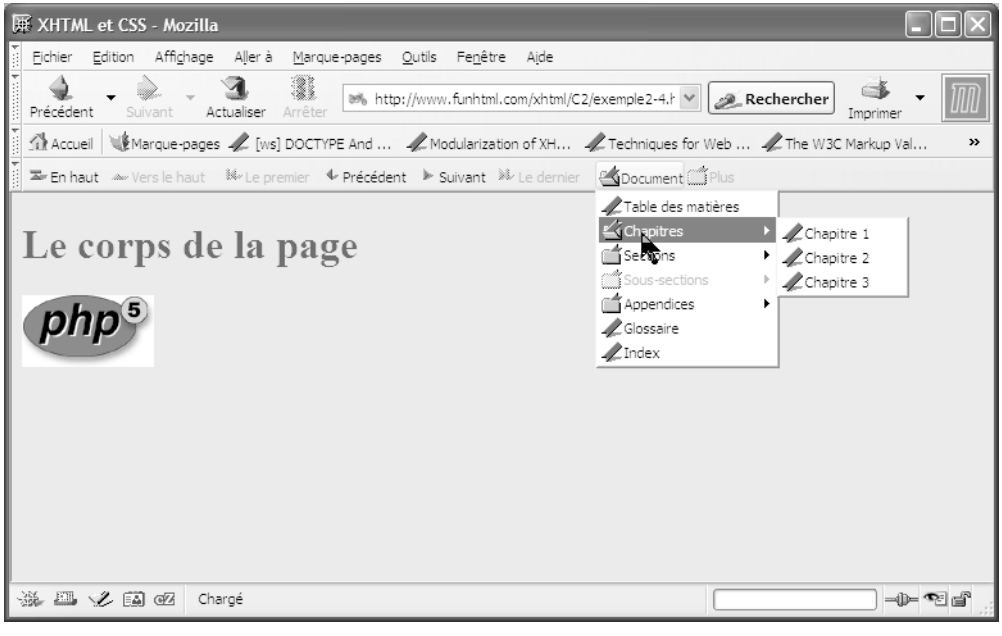


Figure 2-1

L'utilisation des éléments <link /> de l'en-tête

Le corps du document : l'élément <body>

La plupart des éléments de l'en-tête <head> que nous venons d'aborder ont un contenu invisible dans le navigateur. L'élément <body> au contraire est le conteneur de l'ensemble des éléments textuels et graphiques d'une page web. En revanche, et contrairement à beaucoup de vieilles et mauvaises habitudes de codage non conformes, le corps d'un document XHTML délimité par les balises <body> et </body> ne peut pas contenir n'importe quel autre élément XHTML existant, pas plus qu'il ne peut contenir quoi que ce soit qui ne soit délimité par un autre élément. En particulier, il est interdit d'y inclure directement du texte brut, comme on le voit encore trop souvent. Le tableau 2-2 donne la liste exhaustive de tous les éléments qui peuvent être inclus directement dans l'élément <body>. Tout autre élément y est interdit et le respect de cette liste est une convention primaire de validation d'un document XHTML. C'est donc le premier contrôle auquel vous devez procéder.

Tableau 2-2. L'ensemble des éléments enfants de l'élément <body>

address, blockquote, del, div, dl, fieldset, form, h1, h2, h3, h4, h5, h6, hr, ins, ol, noscript, p, pre, script, table, ul

Les éléments ne figurant pas dans cette liste doivent d'abord être inclus dans des éléments de cette liste (c'est le cas par exemple de l'élément <img/>). Chacun de ces éléments de premier niveau dans la hiérarchie des éléments au sens XML du terme peut à son tour contenir un certain nombre d'autres éléments, et ainsi de suite. À chacun de ces éléments correspond donc un certain nombre d'éléments enfants autorisés qu'il nous faut respecter. Au fur et à mesure que nous aborderons les différents éléments XHTML, nous donnerons systématiquement le contenu qu'il est possible de leur donner. Vous trouverez en annexe A la liste de tous les éléments de la DTD XHTML 1.1, leurs éléments enfants et parents autorisés et leurs attributs. À titre indicatif, le code de l'exemple 2-5 crée un document XHTML de base contenant tous les éléments admis dans le corps d'un document par ordre alphabétique.

Exemple 2-5. Les éléments admis dans le corps d'une page XHTML 1.1

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html  xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les éléments du corps de la page</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <meta name="Author" content="Jean ENGELS" />
  <meta name="Keywords" content="XHTML, CSS , Web" />
  <meta name="Description" content="Le site de XHTML 1.1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="">
```

```
 </style>
</head>
<body>
 <address>Contact : xhtml@funhtml.com</address>
 <blockquote>
   <h4>titre h4</h4>
 </blockquote>
 <del>L'ancienne version HTML 3.2</del>
 <div>bloc div</div>
 <dl>
  <dt>Liste de d&eacute;finition</dt>
 </dl>
 <fieldset>
   <legend>legende</legend>texte<button>bouton</button>
 </fieldset>
 <form action="script.php">
  <fieldset><legend>commentaires</legend><br />
   <textarea cols="30" rows="5">texte</textarea>
  </fieldset>
 </form>
 <h1>titre de niveau 1</h1>
 <h2>titre de niveau 2</h2>
 <h3>titre de niveau 3</h3>
 <h4>titre de niveau 4</h4>
 <h5>titre de niveau 5</h5>
 <h6>titre de niveau 6</h6>
 <ins>nouvelle version html 4.01</ins>
 <ol>
  <li>Item 1</li>
  <li>Item 2</li>
 </ol>
 <p>paragraphe 1: de l'action de la pluie sur la vie des grenouilles....</p>
 <pre>
   code javascript:
    document.write("boucle");
     for(i=0;i&lt;5;i++)
     {
     document.write( i);
     }
 </pre>
 <noscript><p>votre navigateur ne reconnait pas javascript</p></noscript>
 <script type="text/javascript">
 <!--
  document.write("boucle javascript <br />");
  for(i=0;i<5;i++){
  document.write( i,'  ')}
 // -->
 </script>
```

```
<table border="1">
 <tbody>
  <tr><td>Cellule de tableau</td></tr>
 </tbody>
</table>
<ul>
 <li>Item 1</li>
 <li>Item 2</li>
</ul>
</body>
</html>
```

Les figures 2-2 et 2-3 présentent les affichages obtenus dans un navigateur après l'exécution du code de l'exemple 2-5.

Figure 2-2

L'ensemble des éléments enfants de <body> (première partie)

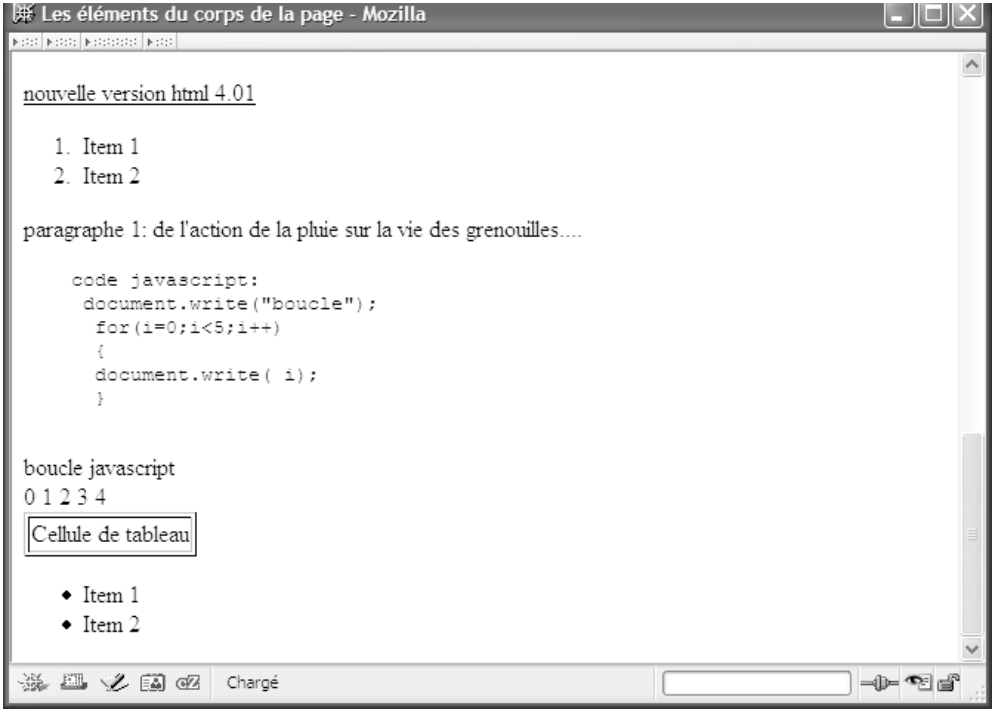


Figure 2-3
L'ensemble des éléments enfants de <body> (deuxième partie)

Dans tous les chapitres de la première partie de cet ouvrage, nous allons aborder l'ensemble des éléments incorporables dans l'élément <body> et nous y procéderons successivement, en fonction de leur rôle, pour créer toutes les fonctionnalités utiles dans un site.

Exercices

Exercice 1 : La déclaration XML est-elle obligatoire ? Où doit-elle être placée ? Quelles informations contient-elle ? Dans quel cas pose-t-elle des problèmes ?

Exercice 2 : Que faut-il faire pour que le code soit validé si la déclaration XML est omise ?

Exercice 3 : À quel emplacement est définie la DTD utilisée dans le document ? Quelles sont les différentes DTD utilisées dans un document XHTML 1 ?

Exercice 4 : Où est déclaré le nom de l'élément racine du document ? Quel est son rôle et son nom ?

Exercice 5 : Quels sont les éléments enfants de l'élément <html> ?

Exercice 6 : À quoi sert l'élément <head> ? Quels sont ses éléments enfants ? Peuvent-ils être employés plusieurs fois dans le même document ?

Exercice 7 : Quel élément est obligatoire dans l'élément <head> ? À quoi sert-il ?

Exercice 8 : Écrivez le code nécessaire à la liaison d'une feuille de style avec un document.

Exercice 9 : Écrivez le code nécessaire à la liaison d'un document externe contenant des scripts JavaScript avec un document.

Exercice 10 : Comment déclarer les mots-clés associés au site ? Quelle est l'utilité de cette déclaration ? Écrivez en un exemple.

Exercice 11 : Quel est le rôle de l'élément <base /> ? Écrivez-en un exemple.

Exercice 12 : Peut-on écrire le code suivant dans une page ?

```
<body>
 Bienvenue dans notre site
 <h1>Le site du XHTML et de CSS2</h1>
</body>
```

Exercice 13 : Peut-on inclure les éléments <img /> <form> et <li> dans l'élément <body> ?

Exercice 14 : Écrivez un script qui affiche un message d'alerte quand l'utilisateur arrive sur le site.

Exercice 15 : Écrivez le code CSS suivant à l'endroit adéquat :

```
body {background-color:white;color:green;font-size :20px}
```

Incluez ensuite un texte dans la page et testez le résultat. Vous devez obtenir un fond rouge, un texte bleu avec des caractères de 20 pixels.

3

Créer du texte et des listes

Créer une page web, c'est d'abord y incorporer du texte, et ce depuis les origines du Web qui ne contenait rien d'autre. Nous avons à notre disposition nombre d'éléments XHTML grâce auxquels on peut incorporer un contenu textuel dans une page. La diversité des éléments qui datent des versions HTML permet une structuration du texte qui pourrait être réalisée avec beaucoup moins d'éléments et l'utilisation conjointe de styles CSS. Cette structuration vise à organiser les différentes parties du texte et à attirer l'attention des visiteurs sur les points importants. Il est donc nécessaire que le contenu textuel soit structuré en titres de différents niveaux, en grandes divisions puis en paragraphes. Les listes constituent également un moyen de structurer l'information dans certains cas particuliers.

Les titres

Dans une page web, c'est en priorité les titres qui identifient les grandes sections de texte, comme nous le faisons dans ce livre. Les titres sont contenus dans les éléments `<h1>…</h1>` pour les titres de premier niveau, à `<h6> … </h6>` pour les titres de plus bas niveau. Entre ces extrêmes nous pouvons utiliser les titres `<h2>` `<h3>` `<h4>` et `<h5>` et nous noterons `<hN>` pour désigner l'ensemble de ces éléments. Si aucun style personnalisé ne leur est donné, les navigateurs affichent les titres dans des polices de tailles dégressives pour les éléments `<h1>` (le plus grand) à `<h6>` (le plus petit). Les éléments de titre faisant partie des éléments de bloc, ils sont automatiquement suivis d'un saut de ligne, après la balise de fin `</h1>` par exemple. On peut cependant adapter cette particularité en définissant un style personnalisé pour ces éléments. En tant que bloc, un élément `<hN>` peut contenir bien sûr du texte, ce qui en constitue l'utilisation la plus fréquente, mais il est utile de

savoir qu'il peut aussi inclure tous les éléments « en ligne ». Il peut par ailleurs être inclut dans les blocs et les cellules des tableaux. Les tableaux 3-1 et 3-2 donnent la liste exhaustive des éléments enfants et parents des éléments de titre <hN>

Tableau 3-1. Liste des éléments enfants de <h1> … <h6>

Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var

Tableau 3-2. Liste des éléments parents de <h1> … <h6>

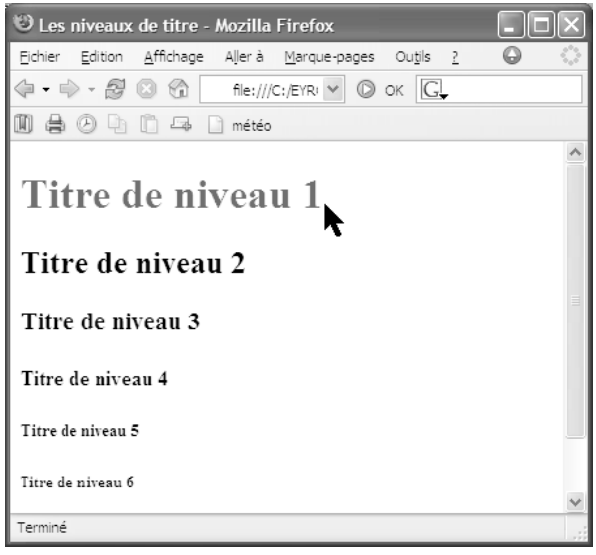
blockquote, body, button, dd, del, div, fieldset, form, ins, li, map, noscript, object, td, th

Les éléments <h1>… <h6> possèdent l'ensemble des éléments communs vus au chapitre 2. Nous utilisons le plus souvent id, class, title, et par exemple les gestionnaires d'événements onmouseover et onmouseout pour changer leur style, par exemple la couleur du titre quand le curseur le survole et la rétablir quand il quitte la zone du titre. Sans utiliser aucun style, les titres de niveaux 1 à 6 sont donc affichés avec des polices de tailles décroissantes comme le montre la figure 3.1, obtenue par l'affichage de la page de l'exemple 3-1 qui crée six titres de niveaux décroissants (repère ³ à ²) et utilise les deux gestionnaires d'événements onmouseover pour écrire le titre de niveau 1 en rouge et onmouseout pour le remettre en noir, sa couleur par défaut (repère ³).

Exemple 3-1. Les différents niveaux de titre et leur style par défaut

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les différents niveaux de titre</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <h1 onmouseover="this.style.color='red';"
   onmouseout="this.style.color='black'">Titre de niveau 1</h1>³
  <h2>Titre de niveau 2</h2>·
  <h3>Titre de niveau 3</h3>»
  <h4>Titre de niveau 4</h4>¿
  <h5>Titre de niveau 5</h5>´
  <h6>Titre de niveau 6</h6>²
 </body>
</html>
```

Figure 3-1
Les différents niveaux de titre



Signalons encore une fois qu'il ne faut pas choisir un niveau de titre en fonction de sa taille physique par défaut telle qu'elle est représentée à la figure 3-1, mais en fonction de la sémantique sous-jacente qui lui est attachée. Si une page ne comporte que des titres d'un seul niveau, il est conseillé de n'utiliser que des titres <h1> sans tenir compte de la taille par défaut et non pas des titres <h2> par exemple, au motif que la taille convient mieux. La taille à laquelle on veut afficher réellement ces titres dans la page sera déterminée en écrivant un style CSS. Dans le code suivant, nous organisons le contenu d'une page en utilisant quatre niveaux de titres selon une structure similaire à celle d'un livre. Le niveau <h1> est celui des parties du livre (repères ³ et), le niveau <h2> est celui des chapitres (repères · et ¾), le niveau <h3> est celui des différentes sections (repères » , ² , µ et) et enfin le niveau <h4> est celui des titres des différents paragraphes (repères , ´ , ¶ , º , ¹ , , et). Les niveaux suivants peuvent par exemple être utilisés pour les titres des exemples et des figures. Chacun peut ensuite être doté d'un style CSS approprié aux goûts du concepteur de la page mais la structure reste identique quelle que soit la présentation adoptée.

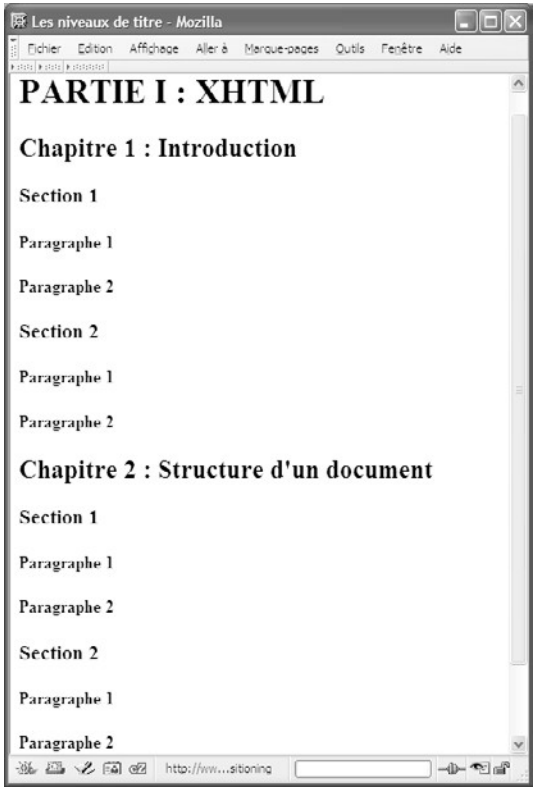
```
<body>
<h1>PARTIE I : XHTML</h1>
 <h2>Chapitre 1 : Introduction</h2>
  <h3>Section 1</h3>
   <h4>Paragraphe 1</h4>
   <h4>Paragraphe 2</h4>
  <h3>Section 2</h3>
   <h4>Paragraphe 1</h4>
   <h4>Paragraphe 2</h4>
```

```
<h2>Chapitre 2 : Structure d'un document</h2>
 <h3>Section 1</h3>
  <h4>Paragraphe 1</h4>
  <h4>Paragraphe 2</h4>
 <h3>Section 2</h3>
  <h4>Paragraphe 1</h4>
  <h4>Paragraphe 2</h4>
<h1>PARTIE II : CSS 2</h1>
 <!—Suite de la structure -->
 </body>
```

La figure 3-2 donne une représentation de cette structure avec les styles par défaut d'un navigateur Mozilla.

Figure 3-2

Structuration d'une page avec des titres



Les divisions de la page

Le corps d'un document, contenu dans l'élément `<body>` peut être divisé en différentes parties qui vont constituer les différents blocs de la page. Ces divisions permettent de bien structurer l'information contenue dans une page.

Les paragraphes : l'élément

Comme dans un traitement de texte, le contenu d'une page peut être divisé en différents paragraphes. Chaque paragraphe sera par défaut précédé et suivi d'un saut de ligne pour marquer la séparation avec le contenu précédent et suivant. Chaque paragraphe doit être contenu dans l'élément et donc délimité par les balises et Chaque paragraphe peut bien sûr contenir du texte mais également tous les éléments en ligne comme des images, des objets multimédia ou des composants de formulaire si l'élément est inclut lui-même dans un formulaire (voir le chapitre 7, *Créer des formulaires*). Les tableaux 3-3 et 3-4 donnent respectivement la liste des éléments enfants et parents d'un paragraphe

**Tableau 3-3. Liste des éléments enfants de l'élément **

Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var

**Tableau 3-4. Liste des éléments parents de l'élément **

blockquote, body, button, dd, del, div, fieldset, form, ins, li, map, noscript, object, td, th

L'élément possède l'ensemble des attributs communs et aucun autre attribut particulier. L'exemple 3-2 montre une utilisation de titres et de paragraphes pour structurer une page. La figure 3-3 montre le résultat obtenu pour cette page.

Figure 3-2. Structuration du texte en paragraphes

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les paragraphes</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <h1>Chapitre 1</h1>
  <h2>La création du monde</h2>
  <p>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
  ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas dixitque
  ➥ Deus fiat lux et facta est lux et vidit Deus lucem
  </p>
```

```
<p>Et vocavit Deus aridam terram congregationesque aquarum appellavit maria
➥ et vidit Deus quod esset bonum et ait germinet terra herbam virentem et
➥ facientem semen et lignum pomiferum faciens fructum iuxta
</p>
<h2>Le huitième jour</h2>
<p>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
</p>
</body>
</html>
```

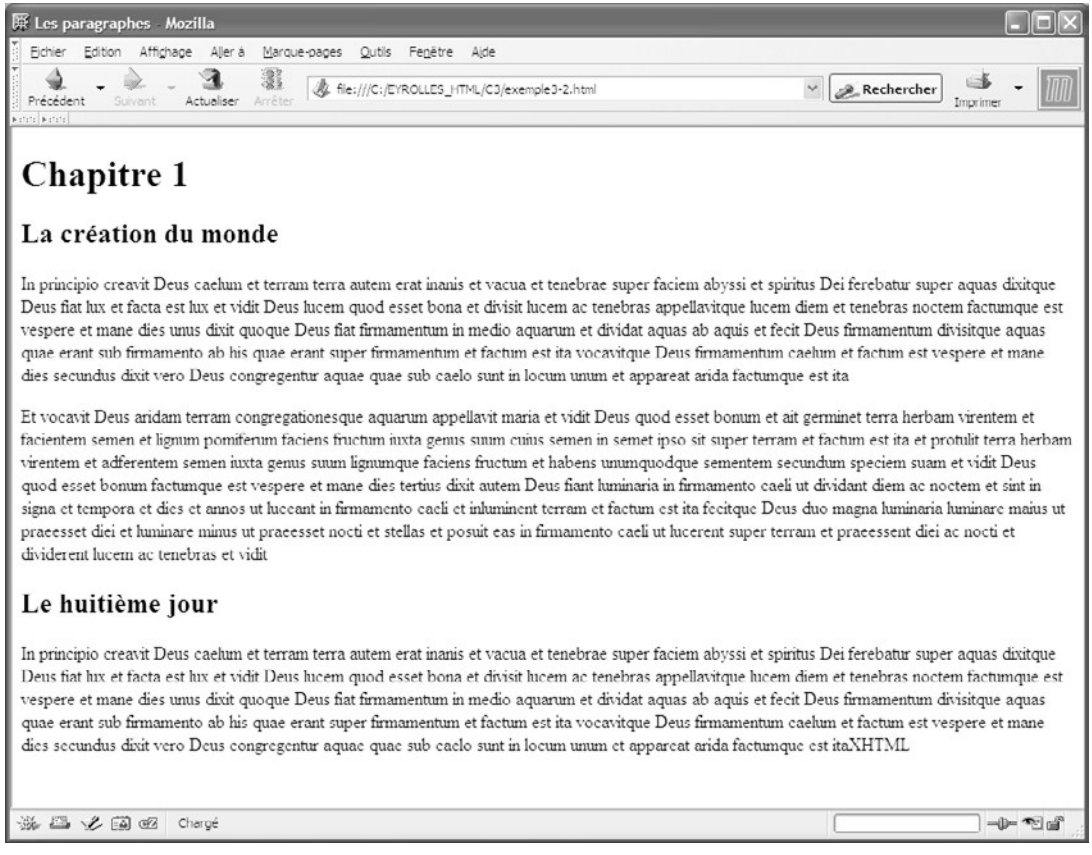


Figure 3-3
Les paragraphes

Notons que si l'utilisateur redimensionne la fenêtre du navigateur, la largeur des paragraphes s'adapte à cette nouvelle taille. Le texte reste totalement visible mais sur davantage de lignes, sans avoir à utiliser la barre de défilement horizontale.

Les divisions de la page : l'élément

Comme l'élément <p> l'élément peut être directement inclut dans le corps du document <body> Il crée une division de la page. Ce type de division permet de grouper dans un seul bloc un ensemble composé soit de texte soit d'éléments inclus, auxquels on pourra appliquer globalement des styles particuliers. À la différence des paragraphes, une division créée avec permet d'inclure une très grande variété d'éléments XHTML, comme du texte brut, les éléments en ligne et la totalité des éléments de niveaux bloc comme l'autorise <body> mais également les titres, les listes et les formulaires, ce qui est interdit dans un paragraphe. Les tableaux 3-5 et 3-6 donnent respectivement la liste de tous les éléments enfants et parents de l'élément C'est donc un élément très riche qui se prête bien à la création de la structure d'une page en grands blocs distincts auxquels il est possible d'appliquer par la suite des styles propres et des positions précises (voir le chapitre 13, *Créer une mise en page : le dimensionnement et le positionnement*).

**Tableau 3-5. Liste des éléments enfants de l'élément **

Texte, a, abbr, acronym, address, b, bdo, big, blockquote, br, button, cite, code, del, dfn, div, dl, em, fieldset, form, h1, h2, h3, h4, h5, h6, hr, i, img, input, ins, kbd, label, map, object, ol, p, pre, q, table, samp, script, select, small, span, strong, sub, sup, textarea, tt, ul, var

**Tableau 3-6. Liste des éléments parents de l'élément **

blockquote, body, dd, del, div, fieldset, form, ins, li, map, noscript, object, td, th

L'élément admet l'ensemble des attributs communs dont id , class , title , dir , xml:lang , qui sont les plus utilisés. Une des applications les plus courantes des divisions consiste à créer des calques auxquels on applique des styles de positionnement permettant de remplacer avantageusement les cadres. Il nous sera, par exemple, possible de diviser la page web en zones bien définies ayant chacune un rôle précis. Comme nous le verrons dans la deuxième partie de cet ouvrage, l'application de styles à des divisions permettra de créer facilement un en-tête, une zone de menu, une zone de contenu et un pied de page distincts qu'il sera possible de répéter dans chaque page (voir le chapitre 13). Il nous sera également possible à l'aide de scripts simples de créer des effets graphiques sur des divisions en les faisant apparaître ou disparaître en fonction des actions du visiteur. Notons aussi que, contrairement aux paragraphes <p> la fin d'une division n'entraîne pas un saut de ligne par défaut, mais seulement un retour à la ligne. Les contenus des différentes divisions peuvent donc se succéder sans rupture. L'exemple 3-3 reprend le même contenu que l'exemple 3-2 en remplaçant les éléments <p> par des divisions (repères ³ , · et »). On notera la différence d'affichage obtenue entre la figure 3-3 (paragraphes <p>) et la figure 3-4 (blocs). Nous ne retrouvons plus entre deux blocs le saut de ligne qui existe entre deux paragraphes. Dans cet exemple, la réalisation d'une séparation du texte entre les deux premiers blocs ne se justifie que si l'on veut leur appliquer des styles différents par la suite.

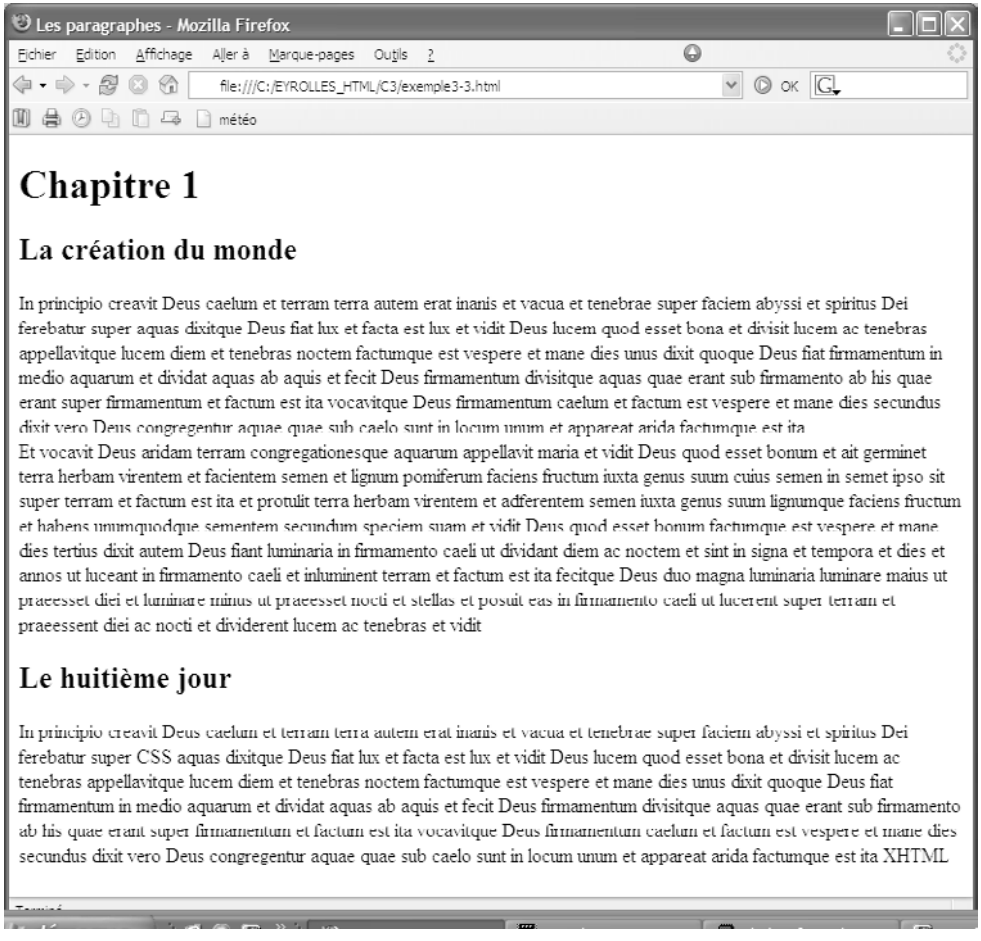


Figure 3-4
*Les divisions en blocs *

**Exemple 3-3. Les divisions en blocs **

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les divisions div</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <h1>Chapitre 1</h1>
```

```
<h2>La création du monde</h2>
<div>In principio creavit Deus caelum et terram terra autem erat inanis
➥ et vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur super
➥ aquas dixitque Deus fiat lux et facta est lux et vidit Deus lucem …
</div>
<div>Et vocavit Deus aridam terram congregationesque aquarum appellavit maria
➥ et vidit Deus quod esset bonum et ait germinet terra herbam virentem et
➥ facientem semen et lignum pomiferum faciens fructum iuxta …
</div>
<h2>Le huitième jour</h2>
<div>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas …
</div>
</body>
</html>
```

Les divisions de bloc locales

Le corps du document peut contenir d'autres éléments de niveau bloc qui étaient destinés en HTML à obtenir un rendu particulier, essentiellement pour des parties de texte. Compte tenu de l'utilisation généralisée des styles, ces éléments perdent aujourd'hui de l'importance car nous pouvons obtenir le même résultat en appliquant un style spécialisé à un élément `<div>` par exemple. Ils resteront cependant utiles à ceux qui veulent obtenir la présentation par défaut de chacun d'entre eux dans les navigateurs courants.

Les blocs de citations : l'élément <blockquote>

L'élément `<blockquote>` sert à créer une division de petite taille dans le corps d'une page. À l'origine, il était destiné à contenir des blocs de citations, comme un petit poème ou une note. Son contenu est par défaut mis en évidence au moyen d'un affichage en retrait par rapport aux bords gauche et droit de la page. Chaque élément est suivi d'un saut de ligne, tout comme pour les paragraphes. Notons bien qu'aucun contenu textuel ne peut y être inclut directement, pas plus que les images d'ailleurs. Il faudra au préalable inclure dans cet élément par exemple un paragraphe `<p>` puis le texte et les images. Les tableaux 3-7 et 3-8 donnent respectivement la liste des éléments enfants et parents de l'élément `<blockquote>` qu'il faut respecter.

Tableau 3-7. Éléments enfants de l'élément <blockquote>

address, blockquote, del, div, dl, fieldset, form, h1, h2, h3, h4, h5, h6, hr, ins, noscript, ol, p, pre, script, table, ul

Tableau 3-8. Éléments parents de l'élément <blockquote>

blockquote, body, button, dd, del, div, fieldset, form, ins, li, map, noscript, object, td, th

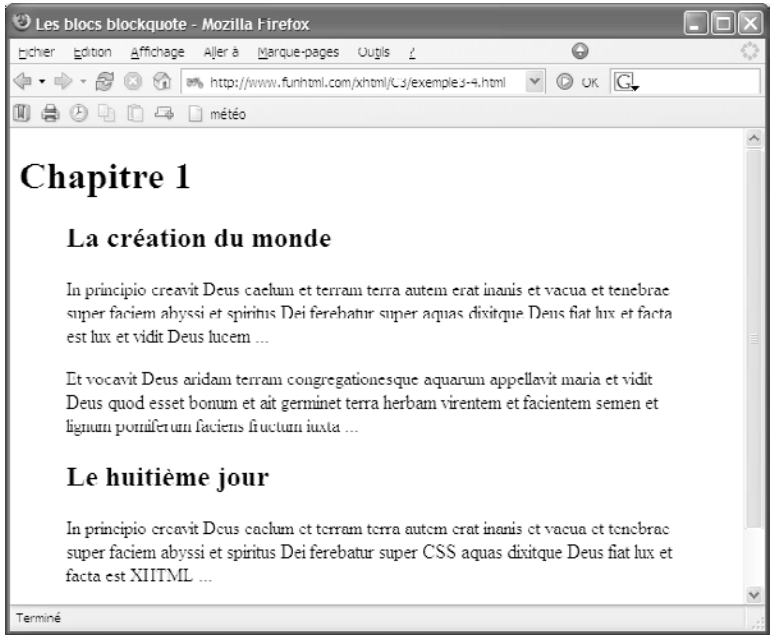
L'exemple 3-4 reprend encore le contenu des exemples précédents, inclus cette fois dans des éléments <blockquote>. Le premier (repère ³) contient à la fois un titre de niveau 2 (repère ·) et un bloc (repère »). Le deuxième (repère ¿) ne contient qu'un bloc (repère ´) et le dernier (repère ²) contient un titre (repère ¶) et un paragraphe <p> (repère º). La figure 3-5 montre le type de présentation que fournissent par défaut les navigateurs pour l'élément <blockquote>.

Exemple 3-4. Les blocs <blockquote>

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les blocs blockquote</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <h1>Chapitre 1</h1>
  <blockquote>³
   <h2>La création du monde</h2>
   <div>In principio creavit Deus caelum et terram terra autem erat inanis
     ➥ et vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur super
     ➥ aquas dixitque Deus fiat lux et facta est lux et vidit Deus lucem ...</div>      »
  </blockquote>
  <blockquote>¿
   <div>Et vocavit Deus aridam terram congregationesque aquarum appellavit maria
     ➥ et vidit Deus quod esset bonum et ait germinet terra herbam virentem et
     ➥ facientem semen et lignum pomiferum faciens fructum iuxta ...</div>´
  </blockquote>
  <blockquote>²
   <h2>Le huitième jour</h2>¶
   <p>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
     ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super CSS aquas
     ➥ dixitque Deus fiat lux et facta est XHTML ... </p>    º
  </blockquote>
 </body>
</html>
```

Les différences avec les exemples précédents ne se situent qu'au niveau de la présentation et celle-ci aurait pu être obtenue en appliquant des styles aux éléments <p> ou employés précédemment. Cela confirme bien qu'il faut s'attacher dans un premier temps à la structure et non à l'aspect.

Figure 3-5
Les divisions <blockquote> dans une page



Le texte préformaté : l'élément <pre>

Usuellement, cet élément sert à afficher le texte qu'il contient en préservant sa présentation et les différents espaces et indentations qui se trouvent dans le code XHTML, alors que par défaut les autres éléments ne conservent pas cette présentation. Le texte qu'il contient est affiché par défaut dans une police à espacement fixe, comme dans un éditeur de code. Pour ces raisons, l'élément <pre> est traditionnellement utilisé en préservant tous ses espaces. Il possède l'ensemble des attributs communs auxquels s'ajoute l'attribut xml:space qui prend la valeur unique preserve et dont le but est de définir explicitement la conservation de tous les espaces contenus dans l'élément. En tant que bloc, l'élément <pre> peut contenir des éléments en ligne. Les tableaux 3-9 et 3-10 donnent respectivement la liste des éléments enfants et parents de ce bloc. On notera par exemple qu'il ne peut pas contenir d'images.

Tableau 3-9. Les éléments enfants de l'élément <pre>

Texte, a, abbr, acronym, b, bdo, br, button, cite, code, del, dfn, em, i, input, ins, kbd, label, map, q, samp, script, select, span, strong, sub, sup, textarea, tt, var

Tableau 3-10. Les éléments parents de l'élément <pre>

blockquote, body, button, dd, del, div, fieldset, form, ins, li, map, noscript, object, td, th

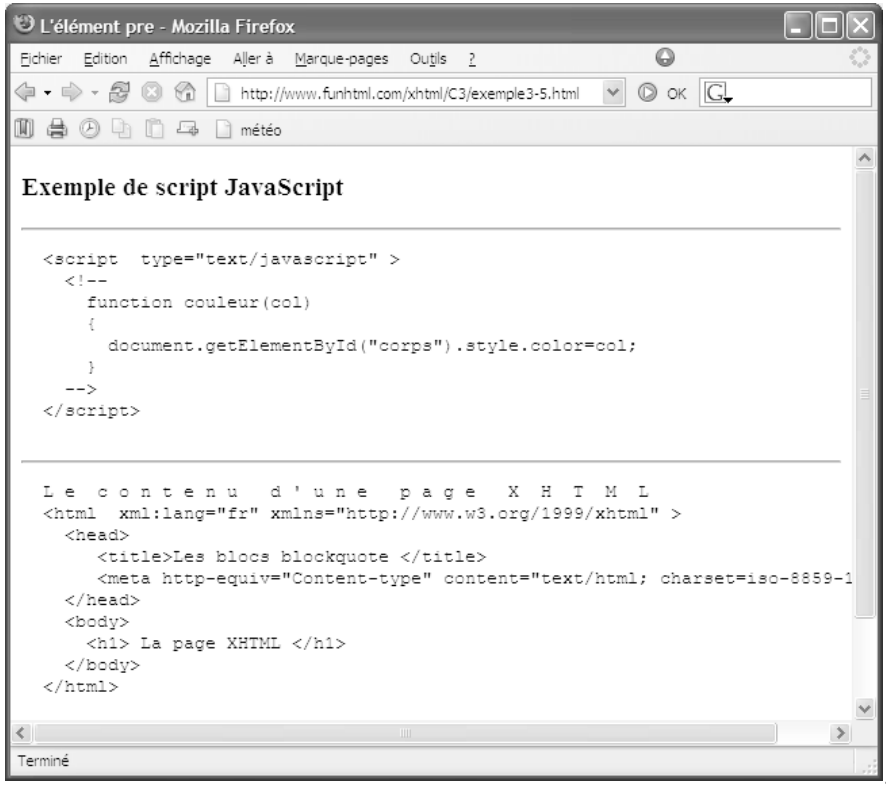
L'exemple 3-5 illustre l'utilisation de cet élément pour l'affichage du code d'un script JavaScript et du code d'une page XHTML en leur conservant leurs indentations.

Exemple 3-5. Utilisations de l'élément <pre>

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <title> L'élément pre </title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
 </head>
 <body>
  <h3>Exemple de script JavaScript</h3>
  <hr />
  <pre xml:space="preserve">
  &lt;script  type="text/javascript" &gt;
    &lt;!--
      function couleur(col)
      {
        document.getElementById("corps").style.color=col;
      }
    --&gt;
  &lt;/script&gt;
  </pre>
  <hr />
  <pre>»
  L e  c o n t e n u  d ' u n e  p a g e  X H T M L
  &lt;html  xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml" &gt;
    &lt;head&gt;
      &lt;title&gt;Les blocs blockquote &lt;/title&gt;
      &lt;meta http-equiv="Content-type" content="text/html; charset=iso-8859-1"
        ➥ /&gt;
    &lt;/head&gt;
    &lt;body&gt;
      &lt;h1&gt; La page XHTML &lt;/h1&gt;
    &lt;/body&gt;
  &lt;/html&gt;
  </pre>
 </body>
</html>
```

La figure 3-6 montre le résultat obtenu.

Figure 3-6
L'aspect par défaut des éléments <pre>



Les autres éléments de niveau bloc

Le langage XHTML définit plusieurs éléments mineurs de niveau bloc dont l'utilisation se révèle accessoire puisqu'ils peuvent être remplacés par exemple par un élément `<p>` ou `<div>` auquel est associé un style particulier. Il s'agit des éléments suivants :

- L'élément `<address>` dont le contenu est identique à l'élément `<p>` (éléments en ligne et texte brut). Son principal rôle est d'afficher l'adresse de contact avec le responsable du site. Son contenu est affiché par défaut en italique et en petits caractères. Il est traditionnellement inséré en bas de page (voir la figure 3-7). Il possède l'ensemble des attributs communs.

- Les éléments `<del>` et `<ins>`, généralement associés l'un à l'autre, sont destinés à contenir respectivement des informations actuellement supprimées, pour cause d'obsolescence par exemple, pour `<del>` et la version actualisée de ces informations pour `<ins>`. Ces deux éléments possèdent l'ensemble des attributs communs ainsi que l'attribut `cite` qui contient l'URL d'un document contenant les informations détaillées sur le contenu supprimé. L'attribut `datetime` contient la date de la suppression ou de la validité des contenus. Cette date doit être au format `AAAA-MM-JJ`

T hh-mm-ss Z. Les navigateurs actuels n'exploitent pas directement ces attributs. Pour y accéder, il faut avoir recours soit à un script, soit à un style CSS qui permet d'en afficher le contenu (voir l'utilisation de la propriété content).

- L'élément <form> permet de créer un formulaire. Il contient des éléments de niveau bloc qui renferment à leur tour les différents composants du formulaire. Compte tenu de l'importance de cet élément dans la création des pages web dynamiques, nous lui consacrons un chapitre particulier (voir le chapitre 7, *Créer des formulaires*).

- L'élément <fieldset> fait également partie du bloc. Son rôle habituel est également d'être inclut dans l'élément <form> pour créer des groupes de composants d'un formulaire (voir le chapitre 7). Par défaut, ces groupes sont délimités par un liseré fin. S'il contient l'élément <legend> le contenu de ce dernier est affiché dans le liseré. L'élément <fieldset> pouvant contenir tous les autres éléments de bloc, les éléments en ligne, les titres et les listes, il est donc envisageable de l'utiliser pour créer des divisions particulières dans la page, même en dehors du contexte des formulaires, comme , en profitant de son style par défaut. Il possède la totalité des attributs communs.

L'exemple 3-6 permet de tester les éléments <del> et <ins> pour lesquels nous gérons l'événement onclick qui permet d'afficher une boîte d'alerte affichant quant à elle la valeur de l'attribut cite (repères ³ et ·). Avec l'élement <fieldset> (repère »), on peut afficher une zone de note encadrée contenant un titre et un paragraphe (repères ¿ , ´ et ²). Enfin, l'élément <address> contenant un lien permet d'entrer en contact avec le responsable du site (repère ¶).

Exemple 3-6. Les éléments de bloc annexes

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <title> Divers éléments de bloc </title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
 </head>
 <body>
  <h1>Les éléments annexes de blocs</h1>
  <p>La version ancienne
   <del cite="http://www.w3.org" datetime="1998-12-31"
    ➥ onclick="alert(this.cite)">HTML</del> ³
    ➥ a été remplacée par les spécifications
  <ins cite="http://www.w3.org/XHTML" onclick="alert(this.cite)" >
    ➥ XHTML 1.1</ins>
```

```
  </p>
  <fieldset>  »
   <legend>Note : </legend>¿
   <h2>La création de XHTML</h2́>
   <p>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
     ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
     ➥ dixitque Deus fiat lux et facta est lux et vidit Deus lucem quod esset bona
     ➥ et divisit lucem ac tenebras appellavitque lucem diem et tenebras noctem
     ➥ </p>²
  </fieldset><br />
  <address>Contact avec l'auteur : <a id="auteur"
     ➥ href="mailto:xhtml@funhtml.com">xhtml@funhtml.com¶/a>
  </address>
 </body>
</html>
```

La figure 3-7 offre un rendu visuel de ces éléments.

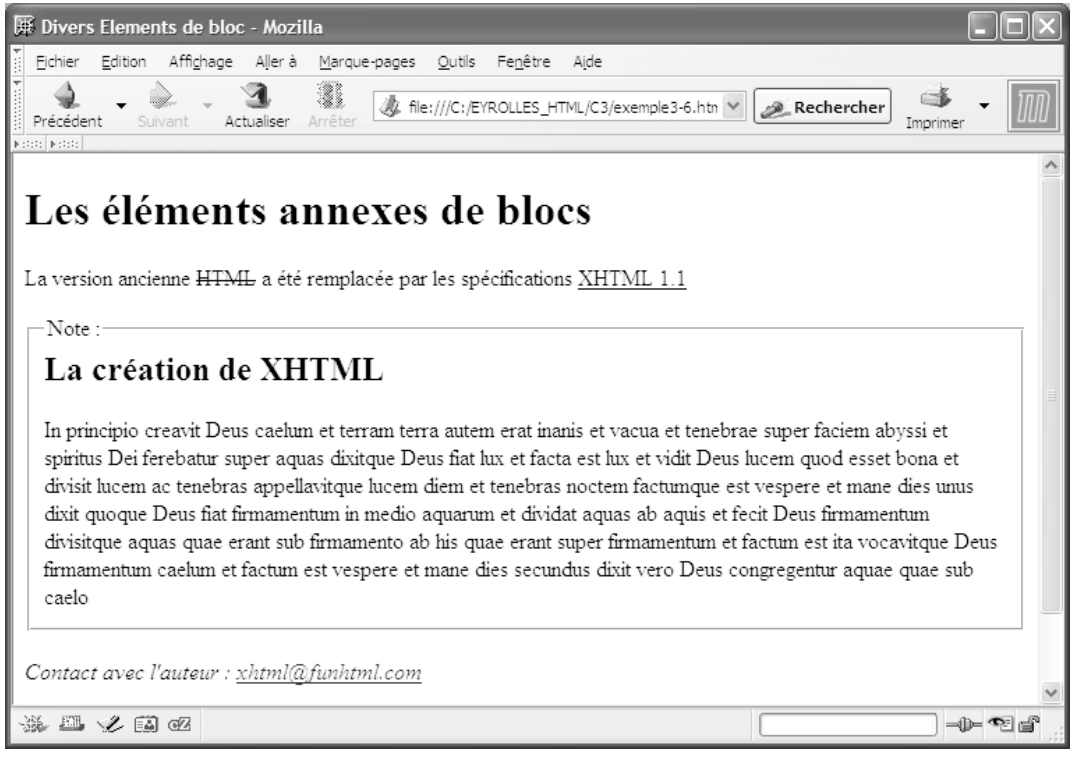


Figure 3-7
Les éléments de bloc annexes

Les éléments des divisions sémantiques en ligne

Le langage HTML offrait un nombre relativement important d'éléments en ligne qui permettaient de délimiter des contenus en ligne. Ces derniers étaient souvent limités à quelques mots afin de les particulariser et de les mettre en évidence, le plus fréquemment au moyen d'un style prédéfini. XHTML a conservé la plupart de ces éléments mais nombre d'entre eux perdent de leur intérêt pratique car, en associant des styles CSS différents à un seul type d'élément, `<span>` par exemple, nous constaterons par la suite que nous pouvons obtenir les mêmes résultats visuels. On peut envisager que certains d'entre eux soient appelés à disparaître. Ils gardent cependant un rôle dans la structuration du contenu d'une page. En tant qu'éléments en ligne, ils doivent être inclus dans un élément de type bloc, et jamais directement dans le corps de la page. Les paragraphes suivants présentent l'ensemble de ces éléments en ligne et leur rôle respectif.

Tous ces éléments ont en commun les mêmes éléments enfants dont la liste figure dans le tableau 3-11.

Tableau 3-11. Les éléments enfants des éléments sémantiques en ligne

Texte , a, abbr, acronym, b, big, bdo, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var

Les abréviations

L'élément `<abbr>` contient une abréviation, par exemple « kg » pour kilogramme ou « ciné » pour cinématographe. Cet élément possède la totalité des attributs communs, les plus adaptés étant `id` , `class` et `title` . Il peut contenir tous les éléments en ligne. Nous utilisons généralement l'attribut commun `title` pour afficher la signification exacte de l'abréviation quand l'utilisateur place le curseur sur son contenu signalé dans les navigateurs par un soulignement en pointillés. Les attributs gestionnaires d'événements `onclick` et `onmouseover` peuvent permettre d'afficher une boîte d'alerte contenant la définition détaillée.

Le code suivant :

```
<abbr title="kilogramme" onmouseover="alert('Unité de masse: kilogramme')">kg</abbr>
```

définit l'abréviation d'une unité physique. Sa signification est donnée dans l'attribut `title` , et elle s'affiche de manière détaillée dans une boîte d'alerte quand le curseur survole le texte en utilisant l'attribut gestionnaire d'événements `onmousever` (voir l'exemple 3-7 (repères · et ») et la figure 3-8). Notez que cet élément est ignoré par Internet Explorer 6 qui affiche son contenu comme du texte ordinaire et ne reconnaît pas l'attribut `title` pour afficher la signification, et encore moins les gestionnaires d'événements.

Les acronymes

L'élément <acronym> est le conteneur d'un acronyme, « XHTML », « PHP » ou « UNICEF » par exemple. Comme <abbr> il possède la totalité des attributs communs et est encore affiché par défaut avec un soulignement en pointillés. C'est encore l'attribut title qui est utilisé pour afficher une bulle contenant la définition in extenso de l'acronyme. Il peut aussi contenir l'ensemble des éléments en ligne.

Le code suivant :

```
<p>
  Utilisez vite <acronym title="eXtensible HypertText Markup Language">XHTML 1.1
  ➥ </acronym> promis à un grand avenir!
</p>
```

donne la définition de l'acronyme « XHTML » dans l'attribut title . Voir l'exemple 3-7 (repère ¿) et la figure 3-8.

Le sens de lecture du texte

L'élément <bdo> permet de modifier localement la définition du sens de lecture de son contenu qui a pu être donné avec l'attribut dir de son élément parent. En plus des attributs de base (id , class , title), il possède en effet l'attribut obligatoire dir qui prend les valeurs habituelles, ltr et rtl , pour indiquer que le sens de lecture se fait respectivement de gauche à droite, ou l'inverse. L'attribut xml:lang permet de préciser le code de la langue associé localement au contenu de l'élément <bdo> Cet élément généralement employé pour des textes courts peut cependant contenir tous les éléments en ligne.

Le code suivant :

```
<p>Lire de gauche à droite
 <bdo dir="ltr" xml:lang="fr">XHTML</bdo> ou de droite à gauche
 ➥ <bdo dir="rtl" xml:lang="ar">XHTML</bdo>
</p>
```

permet d'afficher le mot « XHTML » successivement de gauche à droite et de droite à gauche, ce qui fournit les affichages « XHTML » et « LMTHX » (voir l'exemple 3-7 (repère ´) et la figure 3-8).

Les citations

Nous avons déjà vu que l'élément <blockquote> permet de contenir des citations et de les mettre en évidence. L'élément <cite> a une destination similaire mais est utilisé en ligne pour des citations courtes. Il peut contenir directement du texte brut, tous les éléments en ligne ainsi que <ins>, <del> et <script>, et possède tous les attributs communs.

Le code suivant :

```
<p>Comme le disait Boris Vian
 <cite title="Vian 1920-1959"> La vérité n'est pas du coté du plus grand nombre
  ➥ car on ne veut pas qu'elle y soit. Le jour où il sera à même, par sa culture
  ➥ et ses connaissances, de choisir lui même sa vérité, il y a peu de chance
  ➥ pour qu'il se trompe.</cite>
</p>
```

affiche une citation dans une police en italique par défaut (voir l'exemple 3-7 (repère ²) et la figure 3-8).

L'inclusion du code source

Nous avons déjà utilisé l'élément <pre> pour insérer du texte préformaté dans un bloc. L'équivalent en ligne est l'élément <code> qui nous permet d'inclure dans une phrase le texte d'une instruction et de le mettre en évidence par rapport au contexte, car il est affiché par défaut dans une police à espacement fixe du type listing.

Le code suivant :

```
<p>Pour créer une boîte d'alerte en JavaScript, nous écrivons par exemple:
  <code>alert('Bonjour')</code>. Dans ce cas l'exécution du script s'arrête.
    ➥ <!-- 7 -->
</p>
```

affiche un paragraphe contenant du code JavaScript (voir l'exemple 3-7 (repère ¶) et la figure 3-8).

L'élément <code> peut contenir par ailleurs tous les éléments en ligne. Il possède aussi tous les attributs communs. L'attribut title peut servir par exemple à afficher le nom du langage de programmation du code.

Les définitions en ligne <dfn>

L'élément <dfn> sert de conteneur à une définition de terme. Son emploi est rare en tant que tel. Il contient du texte et les autres éléments en ligne. Par défaut, son contenu est en italique. Il possède tous les attributs communs.

Le code suivant :

```
<p>Ce bâtiment a la forme d'un pentagone
 <dfn>(polygone à cinq cotés)</dfn> et abrite des gens peu recommandables.
</p>
```

affiche le contenu de l'élément en italique (voir l'exemple 3-7 (repère ³) et la figure 3-8).

Les saisies du clavier

Afin d'indiquer au visiteur les saisies à faire au clavier pour effectuer une action particulière, il faut inclure ces saisies dans l'élément <kbd> dont l'emploi se révèle d'ailleurs

aussi rare que le précédent. Par défaut, son contenu textuel est affiché comme celui de l'élément <code> dans une police à espacement fixe.

Nous écrirons par exemple :

```
<p>Si vous en avez assez, tapez : <kbd>Ctrl+Alt+Suppr</kbd>, et au revoir!</p>
```

(voir l'exemple 3-7 (repère ¾) et la figure 3-8).

Les citations courtes

En supplément de l'élément <cite>, nous pouvons utiliser l'élément <q> pour des citations courtes qui sont par défaut automatiquement incluses entre guillemets (en anglais, *quotes* d'où le nom de l'élément) dans la plupart des navigateurs. Toutefois, il n'est pas dans l'esprit XHTML de compter sur une présentation automatique et il vaudra mieux créer un style personnalisé pour obtenir cet effet. On se reportera à ce sujet aux pseudo-éléments :before et :after au chapitre 9, et à la propriété content. L'attribut cite permet de donner l'URL d'une page susceptible de renseigner en détail sur la citation et son auteur.

Nous pouvons écrire, par exemple :

```
<p>Comme Hamlet posons nous la question : <q
➥ cite="http://www.funhtml.com/hamlet.html" title="Hamlet : William Shakespeare"
➥ onclick="alert('Voir '+this.cite)">Etre ou ne pas  &ecirc;tre </q></p>
```

L'attribut cite est ici défini, et le gestionnaire d'événements onclick permet d'afficher une boîte d'alerte contenant l'adresse figurant dans cet attribut cite (voir l'exemple 3-7 (repère µ) et la figure 3-8).

Les exemples en ligne : <samp>

Pour mettre en évidence, en particulier dans un paragraphe de texte, un texte d'exemple pour illustrer le propos qui précède, il est possible d'utiliser l'élément <samp> contenant le texte de l'exemple qui sera par défaut représenté dans une police à espacement fixe du type Courier dans les navigateurs. Notons encore que l'utilisation de l'élément doté d'un style particulier remplirait la même fonction. L'élément <samp> peut contenir l'ensemble des éléments en ligne et possède tous les attributs communs.

Nous pouvons écrire, par exemple :

```
<p>Le type de l'équation du premier degré à deux inconnues est : <samp> ax+by = c
➥ </samp>. Elle n'a pas de solution unique</p>
```

(voir l'exemple 3-7 (repère ¸) et la figure 3-8).

Les divisions en ligne

L'équivalent en ligne de l'élément de bloc peut être réalisé à l'aide de l'élément . C'est le plus employé des éléments créant des divisions sémantiques en ligne. Il n'a pas de rôle prédéfini et peut remplacer nombre des éléments précédents de cette

section à condition de lui attribuer un style particulier adapté à chaque besoin de présentation. Il possède l'ensemble des attributs communs et c'est son attribut `class` qui sera systématiquement utilisé pour lui attribuer un style.

Le code suivant :

```
<p>Le langage <span class="gras">XHTML</span> a pour complément indispensable
➥ les styles <span class="gras">CSS 2</span></p>
```

crée deux divisions en ligne qui utilise un style CSS particulier, défini par ailleurs dans l'élément `<style>` de l'en-tête (repère ❸). Leurs contenus sont donc affichés dans une police plus grande que celle du paragraphe parent et en gras (voir l'exemple 3-7 (repère ❸) et la figure 3-8).

C'est donc un élément passe-partout à usage très divers que nous retrouverons à chaque fois qu'il s'agira d'appliquer un style de façon ponctuelle à un contenu réduit en ligne.

Le conteneur des variables : <var>

Le dernier conteneur en ligne, `<var>`, est destiné à contenir des textes représentant une variable. Son contenu est par défaut affiché dans une police cursive. Il peut par ailleurs renfermer tous les éléments en ligne et possède tous les attributs communs. Comme plusieurs des éléments précédents, il n'a pas un intérêt fondamental car il peut aisément être remplacé par l'élément `<span>` doté d'un style au choix. On peut le considérer comme une survivance des versions anciennes de HTML employées sans CSS.

Le code suivant :

```
<p> Dans l'équation <code> ax+bg = c </code> les variables sont <var> x </var>
➥ et <var> y </var> </p>
```

met en évidence les variables x et y (voir l'exemple 3-7 (repère ❹) et la figure 3-8).

L'exemple 3-7 rend compte d'une utilisation de tous ces éléments en ligne.

Exemple 3-7. Les éléments sémantiques en ligne

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les éléments sémantiques en ligne</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" >
   .gras {font-size:larger; font-weight:bold;}        ❸
  </style>
 </head>
 <body>
```

```
<!-- L'élément abbr -->
<h1>Abréviations</h1>
<p>La masse se mesure en <abbr title="kilogramme" onmouseover="alert('Unité
➥ de masse: kilogramme')">kg</abbr>
 La force se mesure en <abbr title="Newton" onmouseover="alert('Unité de force:
➥ Newton')">N</abbr><br />
</p>
<!-- L'élément acronym -->
<h1>Acronymes</h1>
<p>Utilisez vite <acronym title="eXtensible HypertText Markup Language">
➥ XHTML 1.1</acronym> promis à un grand avenir!
</p>
<!-- L'élément  bdo -->
<h1>Sens de lecture</h1>
<p>Lire de gauche à droite
  <bdo dir="ltr" xml:lang="fr">XHTML</bdo> ou de droite à gauche
    ➥ <bdo dir="rtl" xml:lang="ar">XHTML</bdo>
</p>
<!-- L'élément  cite -->
<h1>Citations</h1>
<p>Comme le disait Boris Vian
  <cite title="Vian 1920-1959"> La vérité n'est pas du coté du plus grand nombre
    ➥ car on ne veut pas qu'elle y soit. Le jour où il sera à même, par sa culture
    ➥ et ses connaissances, de choisir lui même sa vérité, il y a peu de chance
    ➥ pour qu'il se trompe.</cite>
</p>
<!-- L'élément code -->
<h1>Code source en ligne</h1>
<p>Pour créer une boîte d'alerte en JavaScript, nous écrivons par exemple:
  <code>alert('Bonjour')</code>. Dans ce cas l'exécution du script s'arrête.
</p>
<!-- L'élément  dfn -->
<h1>Définition en ligne</h1>
<p>Ce bâtiment a la forme d'un pentagone
  <dfn>(polygone à cinq cotés)</dfn> et abrite des gens peu recommandables.
</p>
<!-- L'élément  kbd -->
<h1>Saisies clavier</h1>
<p>Si vous en avez assez, tapez : <kbd>Ctrl+Alt+Suppr</kbd>, et au revoir!</p>
<!-- L'élément q -->
<h1>Citations courtes</h1>
 <p>Comme Hamlet posons nous la question : <q cite="http://www.funhtml.com/
   ➥ hamlet.html" title="Hamlet : William Shakespeare" onclick="alert
   ➥ ('Voir '+this.cite)">Être ou ne pas  &ecirc;tre </q></p>
<!-- L'élément  samp -->
<h1>Exemples en ligne</h1>
<p>Le type de l'équation du premier degré à deux inconnues est :
  ➥ <samp> ax+by = c</samp>. Elle n'a pas de solution unique</p>
```

```
<!-- L'élément span -->
<h1>Le conteneur span</h1>
<p>Le langage <span class="gras">XHTML</span> a pour complément indispensable
  ➥ les styles <span class="gras">CSS 2</span></p>
<!-- L'élément  var -->
<h1>Variables</h1>
<p> Dans l'équation <code> ax+by = c </code> les variables sont  <var> x </var>
  ➥ et <var> y </var> </p>
</body>
</html>
```

La figure 3-8 montre tous les affichages obtenus en utilisant ces éléments sémantiques en ligne.

Figure 3-8

Le rendu des éléments sémantiques en ligne

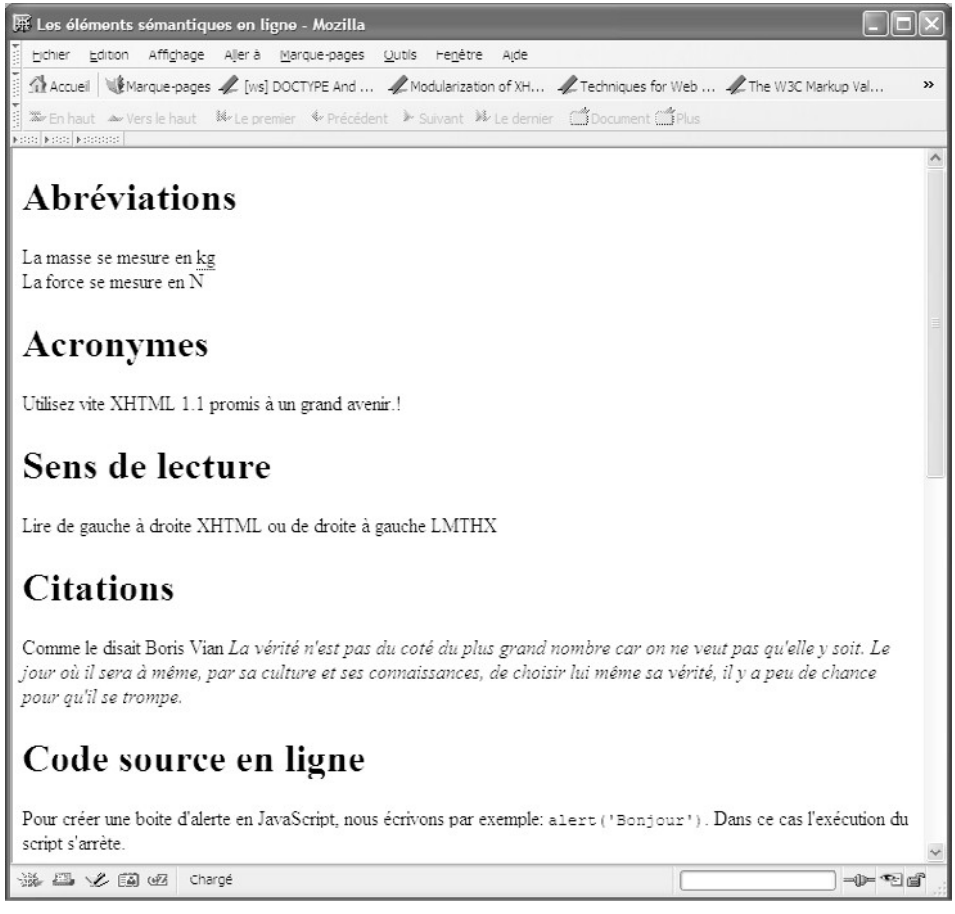
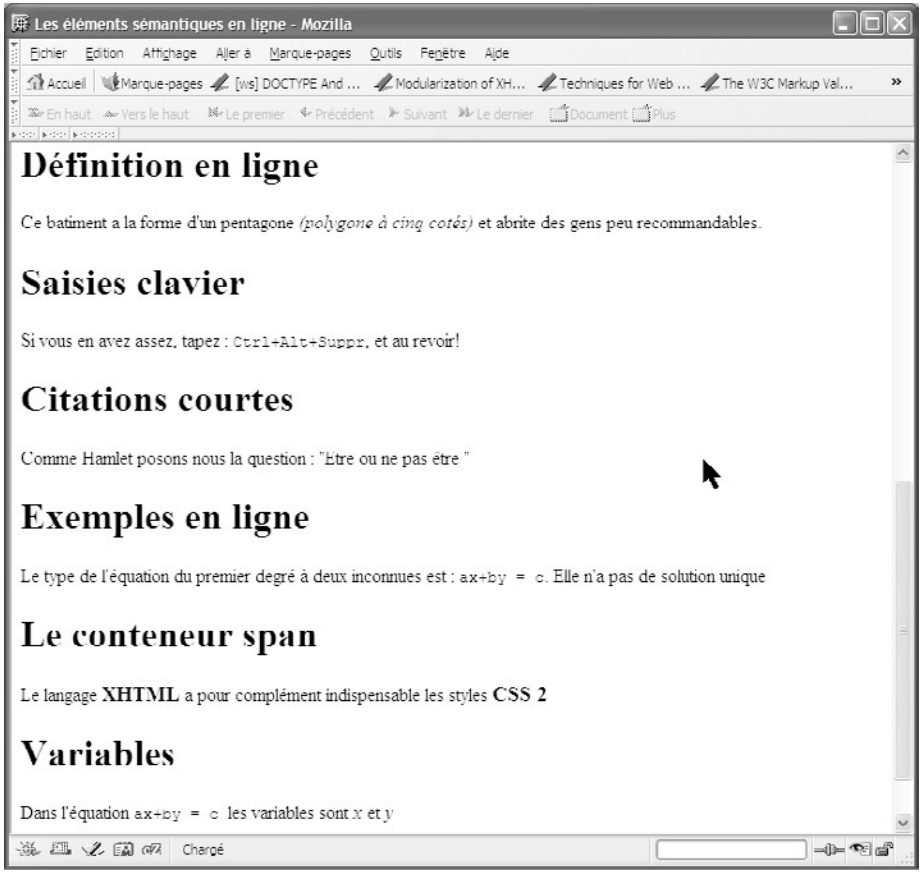


Figure 3-8b
Le rendu des éléments sémantiques en ligne (suite)



Les styles physiques

Parmi les éléments en ligne utilisables pour contenir du texte, certains permettent de créer des styles physiques pour leur contenu. Une partie d'entre eux correspondent aux modifications courantes que chacun peut effectuer dans son traitement de texte, comme mettre un texte en gras, en italique, ou certains caractères en indice ou en exposant. Ce type de marquage est indépendant de la taille et de la police de caractères. D'autres éléments agissent de manière relative sur leur contenu en permettant d'afficher dans une police plus grande ou plus petite que la police utilisée dans le texte qui précède sans préjuger de cette taille. À chacun de ces éléments correspond un style par défaut qui peut donner satisfaction, mais dans le cas contraire ce style pourra être personnalisé à loisir avec CSS.

Mettre un texte en gras

Pour mettre en gras une partie de texte comprise dans un bloc, il faut l'inclure dans l'élément (donc entre et). Il peut aussi contenir tous les éléments en ligne et possède tous les éléments en ligne de tous les attributs communs. Nous pouvons également utiliser l'élément pour obtenir le même effet. Il n'est pas possible d'imbriquer ces éléments les uns dans les autres pour forcer le caractère gras d'un texte.

Nous écrirons par exemple le code suivant (voir l'exemple 3-8 (repère) et la figure 3-9) :

```
<p> Le contenu suivant est <b> important </b>. La suite l'est moins. Mais ceci
➥ est <strong>également remarquable.</strong> </p>
```

Il sera toujours possible de redéfinir l'affichage obtenu pour ces éléments et de différencier et au moyen de styles CSS particuliers.

Mettre du texte en italique

Comme pour les éléments précédents, il existe deux éléments qui permettent d'afficher leur conteneur en italique. Il s'agit des éléments <i> et . Nous pourrons également redéfinir le résultat obtenu en créant des styles personnalisés.

Nous écrirons, par exemple, le code suivant :

```
<p>Éléments i et em : Celui est en <i>caractères italiques </i> et le suivant
➥ en<i><b> caractères italiques gras</b></i>
Celui est en <em>caractères italiques </em> et le suivant en<em><b>
➥ caractères italiques gras</b></em></p>
```

Nous y remarquons qu'il est possible d'imbriquer des éléments dans les éléments <i> et pour que le texte apparaisse à la fois en italique et en caractères gras (voir l'exemple 3-8 (repère ») et la figure 3-9) :

Modifier la taille du texte

À l'intérieur d'une division de la page, nous pouvons modifier la taille relative du texte en utilisant les éléments <big> et <small>. Le premier permet d'afficher son contenu dans une taille de police plus grande que celle de la police qui le précède. Le second permet d'afficher son contenu avec une taille de police plus petite. En imbriquant les éléments <big> ou <small> les uns dans les autres, on peut obtenir des tailles de police de plus en plus petites ou de plus en plus grandes. Le nombre d'imbrications possibles dépend de la taille de la police du conteneur des éléments <big> ou <small>.

Le code suivant :

```
<p> Ce texte là est <big>grand, <big>encore plus grand, <big>et plus grand encore,
➥ <big>toujours plus grand </big></big></big></big></p>
<p>Ce texte ci est <small>petit, <small>encore plus petit, <small>et plus petit
➥ encore, <small>toujours plus petit </small></small></small></small></p>
```

permettra d'obtenir des mots de plus en plus grands en imbriquant les éléments <big> puis de plus en plus petits en imbriquant les éléments <small> sur quatre niveaux (voir l'exemple 3-8 (repères ¿ et ´) et la figure 3-9). Dans l'exemple 3-8, la taille de la police de l'élément <body> a été définie à 16 pixels à l'aide d'un style (repère ³) pour que l'effet des imbrications soit visible sur un plus grand nombre de niveaux qu'avec la taille par défaut.

Créer des exposants et des indices

Les traitements de texte offrent aussi la possibilité de mettre des caractères en exposant ou en indice. Cette opération est également possible en XHTML. Pour mettre un texte en exposant, il faut l'inclure dans l'élément <sup> (entre ^{et}), et pour écrire un texte en indice il faut l'inclure dans l'élément <sub> (entre _{et}).

Nous pouvons avoir par exemple le code suivant dans l'exemple 3-8 (repère ²) dont le résultat est visible sur figure 3-9 :

```
<p>Le n<sup>ième</sup> terme de la suite numérique est noté u<sub>n</sub>.
➥ La fonction cube est notée : x<sup>3</sup></p>
```

Afficher du texte dans une police à espacement fixe

Nous avons déjà utilisé l'élément <code> pour afficher du texte dans un style listing. L'élément <tt> (comme télétype) permet d'obtenir par défaut le même résultat pour des textes courts en ligne. Il peut contenir éventuellement tous les attributs communs.

Nous pouvons, par exemple, écrire le code suivant :

```
<p> La fonction JavaScript <tt> alert() </tt> permet d'afficher une boîte d'alerte.
➥ On peut écrire, par exemple: <tt> alert(" Vérifier votre code ") </tt> pour
➥ prévenir le visiteur.</tt> </p>
```

pour mettre en évidence dans un paragraphe du code JavaScript (voir l'exemple 3-8 (repère ¶) et la figure 3-9).

Créer un retour à la ligne

L'élément <br /> permet de créer un retour à la ligne. C'est un élément vide, d'où l'utilisation du caractère antislash (/) en guise de signe de fermeture. Il possède les attributs de base id, class, title, mais ceux-ci n'ont pas un intérêt particulier.

L'exemple 3-8 utilise successivement l'ensemble de ces éléments, créant des styles physiques.

Exemple 3-8. Les éléments de style physique

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html  xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
 <title>Les éléments de style physique</title>
 <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
 <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 <style type="text/css" >
 body{font-size:16px;}
 </style>
</head>
<body>
 <p>Éléments b et strong : Le contenu suivant est <b> important </b>. La suite
    l'est moins. Mais ceci est <strong>également remarquable.</strong>  </p>
 <p>Éléments i et em : Celui est en <i>caractères italiques </i> et le suivant
    en<i><b> caractères italiques gras</b></i>
Celui est en <em>caractères italiques </em> et le suivant en<em><b> caractères
    italiques gras</b></em></p>
 <p> Ce texte là est <big>grand, <big>encore plus grand, <big>et plus grand encore,
    <big>toujours plus grand </big></big></big></big></p>
 <p>Ce texte ci est <small>petit, <small>encore plus petit, <small>et plus petit
    encore, <small>toujours plus petit </small></small></small></small></p>
 <p>Le n<sup>ième</sup> terme de la suite numérique est noté u<sub>n</sub>.
    La fonction cube est notée : x<sup>3</sup></p>
 <p> La fonction Java Script <tt> alert() </tt> permet d'afficher une boîte
    d'alerte. On peut écrire, par exemple: <tt> alert(" Vérifier votre code ")
    </tt> pour prévenir le visiteur. </p>
</body>
</html>
```

La figure 3-9 présente l'ensemble des résultats obtenus en utilisant ces éléments.

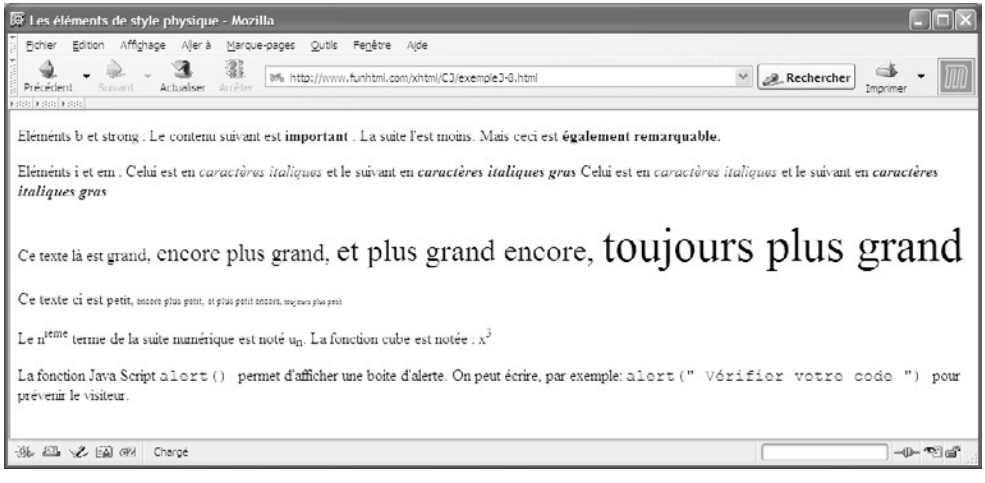


Figure 3-9

Les éléments de style physique

Les listes

La présentation sous forme de liste permet une structuration de l'information telle qu'elle peut apparaître dans une table des matières. On peut également mettre en évidence les points importants. Les utilisateurs de traitement de texte sont familiarisés avec cette façon de procéder. Elle implique qu'une série d'informations aient un rapport entre elles, par exemple sous forme d'énumération d'une liste de tâches à réaliser. Ces listes d'informations peuvent être numérotées ou marquées par une puce graphique. De la même façon, avec XHTML on peut créer des listes d'items numérotées, nommées listes ordonnées, ou de listes à puces, nommées listes non ordonnées. Un troisième type de listes permet également d'énumérer des termes et d'en donner les définitions. Les éléments permettant la création de listes ne sont pas des éléments de bloc à proprement parler, mais ils ont un comportement similaire et peuvent être inclus directement dans l'élément <body>

Les listes ordonnées

Pour créer une liste dans laquelle la notion d'ordre a une importance, nous pouvons utiliser une liste ordonnée dont chaque item sera numéroté par défaut à l'aide d'entiers incrémentés de 1 à N, suivis d'un point puis du contenu de chaque item. Cette présentation type résulte du style utilisé par défaut par les navigateurs mais nous verrons dans la deuxième partie de ce livre (voir le chapitre 15, *Le style des listes*) que la présentation des éléments de liste peut être beaucoup plus variée en utilisant des propriétés CSS 2 spécifiques. Une liste ordonnée doit commencer par l'élément <ol> (pour *ordered list*) qui doit obligatoirement contenir au moins un élément <li> qui lui-même renferme le contenu visible de chaque item. Il faut donc que <ol> contienne autant d'éléments <li> qu'il y a d'items dans la liste désirée. L'élément <ol> ne peut rien contenir d'autre, même pas de texte brut. La structure d'une liste ordonnée est donc la suivante :

```
<ol>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ol>
```

Les éléments <li> peuvent avoir un contenu très varié, qu'il s'agisse de texte, de titres, de tous les éléments de niveau bloc, des éléments en ligne et d'autres éléments de liste. Le tableau 3-12 donne la liste des éléments enfants de l'élément <li> .

**Tableau 3-12. Les éléments enfants de l'élément **

Texte, a, abbr, acronym, address, b, bdo, big, blockquote, br, button, cite, code, del, dfn, div, dl, em, fieldset, form, h1, h2, h3, h4, h5, h6, hr, i, img, input, ins, kbd, label, map, noscript, object, ol, p, pre, q, samp, script, select, small, span, strong, sub, sup, table, textarea, tt, ul, var

L'exemple 3-9 montre une utilisation des éléments <ol> et <li> pour créer des listes ordonnées. L'élément <ol> contient trois éléments <li> créant ainsi trois items (repères ³ , ¿ et ¶). Tous ces éléments <li> renferment chacun un titre <h3> (repères · , ´ et º) et un paragraphe (repères » , ² et ¾).

Exemple 3-9. Une liste ordonnée

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <title>Les listes XHTML</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
 </head>
 <body>
  <h1>Les listes</h1>
  <h2>Les listes ordonnées</h2>
  <ol>
   <li> ³
    <h3>Le premier jour</h3> ·
    <p> In principio creavit Deus caelum et terram terra autem erat inanis et vacua
      ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
      ➥ dixitque Deus fiat lux et facta est lux et vidit Deus lucem quod esset bona
      ➥ et divisit .. </p>  »
   </li>
   <li> ¿
    <h3>Le deuxième jour</h3>´
    <p> In principio creavit Deus caelum et terram terra autem erat inanis et vacua
      ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
      ➥ dixitque Deus fiat lux et facta est lux et vidit Deus lucem quod esset bona
      ➥ et divisit lucem ac tenebras..
    </p> ²
   </li>
   <li> ¶
    <h3>Le troisième jour</h3> º
    <p> In principio creavit Deus caelum et terram terra autem erat inanis et vacua
      ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
      ➥ dixitque Deus fiat lux et facta est lux et vidit Deus lucem quod esset bona
      ➥ et divisit lucem ac tenebras..
    </p> ¾
   </li>
  </ol>
 </body>
</html>
```

La figure 3-10 présente le résultat obtenu.

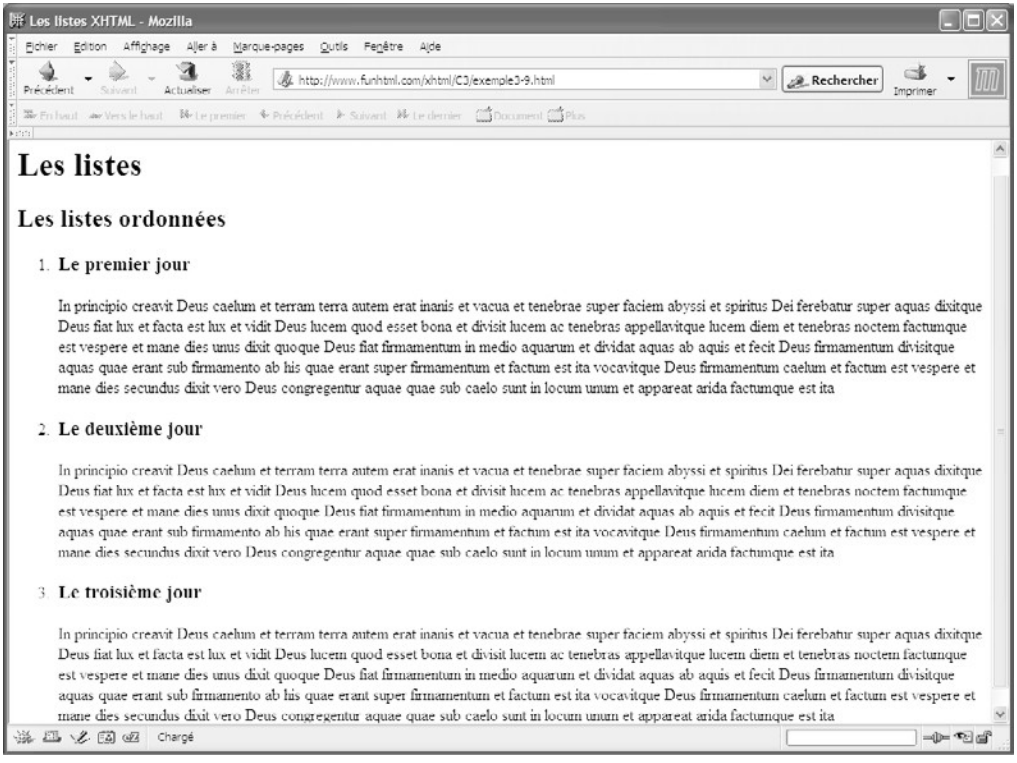


Figure 3-10
Une liste ordonnée et son style par défaut

Les listes non ordonnées

Les listes non ordonnées (*unordered list*) fournissent un outil de structuration similaire au précédent mais sans la notion de numérotation. Elles sont également appelées listes à puces car chaque item est précédé d'une puce graphique qui, par défaut, est un disque plein de la même couleur que le texte qui la suit. Nous verrons au chapitre 15 que ces puces graphiques peuvent être personnalisées à volonté. Une liste à puces est introduite par l'élément `<ul>` et fermée par `</ul>`. Comme l'élément `<ol>`, son seul contenu direct est un ou plusieurs éléments `<li>`. Le contenu de chaque élément `<li>` est le même que pour les listes ordonnées. La structure d'une liste à puces est donc la suivante :

```
<ul>
   <li>Item 1</li>
   <li>Item 2</li>
   <li>Item 3</li>
</ul>
```

L'exemple 3-10 montre une utilisation complète d'une liste à puces. L'élément <ul> contient trois éléments <li> créant ainsi trois items (repères ³ , ¿ et ¶). Tous ces éléments <li> contiennent chacun un titre <h3> (repères · , ´ et º) et un paragraphe (repères » , ² et ¾). Nous pouvons noter que le code est pratiquement identique à celui de l'exemple 3-9, la seule différence résidant dans le remplacement de l'élément <ol> par <ul>.

Exemple 3-10. Une liste à puces

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <title>Les listes XHTML</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
 </head>
 <body>
  <h1>Les listes</h1>
  <h2>Les listes ordonnées</h2>
  <ol>
   <li> ³
    <h3>Le premier jour</h3> ·
    <p> In principio creavit Deus caelum et terram terra autem erat inanis et vacua
      ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
      ➥ dixitque Deus fiat lux et facta est lux et vidit Deus lucem quod esset bona
      ➥ et divisit ..  </p>  »
   </li>
   <li> ¿
    <h3>Le deuxième jour</h3>´
    <p> In principio creavit Deus caelum et terram terra autem erat inanis et vacua
      ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
      ➥ dixitque Deus fiat lux et facta est lux et vidit Deus lucem quod esset bona
      ➥ et divisit lucem ac tenebras..
    </p> ²
   </li>
   <li> ¶
    <h3>Le troisième jour</h3> º
    <p> In principio creavit Deus caelum et terram terra autem erat inanis et vacua
      ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
      ➥ dixitque Deus fiat lux et facta est lux et vidit Deus lucem quod esset bona
      ➥ et divisit lucem ac tenebras..
    </p> ¾
   </li>
  </ol>
 </body>
</html>
```

La figure 3-11 montre le résultat obtenu.

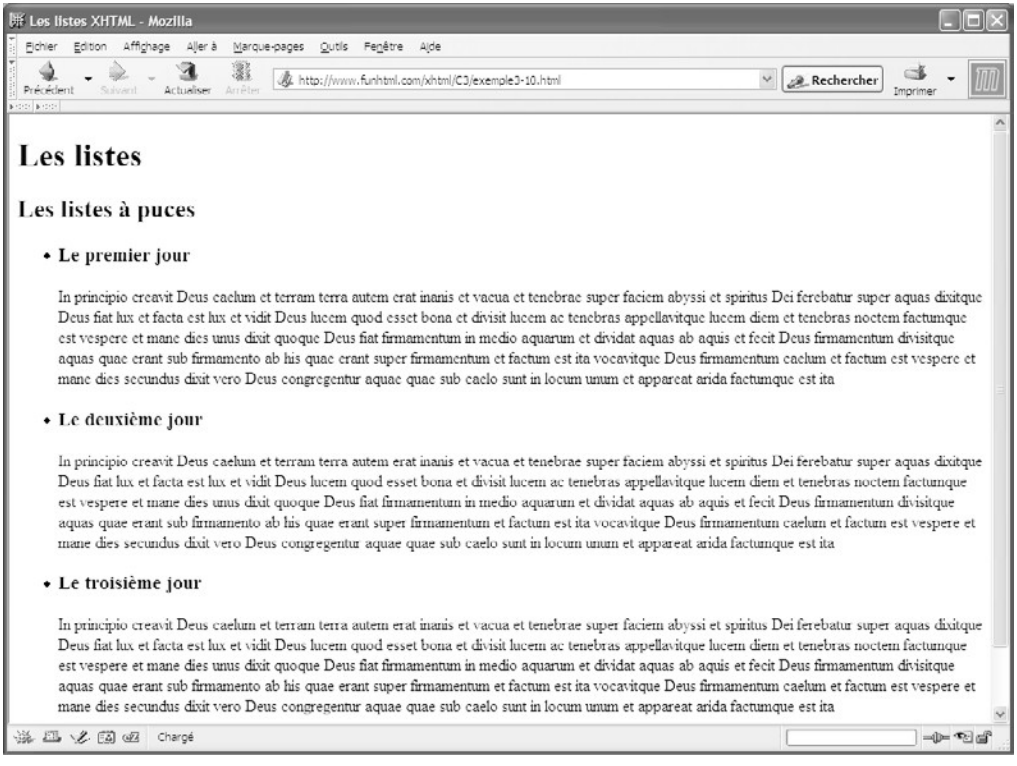


Figure 3-11

Une liste à puces et son style par défaut

Les listes imbriquées

L'élément <li> , seul contenu autorisé des éléments <ol> et <ul>, peut lui-même avoir un contenu très varié, et contenir d'autres éléments <ol> ou <ul>. Ainsi, nous pouvons exploiter cette propriété pour créer des listes imbriquées les unes dans les autres. Nous obtenons ainsi plusieurs niveaux de numérotation ou de puces, à l'instar d'une table des matières qui reflète les titres des chapitres puis les titres des différentes sections les composant. Chaque élément <li> de premier niveau contient une élément ou <ul> qui lui-même renferme des éléments <li> de deuxième niveau.

La structure des listes ordonnées imbriquées est donc la suivante :

```
<ol>
 <li>Item 1
  <ol>
    <li>Item 1 A</li>
```

```
    <li>Item 1 B</li>
   </ol>
  </li>
  <li>Item 2
   <ol>
    <li>Item 2 A</li>
    <li>Item 2 B</li>
   </ol>
  </li>
</ol>
```

La numérotation des listes ordonnées de deuxième niveau figure par défaut en chiffres arabes et reprend à 1 pour chaque liste de deuxième niveau. De plus, les contenus des items de deuxième niveau sont indentés par défaut par rapport à ceux du niveau supérieur. Pour créer des listes imbriquées à puces, il suffit de remplacer les éléments `<ol>` par `<ul>`. Notons que, dans ce cas, les puces de deuxième niveau sont par défaut des cercles au lieu de disques, ce qui permet de distinguer chaque niveau.

Il est aussi envisageable de créer des listes à puces imbriquées mixtes, c'est-à-dire dans lesquelles un élément `<ul>` est utilisé comme descendant de l'élément `<ol>`, et réciproquement. L'exemple 3-11 montre une possibilité de mise en œuvre de listes imbriquées. Le premier niveau est constitué d'une liste numérotée (repère³) qui contient deux éléments `<li>` de premier niveau (repères · et ²). Chacun d'eux contient alors une liste à puces (repères » et ¶) qui inclut à son tour deux éléments `<li>` de second niveau (repères ¿ et ´ puis º et ¾).

Exemple 3-11. Les listes imbriquées

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les listes imbriquées</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <h1>Listes imbriquées</h1>
  <ol> ³
   <li><h2>Le premier jour</h2>·
    <ul> »
     <li><p>In principio creavit Deus caelum et terram terra autem erat inanis et
        ➥ vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
        ➥ dixitque Deus fiat lux ..</p>
     </li>  ¿
     <li><p>Benedixitque illis Deus et ait crescite et multiplicamini et replete
        ➥ terram et subicite eam et dominamini piscibus maris et volatilibus caeli
        ➥ et..</p>
```

```
    </li>
   </ul>
  </li>
  <li><h2>Le deuxième jour</h2>
   <ul>
    <li><p>In principio creavit Deus caelum et terram terra autem erat inanis et
     ➥ vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
     ➥ dixitque Deus fiat lux ..</p></li>
    <li><p>Benedixitque illis Deus et ait crescite et multiplicamini et replete
     ➥ terram et subicite eam et dominamini piscibus maris et volatilibus caeli
     ➥ et ..</p></li>
   </ul>
  </li>
 </ol>
</body>
</html>
```

La figure 3-12 montre le résultat obtenu.

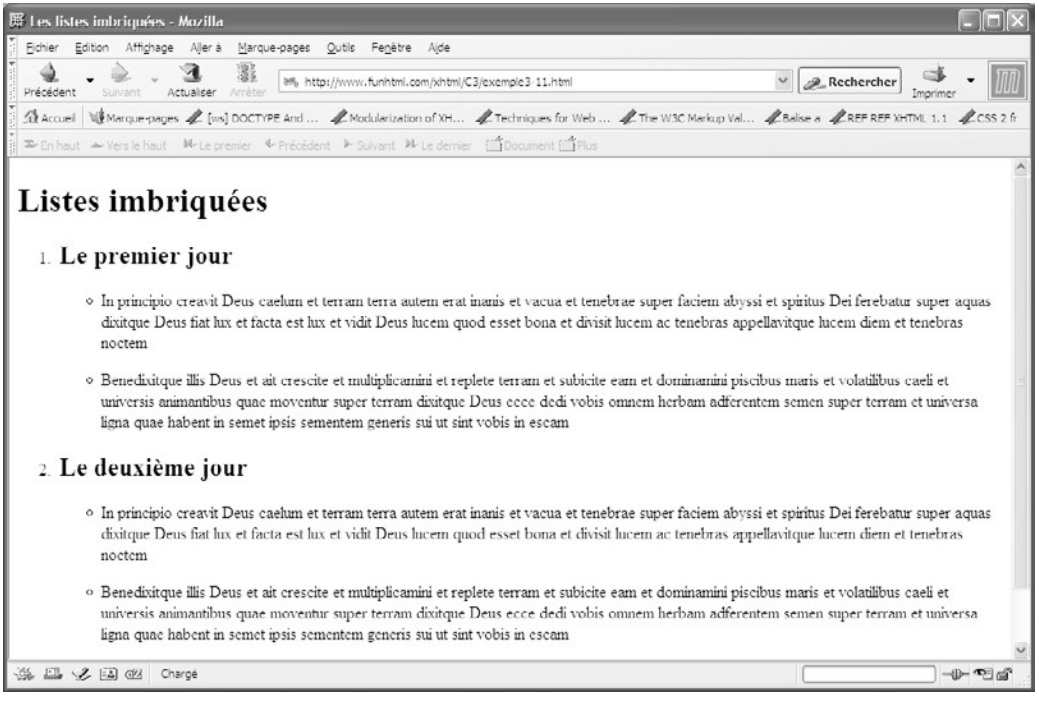


Figure 3-12

Des listes imbriquées

Les listes de définitions

Une liste de définitions permet de créer une liste de termes, chacun d'entre eux étant suivi de sa définition. Le conteneur de l'ensemble de la liste est l'élément <dl> qui ne peut contenir que des éléments <dt>, <dd> ou l'élément <dl> lui-même et rien d'autre. Le plus souvent, l'élément <dt> contient le terme et <dd> en renferme la définition. Par défaut, la définition est affichée à la ligne et indentée par rapport à la ligne du terme. L'élément <dl> peut être inclut directement dans le corps du document <body> mais aussi dans d'autres éléments divers, dont la liste figure dans le tableau 3-13.

Tableau 3-13. Les éléments parents de <dl>

blockquote, body, button, dd, del, div, fieldset, form, ins, li , map, noscript, object, td, th

L'élément <dt> peut contenir tous les éléments en ligne dont la liste figure dans le tableau 3-14.

Tableau 3-14. Les éléments enfants de <dt>

Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var

L'élément <dd> a un contenu plus étendu car il peut contenir tous les éléments en ligne mais également des éléments de bloc, dont la liste figure dans le tableau 3-15.

Tableau 3-15. Les éléments enfants de <dd>

PCDATA, a, abbr, acronym, address, b, bdo, big, blockquote, br, button, cite, code, dl, dfn, div, dl, em, fieldset, form, h1, h2, h3, h4, h5, h6, hr, i, img, input, ins, kbd, label, map, noscript, object, ol, p, pre, q, samp, script, select, small, span, strong, sub, sup, table, textarea, tt, ul, var

La diversité de ces éléments enfants permet de structurer des informations beaucoup plus variées que de simples définitions de termes. Nous pouvons par exemple envisager de créer des listes d'images ou de paragraphes.

La structure de base d'une liste de définitions est donc la suivante :

```
<dl>
  <dt>Terme 1</dt>
  <dd>Définition 1</dd>
```

```
  <dt>Terme 2</dt>
  <dd>Définition 2</dd>
</dl>
```

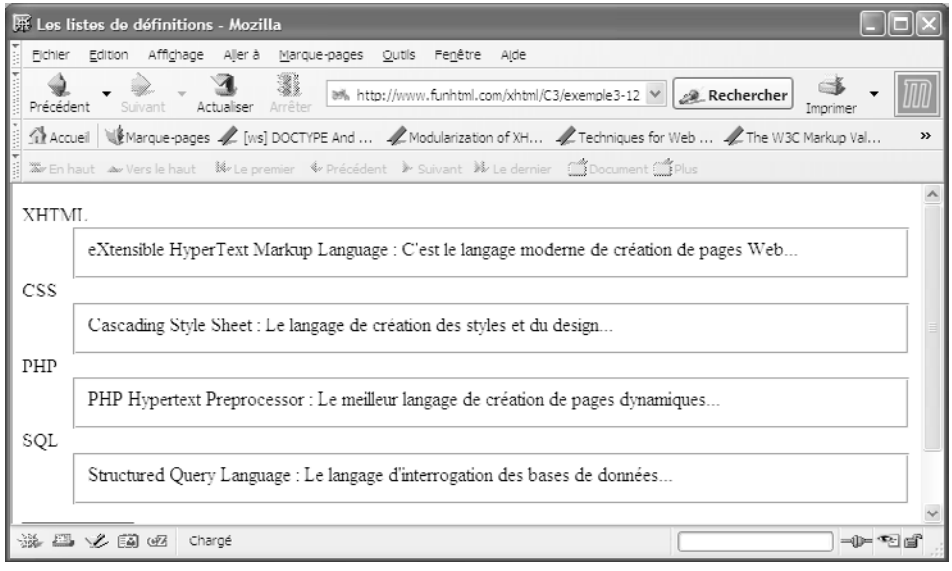
Les navigateurs affichent les définitions indentées de plusieurs caractères par rapport aux termes. L'exemple 3-12 crée une liste de définitions (repère ³) contenant quatre éléments <dt>, lesquels renferment des sigles informatiques (repères · , ¿ , ² , º). Les éléments <dd> qui suivent chacun d'entre eux donnent les définitions contenues dans des éléments <fieldset> qui pourraient s'étendre sur plusieurs lignes (repères » , ´ , ¶ et ¾).

Exemple 3-12. Les listes de définitions

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html  xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les listes de définitions</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <dl> ³
    <dt>XHTML</dt>
    <dd><fieldset>eXtensible HyperText Markup Language : le langage moderne de
      ➥ création de pages web...</fieldset></dd> »
    <dt>CSS</dt>¿
    <dd><fieldset>Cascading Style Sheet : le langage de création des styles et du
      ➥ design...</fieldset></dd> ´
    <dt>PHP</dt>²
    <dd><fieldset>PHP Hypertext Preprocessor : le meilleur langage de création de
      ➥ pages dynamiques...</fieldset></dd¶
    <dt>SQL</dt>º
    <dd><fieldset>Structured Query Language : le langage d'interrogation des bases
      ➥ de données...</fieldset></dd>¾
  </dl>
</body>
</html>
```

La figure 3-13 montre le résultat obtenu avec le style par défaut.

Figure 3-13
Une liste de définitions



Comme les listes numérotées ou à puces, il est possible de créer des listes imbriquées en incluant des éléments `<dl>` dans l'élément `<dd>`

Exercices

Exercice 1 : Écrivez une page contenant des titres de chapitre suivis d'une introduction puis de paragraphes ayant chacun un titre. Choisissez les éléments appropriés à une bonne structuration.

Exercice 2 : Créez deux divisions contenant chacune un titre général et deux paragraphes ayant chacun un titre. Quel intérêt peut avoir cette disposition par rapport à celle de l'exercice 1 ?

Exercice 3 : Écrivez des titres de chansons puis le texte de la chanson dans un bloc de citation. Tracez une ligne de séparation entre chaque bloc.

Exercice 4 : Écrivez un bloc qui contient du code XHTML en préservant les espaces du code.

Exercice 5 : Écrivez une liste d'adresses e-mail indiquant les différents contacts possibles sur le site.

Exercice 6 : Écrivez plusieurs abréviations et leur définition respective dans leur attribut `title` . L'ensemble doit être inclut dans un élément adéquat. Une boîte d'alerte doit afficher cette même définition quand le curseur survole le texte de l'abréviation.

Exercice 7 : Incluez des citations courtes dans le texte d'un paragraphe.

Exercice 8 : Écrivez les textes suivants : $u_n=2(u_{n-1} + u_{n-2})$ et $f(x)=3x^4+7x^3-9x^2$ pour qu'ils s'affichent de cette façon.

Exercice 9 : Incluez du code source de plusieurs balises XHTML dans un texte normal, par exemple <html>ou <head>

Exercice 10 : Écrivez les mots « XHTML » et « CSS » en gras dans un texte normal.

Exercice 11 : Écrivez les mots « XHTML » et « CSS » dans un élément afin de leur appliquer un style particulier par la suite.

Exercice 12 : Écrivez des mots en italique, des mots dans une taille plus grande et d'autres dans une taille plus petite dans un texte inclut dans un paragraphe.

Exercice 13 : Créez une liste de noms numérotée de 1 à 4.

Exercice 14 : Créez la même liste de noms avec des puces.

Exercice 15 : Créez une liste imbriquée. Le premier niveau doit avoir des puces et le second des numéros.

4

Insérer des images et du multimédia

On n'envisage plus aujourd'hui un site sans images, comme paradoxalement peut l'être celui de W3C, qui a l'excuse de ne pas être grand public, mais uniquement très technique et destiné aux spécialistes. Les sites grand public se doivent donc d'égayer leurs pages avec des illustrations photographiques ou simplement graphiques. L'abus d'images pouvant se révéler aussi nocif que leur absence en termes d'attractivité du site pour le visiteur, il appartient au concepteur d'effectuer un choix judicieux de ses illustrations. De plus, il faut encore tenir compte du poids des images en kilo-octets, car les connexions ADSL ne sont pas encore majoritaires. Le poids des images est sans commune mesure avec celui du code d'une page XHTML et vient rapidement allonger le temps d'affichage complet d'une page si elle en contient beaucoup.

Les types d'images

Les navigateurs actuels n'acceptent qu'un nombre restreint de types d'images et il faudra vous limiter aux trois grands types les plus utilisés et enregistrer vos images selon les formats présentés ci-après, qui sont suffisants pour satisfaire tous les besoins d'un concepteur de sites :

- GIF : Graphics Interface Format, ayant pour extension `.gif`. Ce format est limité à 256 couleurs et est donc déconseillé pour les photographies ayant une grande plage de teintes différentes. On l'utilisera en priorité pour des icônes, des dessins ou des bandeaux publicitaires car il présente aussi l'avantage de permettre la création de petites animations

et l'entrelacement qui va permettre l'affichage progressif de l'image, d'abord en basse résolution puis, au fur et à mesure du chargement, à la résolution maximale.

- JPEG : Joint Photographic Experts Group, ayant pour extensions `.jpeg` ou `.jpg`. Ce format permet la création d'images en 24 bits (16 millions de couleurs), et est donc très adapté aux photographies réalistes. En contrepartie toutefois, les images JPEG ont habituellement un poids plus important en Ko, ce qui ralentit leur chargement.
- PNG : Portable Network Graphics, ayant pour extension `.png`. Ce format est assez récent et a été conçu comme alternative au format GIF qui faisait l'objet (théoriquement, car qui s'en préoccupait ?) de droits d'auteurs, les inventeurs de ce format pouvant vous réclamer des royalties. Le format PNG créé à l'initiative du W3C est donc libre de droit et permet normalement la création de graphiques et de photographies. Comme il est de création récente, les vieux navigateurs (de génération 3) ne le reconnaissent pas, mais comme ils sont désormais très minoritaires, vous pouvez l'employer sans trop de crainte.

À première vue, une image GIF ou PNG aura un poids plus faible qu'une image JPEG ; cependant, c'est la variété des teintes de l'image (la palette des couleurs) qui va influencer le poids final de l'image. Quand vous créez une image, essayez donc systématiquement de l'enregistrer sous les trois formats. Vous serez parfois surpris par le poids réel des images obtenues et vous pourrez choisir la moins lourde, donc celle qui s'affichera le plus vite dans votre page. Ne vous fiez pas non plus au temps d'affichage des images dans votre navigateur lors de vos tests, car le fichier provient directement de votre disque dur dans le navigateur, et ce temps de chargement n'a rien à voir avec celui que les visiteurs du site auront à subir si votre image « pèse » 100 Ko par exemple. Pour faire bonne mesure, diminuez la définition des photos numériques pouvant atteindre une taille de 2 Mo. De plus, on estime généralement qu'après 5 secondes d'attente, un visiteur moyen commence à s'impatienter. Testez donc vos pages en ligne pour vous mettre dans la situation de vos visiteurs.

L'insertion d'images

L'élément <img />

L'élément `<img />` permet d'inclure des images dans une page web. Il s'agit d'un élément de type en ligne et généralement il doit être inclus directement dans un élément de type bloc, dans une liste, ou encore un grand nombre d'autres éléments en ligne. C'est un élément vide, d'où l'utilisation du symbole de fermeture incorporé à la fin de la balise d'ouverture.

Le tableau 4-1 donne la liste de tous les éléments parents de l'élément `<img />`

Tableau 4-1. Les éléments enfants de l'élément <img />

a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, q, samp, small, span, strong, sub, sup, td, th, tt, var

L'élément <img /> est qualifié d'élément remplacé car l'affichage obtenu n'est pas le contenu de l'élément qui est vide. C'est en agissant sur ses attributs que l'on peut définir l'image et ses caractéristiques. L'image obtenue à l'écran provient d'un fichier externe simplement référencé par l'attribut src. En outre, les dimensions de l'image ne sont pas nécessairement égales à celles de l'image originale car elles sont fixées par les attributs height et width. L'élément <img /> possède tous les attributs communs, tels id et title, et les gestionnaires d'événements communs. Nous allons voir maintenant les différents attributs de cet élément qui permettent de bien gérer les images.

- L'attribut src : précise l'adresse relative ou absolue du fichier image que l'on désire afficher dans la page. Son utilisation est donc obligatoire. Nous écrivons par exemple :

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

pour une adresse relative, ce qui suppose que le fichier PNG est situé dans le même répertoire que le fichier XHTML qui l'incorpore. En écrivant :

```
<img src="http://www.votresite.com/images/image.png" />
```

nous définissons une adresse absolue, l'image pouvant donc figurer sur un autre serveur que le nôtre. L'utilisation des adresses relatives est préférable en vue de faciliter la maintenance du site. Si le fichier n'est pas disponible, les navigateurs affichent une icône à la place.

- L'attribut alt : sa présence est également nécessaire et il doit contenir un texte fournissant une brève description de l'image. Ce texte apparaîtra à la place de l'image si celle-ci n'est pas disponible (absente du répertoire cible ou illisible). Il est également utilisé par les navigateurs n'affichant pas les images et il sera lu par les navigateurs spécialisés pour non-voyants. L'attribut title peut contenir un texte similaire.
- Les attributs height et width : permettent de définir respectivement la hauteur (height) et la largeur (width) qu'aura l'image sur le média d'affichage. Ces dimensions peuvent être définies à l'aide d'un pourcentage. Dans ce dernier cas, les pourcentages font référence aux dimensions de l'élément parent de l'image. Si la fenêtre du navigateur est redimensionnée, les dimensions de l'image seront alors recalculées, laissant l'image entièrement visible. La définition de ces attributs accélère l'affichage dans les navigateurs car elle leur permet de déterminer la zone d'affichage avant même d'avoir téléchargé l'image. Le poids de l'image en kilo-octets ayant une influence évidente sur le temps de chargement de la page, il est déconseillé de définir des dimensions nettement inférieures à celles de l'image originale. Il vaut mieux dans ce cas créer un autre fichier image de dimensions réduites dont le poids sera plus petit. Si le rapport entre les valeurs des attributs height et width n'est pas le même que celui de l'image réelle, il en résulte une déformation de cette dernière, sans doute peu esthétique (à moins que cela ne soit volontaire). Il faut donc vérifier les dimensions réelles de l'image avant d'utiliser ces attributs.
- L'attribut longdesc : contient l'URL d'un document écrit qui peut fournir des explications beaucoup plus détaillées que celles des attributs alt ou title. Les navigateurs n'affichent pas directement ce contenu ni l'URL indiqué. Pour exploiter cette

information, il faudra utiliser la propriété CSS `content` qui permet de récupérer la valeur de l'attribut `longdesc` et d'afficher l'URL de la description.

- L'attribut `ismap` : s'il est utilisé, il précise qu'il existe une grille créant des zones sensibles au clic dans l'image. Ces zones sont l'origine de liens hypertextes depuis l'image vers des documents externes ou des parties de la page. L'attribut prend la valeur booléenne unique `ismap`. Il est toujours associé à l'attribut `usemap` présenté ci-après.
- L'attribut `usemap` indique l'identificateur (l'attribut `id`) de l'élément `<map>` qui définit les zones sensibles au clic de l'image. La valeur de l'identificateur doit être précédée du caractère dièse (#). Nous allons voir dans la section suivante comment délimiter ces zones.

L'exemple 4-1 permet d'incorporer plusieurs images dans une page. La première est incluse dans un paragraphe et possède les dimensions initiales du fichier image (soit 466 × 378 pixels) car les attributs `height` et `width` ne sont pas précisés (repère ³). La deuxième, incluse dans le même élément `<p>`, a des dimensions fixées à 20 % des dimensions réelles de l'image (repère ·). Remarquons que si l'élément parent n'a pas une hauteur fixée, il n'est pas utile de définir une hauteur d'image en pourcentage car cela peut causer des déformations de cette dernière. La seule définition de la largeur suffit, et les proportions sont ainsi conservées. La troisième à être incluse dans un élément `<div>` est du type TIFF et ne peut pas être affichée directement par les navigateurs (repère »). C'est donc le contenu de l'attribut `alt` qui apparaît à sa place. Les images suivantes sont incluses dans des éléments d'une liste non ordonnée avec des dimensions fixées explicitement en pixel et en donnant une adresse absolue (repères ¿ et ´).

Exemple 4-1. Insertion d'images dans une page

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Insertion d'images</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <h1>Insertion d'images</h1>
  <p>
    ³ <img src="../images/xjr1300.jpg" alt="Ma moto : Yamaha XJR 1300"
    ➥ title="Yamaha XJR 1300: pas mal!" />
    · <img src="../images/xjr1300.jpg" alt="Ma moto : Yamaha XJR 1300"
    ➥ title="Yamaha XJR 1300: fun!!" width="20%"/>
   <!-- Ne pas définir la hauteur en pourcentage sauf si le conteneur a une hauteur
    ➥ connue -->
  </p>
  <div>
```

```
    » <img src="romy.tif" alt="Cette image n'est pas accessible"/>
  </div>
  <ul>
    ¿ <li><img src="http://funhtml.com/images/xjr1300.jpg" alt="xjr 1300" width="45"
    ➥ height="35" /> YAMAHA XJR 1300</li>
    ´ <li><img src="http://funhtml.com /images/zrx1200.jpg" alt="zrx 1200"
    ➥ width="45" height="35" /> KAWASAKI ZRX 1200</li>
  </ul>
 </body>
</html>
```

Les figures 4-1 et 4-2 montrent les résultats obtenus quand le navigateur occupe toute la fenêtre (figure 4-1) et quand celle-ci est réduite (figure 4-2). Nous pouvons y remarquer que, si la première image a toujours la même taille, la deuxième a été réduite pour avoir toujours une largeur de 20 % du paragraphe qui la contient. Cette dernière montre l'avantage de l'utilisation d'un dimensionnement en pourcentage qui laisse l'image entièrement visible quelle que soit la taille de l'écran du visiteur. En effet, une forte réduction de la taille de la fenêtre du navigateur ne permettrait pas de voir la première image dans son intégralité.

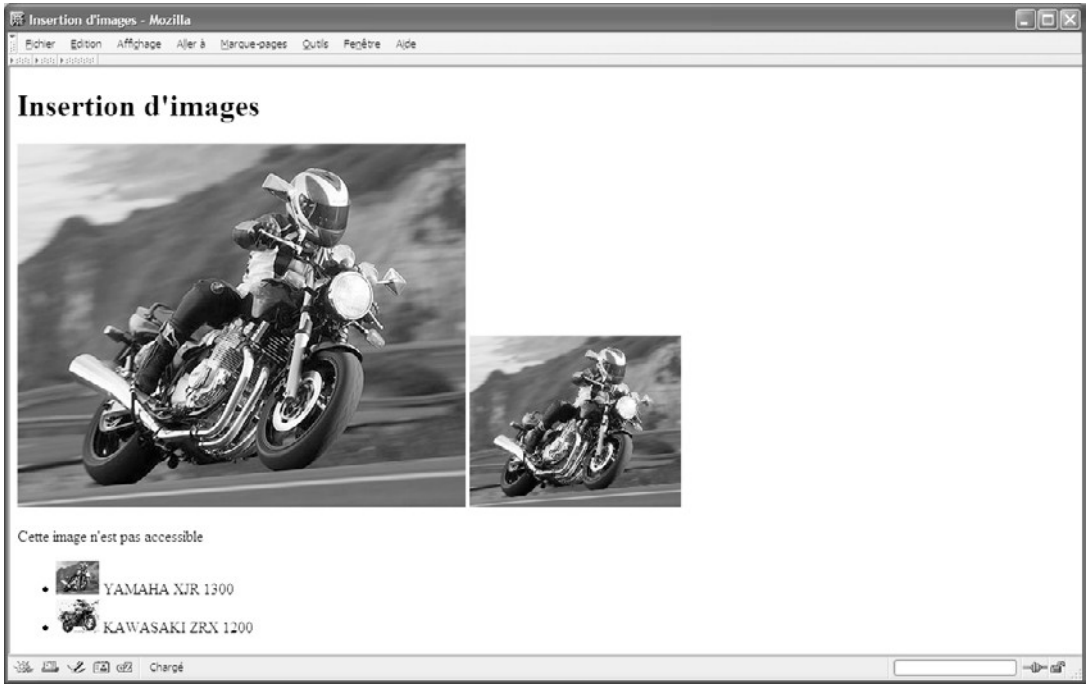
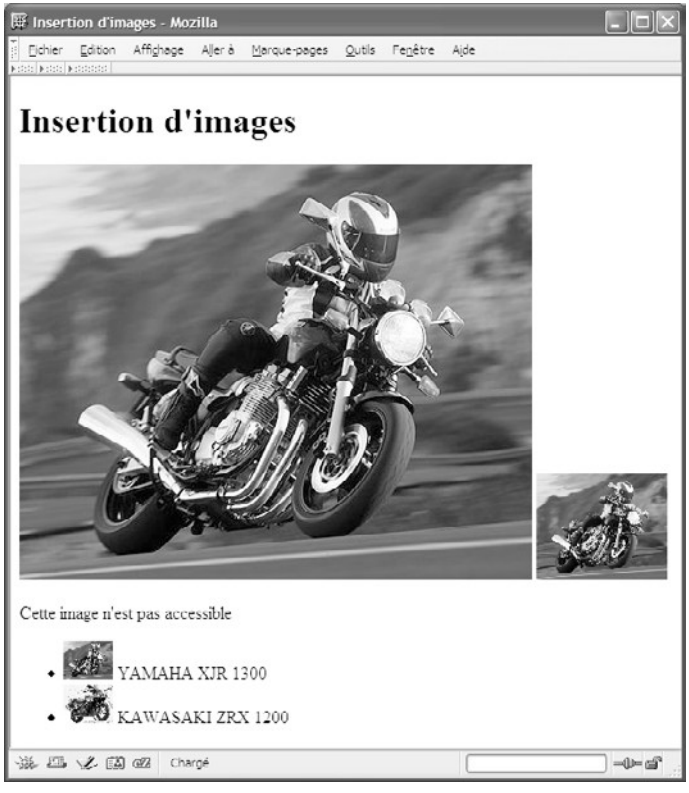


Figure 4-1
L'insertion d'images dans une fenêtre entière

Figure 4-2
L'insertion d'images dans une fenêtre réduite



Les images réactives

Grâce aux gestionnaires d'événements `onclick` ou `onfocus` il est possible de déclencher un script JavaScript en réponse à l'action d'un visiteur. Cependant, ces événements ont lieu quel que soit le point de l'image où ils se produisent. Pour affiner ce comportement, il est possible de créer une carte contenant la définition de plusieurs zones différentes de la même image, chacune d'elles pouvant par exemple déclencher un script différent ou être à l'origine d'un lien vers un autre document, comme nous le verrons au chapitre 5. Une telle carte contient la définition de toutes les zones sensibles de l'image. Une carte est définie à l'aide de l'élément `<map>`. Ce dernier possède l'ensemble des attributs communs. La présence de l'attribut `id` est obligatoire car c'est lui qui permet d'associer la carte à une image à l'aide de l'attribut `usemap` de l'élément `<img />`. La même carte peut donc servir pour plusieurs images différentes. L'élément `<map>` doit contenir la définition d'au moins une zone sensible. Chaque zone est créée par un élément `<area />`.

La définition d'une carte doit donc avoir la structure suivante :

```
<map id="carte1">
  <!--Définition de la zone 1ère zone-->
```

```
<area alt="zone1" />
<!--Définition de la 2ème zone-->
<area alt="zone2" />
…
</map>
```

On constate que l'élément `<area />` est un élément vide. La définition des zones se fait par l'intermédiaire de certains de ses attributs dont voici la liste.

- L'attribut `accesskey` : sa valeur est un caractère qui permet de créer un raccourci clavier (Alt + caractère sous Windows) simulant un clic sur la zone définie.
- L'attribut `alt` : contient un texte qui joue le même rôle que dans l'élément `<img />` et il est également obligatoire.
- L'attribut `href` : contient l'URL du document cible qui sera affiché après un clic sur la zone. Il est facultatif car le concepteur peut vouloir qu'un clic sur la zone entraîne une autre action, le déclenchement d'un script par exemple.
- L'attribut `nohref` : prend la valeur booléenne unique `nohref` pour définir explicitement que la zone sensible ne dirige pas vers un autre document.
- L'attribut `tabindex` : sa valeur est un nombre entier qui donne une position à la zone sensible dans l'ordre de tabulation. Si par exemple nous écrivons `tabindex="2"`, il suffit à l'utilisateur de taper deux fois sur la touche Tab pour rendre la zone active. La frappe de la touche Entrée équivaut alors à un clic sur la zone.
- L'attribut `shape` : permet de définir la forme de la zone sensible. Il peut prendre quatre valeurs :
  - `rect` pour un rectangle, qui est la valeur par défaut ;
  - `circle` pour un disque ;
  - `poly` pour un polygone quelconque ;
  - `default` pour gérer les clics qui sont effectués en dehors d'une des zones sensibles définies sur l'image.
- L'attribut `coords` : vient en complément du précédent pour indiquer les dimensions de la forme choisie. L'origine des coordonnées est le sommet supérieur gauche de l'image, et celles-ci sont exprimées en pixel sans avoir à préciser l'unité `px`.
  - Si `shape="rect"`, il faut indiquer dans cet ordre les coordonnées des sommets supérieur gauche et inférieur droit du rectangle sous la forme :

    ```
    shape="rect" coords="XH, YH, XB, YB"
    ```

  - Si `shape="circle"`, il faut indiquer les coordonnées du centre suivies du rayon sous la forme :

    ```
    shape="circle" coords="XC, YC, R"
    ```

– Si `shape="poly"`, il faut donner les coordonnées de tous les sommets dans l'ordre et terminer cette liste en répétant celles du point de départ. Pour un triangle, nous pouvons écrire :

```
shape="poly" coords="X1, Y1, X2, Y2, X3, Y3, X1, Y1"
```

L'élément `<area />` possède également deux attributs gestionnaires d'événements particuliers à partir desquels on peut gérer les actions du visiteur :

- L'attribut `onfocus` : permet de gérer l'événement qui se produit quand la zone sensible reçoit le focus, ce qui peut se reproduire à l'aide de la souris ou en utilisant la touche Tab ou un raccourci clavier si les attributs `tabindex` ou `accesskey` ont été définis.

- L'attribut `onblur` : permet de gérer l'événement qui se produit quand la zone perd le focus par les mêmes moyens.

L'exemple 4-2 permet la réalisation d'une image réactive. La création de la carte à l'aide des éléments `<map>` et `<area />` amène à définir trois zones. La première est un rectangle (repère ³), la deuxième un disque (repère ») et la troisième un polygone (repère ´).

La définition de l'attribut `href` déclenche l'affichage d'une nouvelle page après un clic sur la première zone. Pour la deuxième zone, l'attribut `nohref` précise que le clic ne redirige pas vers une autre page. Il provoque en revanche l'exécution d'un script JavaScript (repère · , ici une simple boîte d'alerte). La définition de l'attribut `onfocus` pour la troisième zone entraîne l'affichage d'une boîte d'alerte JavaScript quand elle reçoit le focus (repère ¿). La carte ainsi définie est utilisée par une image (repère ¶) dont l'attribut `usemap` permet d'identifier l'élément `<map>` concerné (repère ²).

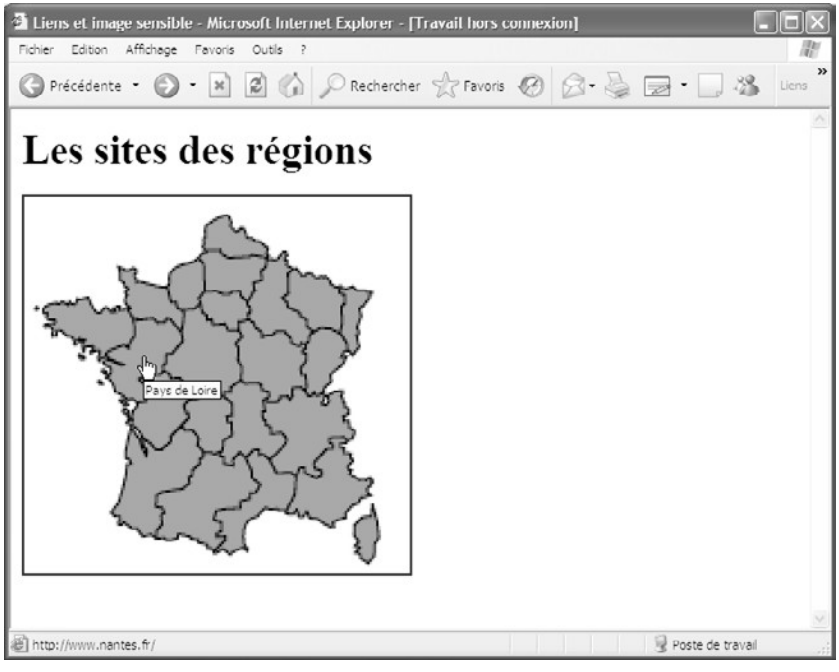
Exemple 4-2. Création d'une image réactive

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Liens et image sensible</title>
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
 <div>
  <h1>Les sites des régions</h1>
  <map id="regions">
   <area href="http://www.orleans.fr" alt="Région Centre" title="Région Centre"
    ➥ shape="rect" coords="142,118,188,180" /> ³
```

```
 <area nohref="nohref"  ·  onclick="alert('Visitez la plus belle ville du monde')"
   ➥ alt="Région parisienne" title="Région parisienne" shape="circle"
   ➥ coords="180,98,16" />»
 <area href="http://www.nantes.fr"  ¿  onfocus="alert('Nantes en plein
   ➥ développement')" alt="Pays de Loire" title="Pays de Loire" shape="poly"
   ➥ coords="76,146,95,110,138,130,94,182,76,146" />
 </map>
 <img ²  usemap="#regions" src="../images/france.gif" alt="Carte des régions"
   ➥ width="344" height="336"/>¶
</div>
</body>
</html>
```

La figure 4-3 montre le résultat obtenu et la forme du curseur au survol des liens pour la troisième zone.

Figure 4-3
Une image réactive



L'insertion d'image en tant qu'objet

Nous avons vu que les navigateurs ne supportent qu'un nombre limité de formats d'image. Si par exemple nous essayons d'insérer une image TIFF dans une page au moyen de l'élément <img />, cette image ne sera pas affichée (comme nous l'avons vu à la figure 4-1). Pour pouvoir afficher un plus grand nombre de documents graphiques et

multimédias, le W3C a introduit depuis HTML 4 le nouvel élément <object> qui peut être inclus dans de nombreux autres éléments de bloc ou en ligne. Le tableau 4-2 rappelle la liste des différentes valeurs possibles pour l'élément <object>.

Tableau 4-2. Les éléments parents de <object>

a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, head, i, ins, kbd, label, legend, li, object, p, q, samp, small, span, strong, sub, sup, td, th, tt, var

Il peut contenir à son tour de nombreux éléments dont la liste figure dans le tableau 4-3.

Tableau 4-3. Les éléments enfants de <object>

Texte, a, abbr, acronym, address, b, bdo, big, blockquote, br, button, cite, code, del, dfn, div, dl, em, fieldset, form, h1, h2, h3, h4, h5, h6, hr, i, img, input, ins, kbd, label, map, noscript, object, ol, p, param, pre, q, samp, script, select, small, span, strong, sub, sup, table, textarea, tt, ul, var

L'élément le plus couramment employé dans <object> est l'élément <param /> qui est un élément vide et qui permet de passer à l'application qui lit le fichier multimédia, et dont un certain nombre de paramètres influencent la lecture. Ses paramètres sont donnés sous la forme de paire nom/valeur dans les attributs name et value de l'élément. En voici la forme générale :

```
<param name="nom_param" value="valeur_param"/>
```

Les valeurs de ces paramètres varient selon le type du fichier à lire. Pour une animation ou un fichier son que l'on veut lire en boucle, nous pouvons écrire par exemple :

```
<param name="loop" value="true"/>
```

Avant d'utiliser un tel paramètre, il est bon de vérifier s'il correspond bien à l'application et s'il est bien accepté par les différents navigateurs. L'ajout d'un élément <param /> inapproprié bloque parfois la lecture du fichier dans certains navigateurs, tels qu'Internet Explorer ou même Mozilla, alors même que son absence permet la lecture correcte du fichier.

Le contenu de l'élément <object> quand il est différente de l'élément <param /> ne sert que d'alternative en cas de non-affichage de l'objet multimédia et, normalement, n'est donc pas visible.

L'élément <object> permet entre autres d'afficher des images et possède un grand nombre d'attributs. Nous n'allons utiliser ici que les attributs utiles à l'insertion d'image et nous recourons aux autres dans les sections suivantes consacrées à l'insertion de contenus multimédias.

- L'attribut data : contient l'URL relatif ou absolu de l'image à afficher. C'est l'équivalent de l'attribut src de l'élément <img />

- L'attribut `type` : précise le type MIME du fichier indiqué dans l'attribut `data`.
- L'attribut `standby` : contient le texte qui est affiché pendant le temps de chargement de l'image. Il équivaut à l'attribut `alt` de l'élément `<img />`.
- Les attributs `width` et `height` : permettent de définir les dimensions de l'objet dans la page, et en particulier celle d'une image.

On retrouve également les attributs `usemap` et `tabindex` qui jouent le même rôle que dans l'élément `<img />` En particulier l'attribut `usemap` permet de créer des zones réactives dans l'image. Dans l'exemple 4-3 nous incluons d'abord une image de type JPEG au moyen de l'élément `<object>` aussi facilement qu'avec l'élément `<img />` (repère ³). Le contenu de l'élément `<object>` n'est pas visible et ne sert que de solution de substitution dans le cas où l'image ne serait pas affichée. L'insertion d'une image de type TIFF, qui était impossible avec `<img />`, devient alors possible avec `<object>` en précisant le type et les dimensions voulues pour l'image (repère ·). De même, l'insertion d'une image dans un format exotique pour le Web, puisqu'il s'agit de la même image mais au format BMP, est aussi réalisable en utilisant l'élément `<object>` et en précisant encore le type et les dimensions du rendu désiré (repère »). La largeur de l'image est ici donnée en pourcentage.

Exemple 4-3. Insertion d'images en tant qu'objet

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <title>Les objets images</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <div>
   <!--  -->
   <object data="../images/xjr1300.jpg" id="img1" width="400" height="350"
    ➥ title="yamaha XJR 1300">³
    <h1>Moto classe</h1>
   </object>
   <!--  -->
   <object data="romy.tif" id="img2" title="Romy" type="image/tiff" standby="Romy
    ➥ en tifs" width="400" height="300">·
    <h1>ROMY en format TIFF</h1>
   </object>
   <object data="romy.bmp" id="img3" title="Romy" type="image/bmp" standby="Romy
    ➥ en tifs" width="30%" height="350">»
    <h1>ROMY en format BMP</h1>
```

```
    </object>
  </div>
 </body>
</html>
```

La figure 4-4 montre les images affichées dans le navigateur Mozilla. Pour réaliser cet affichage, certains navigateurs peuvent nécessiter l'emploi d'un plug-in comme QuickTime ou Windows Media Player installés par défaut ou que l'utilisateur doit télécharger. Il est toujours utile de créer un lien dans le site vers les sites des fabricants de ces plug-ins.

Figure 4-4
Affichage d'images en tant qu'objets

Image et bouton

Il est courant de rencontrer dans des pages web des images ayant l'apparence de boutons sur lesquels l'utilisateur peut cliquer pour déclencher une action gérée par un script (écrit par exemple en JavaScript) exécuté côté client, ou par un script (écrit par exemple en PHP) côté serveur. Ce type d'images implique la création de deux images, l'une correspondant à l'état initial du bouton et la seconde à l'état du bouton quand l'utilisateur clique dessus. L'affichage successif de ces images en cas de clic crée un effet d'enfoncement du bouton. Pour créer le même effet sans avoir recours à la création de plusieurs images, il existe l'élément `<button>` apparu en HTML 4. Créé pour être utilisé plus particulièrement dans les formulaires (voir le chapitre 7), il peut être incorporé en dehors de ce contexte pour déclencher des scripts côté client. Comme il s'agit d'un élément en ligne, il devra être inclus dans un bloc `<p>` ou `<div>` par exemple.

Pour que la face du bouton affiche une image, il suffit d'inclure un élément <img /> dans l'élément <button>

Le code de création d'un bouton avec image a donc la structure suivante :

```
<button>
 <img alt="Texte" src="image.gif"/>
</button>
```

L'élément <button> possède l'ensemble des attributs communs et par exemple le gestionnaire d'événements onclick qui va nous permettre de déclencher l'exécution d'un script à la suite d'un clic. Comme pour les éléments <area> nous retrouverons ici les attributs accesskey, tabindex, onblur et onfocus. L'attribut disabled qui prend la valeur booléenne unique disabled rend le bouton inactif. Il ne sera utilisé que pour rendre le bouton inactif à l'aide de code JavaScript après une action particulière d'un utilisateur. L'attribut type permet de déterminer le rôle assigné au bouton. Il peut prendre les valeurs button, submit ou reset, ces deux dernières n'étant utiles que dans le cadre des formulaires. Quand il prend la valeur button, il n'a pas de rôle prédéterminé et c'est au concepteur de lui en donner un, en gérant les attributs gestionnaires d'événements comme onclick par exemple.

L'exemple 4-4 crée une page comportant deux boutons qui contiennent chacun une image (repères ¿ et ¶), et dont l'attribut type a la valeur button. Le premier bouton (repère ·) est réactif au clic par l'utilisation de l'attribut onclick (repère ») et affiche le nom du modèle de la moto représentée par l'image. Le second (repère ´) affiche une boîte d'alerte JavaScript contenant la date et l'heure actuelle en réagissant à l'événement onmouseover (repère ²) quand le curseur survole le bouton. Le style créé (repère ³) permet de créer une couleur de fond pour le bouton.

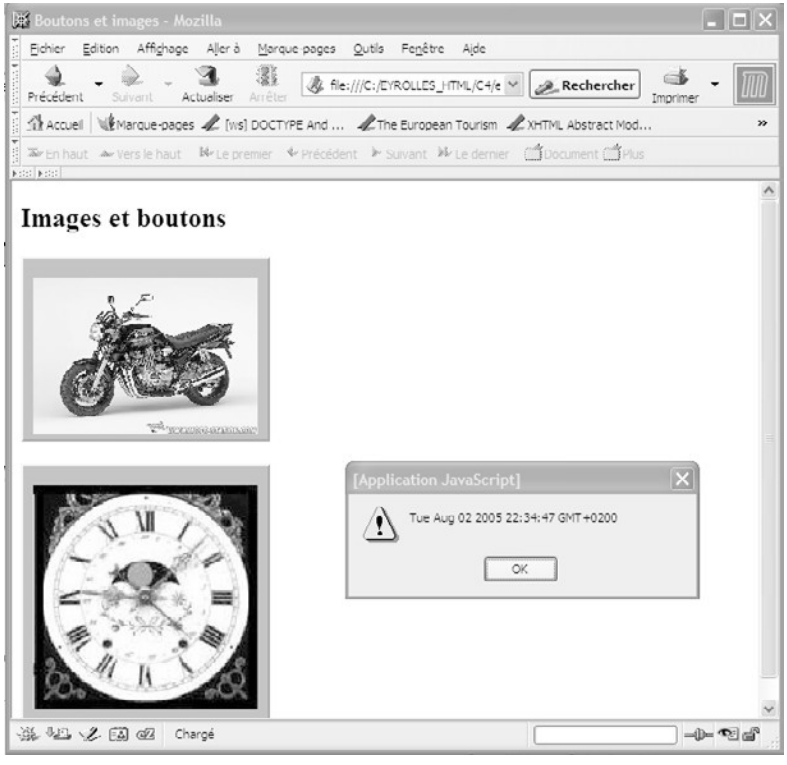
Exemple 4-4. Images et boutons

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
<head>
 <title>Boutons et images</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<style type="text/css" >
 button {background-color: orange;} ³
</style>
</head>
<body>
 <div><h2>Images et boutons</h2>
    · <button type="button" »  onclick="alert('Modèle: YAMAHA XJR 1300')"><br/>
    ¿ <img src="../images/xjr1300a.jpeg" width="200" height="140" alt="Moto Yamaha">
  </button><br/><br/>
```

```
    <button type="button" onmouseover="time=new Date();alert(time)"><br/>
      <img src="../images/horloge.jpg" width="200" height="200"
      alt="La pendule"/><br/><br/>
    </button>
  </div>
 </body>
</html>
```

La figure 4-5 représente les boutons obtenus ainsi que la boîte d'alerte créée par le survol du second bouton.

Figure 4-5
Création de boutons avec une image



L'insertion du multimédia

Les images ne sont pas les seuls éléments décoratifs d'une page web. À cet effet, nous disposons désormais de multiples éléments multimédias qui vont des animations Flash à la vidéo, en passant par les fichiers sons MP3 et les applets Java. Avec le langage XHTML, on peut incorporer ces fichiers multimédias de manière conforme en oubliant les balises <embed> <applet>désormais obsolètes. C'est l'élément <object>qui se prête aujourd'hui le mieux à l'inclusion de ces différents composants.

Notez bien cependant que la lecture des contenus multimédias par les navigateurs n'est pas une évidence comme celle du texte d'une page. Il est souvent nécessaire que ceux-ci soient munis de plug-ins spécialisés que le visiteur du site doit avoir installés au préalable, faute de quoi ils ne seront pas visibles ou audibles. Pour cette raison, il est bon de préciser dans la page les adresses de téléchargement de ces plug-ins, tel que le site de Macromedia pour Macromedia Flash Player (http://macromedia.com/shockwave/download/download.cgi?P1_Prod_Version=ShockwaveFlash&promoid=BIOX) ou QuickTime (http://www.apple.com/quicktime/download/win.html).

L'insertion d'une animation Flash

Pour incorporer une animation au format Flash, nous devons, comme le montre l'exemple 4-5, préciser l'adresse relative ou absolue du fichier Flash dont l'extension est .swf dans l'attribut data de l'élément <object> (repère ³). Nous pouvons préciser le type MIME du fichier dans son attribut type (repère ·) ainsi que les dimensions de l'animation dans la page à l'aide des attributs width (repère ») et height (repère ¿). L'attribut standby (repère ´) peut également être précisé pour contenir un texte qui s'affiche pendant le temps de chargement du fichier si celui-ci semble devoir être long pour le visiteur. Le titre <h1> (repère ²) n'est visible que si l'animation n'est pas lue par le navigateur. Il pourrait être complété par un lien vers le site de téléchargement de Macromedia Flash Player.

Exemple 4-5. Inclusion d'une animation Flash

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Objet Flash</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <h1>OBJET FLASH</h1>
  <div>
   <object ³ data="nav.swf" title="FLASH"· type="application/x-shockwave-flash"
    ➥ » width="880"¿ height="165"´ standby="Menu en Flash">
    <h1>OBJET FLASH NON AFFICHE</h1>
   </object>
  </div>
 </body>
</html>
```

La figure 4-6 montre le résultat obtenu par l'incorporation d'une superbe animation Flash du site www.portix.orgqui sert de menu commun à ses pages.

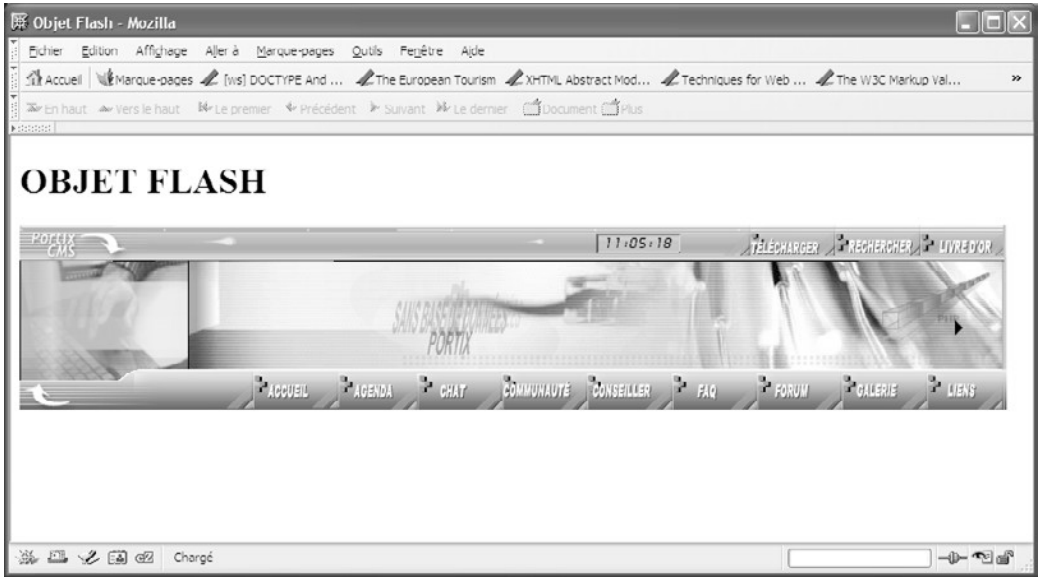


Figure 4-6
Incorporation d'une animation Flash

L'insertion d'une vidéo

L'insertion d'une vidéo dans une page est à distinguer de la lecture simple de ce type de fichier à l'aide d'un logiciel spécialisé, en dehors d'un navigateur, comme chacun peut le faire avec les ordinateurs récents après le téléchargement du fichier. Il s'agit ici que la vidéo soit visible à l'intérieur même de la page comme l'est une image. Nous allons envisager successivement dans l'exemple 4-6 deux techniques qui nécessitent l'emploi d'un plug-in externe, les tests ayant été effectués avec QuickTime installé sur le poste client.

La première méthode est similaire à celle employée dans l'exemple précédent. Nous incluons encore un élément <object>(repère ³) en précisant ses attributs data (repère ·) qui contient le nom du fichier vidéo (ici de type MPEG mais pas nécessairement), l'attribut type (repère ») qui donne le type MIME de ce fichier ainsi que les attributs width (repère ¿) et height (repère ´) pour préciser les dimensions de la zone qui contient la fenêtre de visualisation du plug-in. Cela ne veut pas dire que l'image vidéo prend les dimensions de cette fenêtre, car contrairement aux images fixes, elle garde ses dimensions initiales déterminées lors de sa création. Cette méthode simple fonctionne dans tous les navigateurs récents comme Explorer, Mozilla, FireFox Opera et Netscape, à cette

réserve près que, pour fonctionner dans Explorer, il ne faut pas définir de paramètres du type :

```
<param name="autoplay" value="true" />
<param name="loop" value="false" />
```

afin que, d'une part, la lecture de la vidéo démarre tout de suite sans intervention du visiteur, et d'autre part qu'elle ne soit pas lue en boucle.

La seconde méthode utilise encore un élément `<object>` (repère ¶). Elle consiste à employer un contrôle ActiveX dont le code est indiqué dans l'attribut `classid` (repère º), l'attribut `codebase` indiquant au navigateur l'adresse du code qui est censé lui permettre de visualiser le fichier, n'étant pas obligatoire. Les attributs `width` (repère ¾) et `height` (repère µ) ont pour rôle de dimensionner la zone occupée par l'objet. Contrairement à la première méthode, nous pouvons ici employer autant d'éléments `<param />` que souhaité pour agir sur la vidéo. Le paramètre nommé `src` donne l'adresse du fichier placé ici dans le même répertoire que le document XHTML (repère ¸). Avec la valeur `false` (faux) le paramètre `autoplay` (repère ¹) oblige le visiteur à commander lui-même le démarrage de la vidéo. Le paramètre `controller` (repère) avec la valeur `true` (vrai) permet d'afficher les boutons de commande de lecture, de pause et de rembobinage. Quant à celui nommé `loop` déjà cité (repère), ayant la valeur `false`, il empêche la lecture en boucle. La seconde méthode est acceptée par Explorer, Opera, mais pas par Mozilla ni FireFox.

Les paragraphes (repères ² et) inclus dans chacun de ces objets permettent un affichage alternatif en cas d'absence d'un plug-in adapté à la lecture de la vidéo. Ils contiennent un conseil de téléchargement (repères et) et un lien direct vers le fichier vidéo que l'utilisateur peut alors visualiser plus facilement dans un lecteur multimédia externe.

Exemple 4-6. Lecture d'une vidéo

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Objet Vidéo</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <h1>OBJET VIDEO</h1>
  <div>
   <!—Première méthode-->
    ³ <object · data="romy.mpg" title="Romy» type="video/mpeg" standby="Video
    ➥ MPEG¿ width="320" ´ height="255">
     ² <p>OBJET VIDEO NON AFFICHE. Vous ne pouvez pas voir cette vidéo : téléchargez
    ➥ <a href="http://www.apple.com"></a>QuickTime.<a href="romy.mpg">Voir la vidéo
    ➥ dans votre lecteur</a></p>
   </object>
```

```
<!—Seconde méthode-->
¶ <object °  classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B" codebase=
➥ "http://www.apple.com/qtactivex/qtplugin.cab"  ¾ width="320"µ  height="255">
  ¸
  ¹ <param name="src" value="romy.mpg"/>
    <param name="autoplay" value="false"/>
    <param name="controller" value="true"/>
    <param name="loop" value="false"/>
    <p>OBJET VIDEO  NON AFFICHE.<br /> Vous ne pouvez pas voir cette vidéo :
  ➥ téléchargez <a href="http://www.apple.com"></a>QuickTime. <a href="romy.mpg">
  ➥ <br />Voir la vidéo dans votre lecteur habituel</a></p>
 </object>
</div>
</body>
</html>
```

La figure 4-7 montre le résultat obtenu dans Opera, dans lequel les deux objets vidéo sont visibles simultanément.

Figure 4-7
Lecture d'une vidéo

L'insertion d'éléments audio

L'insertion de fichiers sonores dans une page se réalise avec un élément `<object>`, tout comme pour les vidéos, comme le montre l'exemple 4-7. Par rapport à l'exemple 4-6, nous modifions simplement les valeurs des attributs `type`, lesquels prennent la valeur `audio/mp3` (repère ³) ou `audio/mpeg` et `data`, contenant le nom du fichier dont l'extension est `.mp3` (repère ·).

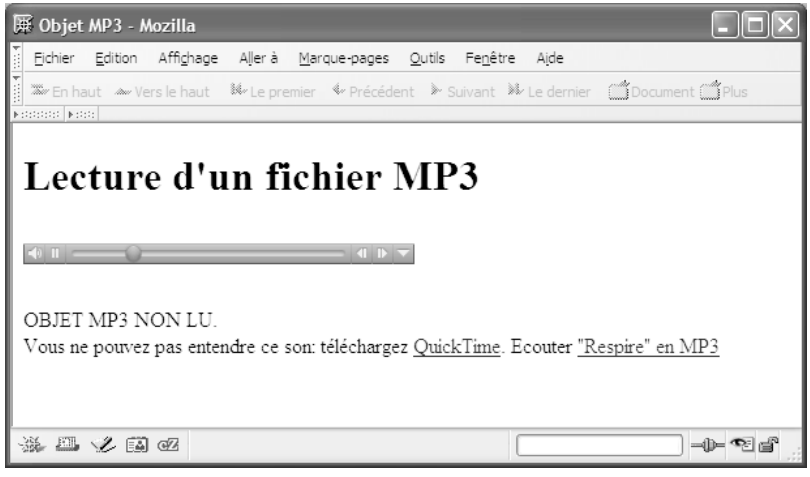
Dans la seconde méthode (repère »), il suffit de modifier la valeur du paramètre dont l'attribut `name` a pour valeur `src` (repère ²). Les attributs `width` (repère ¿) et `height` (repère ´) de l'élément `<object>` peuvent être adaptés pour ne laisser visibles que les boutons de contrôle du lecteur dans la page. Pour diffuser ce type de fichier en musique de fond sans intervention possible du visiteur, il suffit de supprimer ces attributs dans la première méthode ou de mettre le paramètre nommé `controller` (repère ¶) à la valeur `false` dans la seconde.

Exemple 4-7. Lecture d'un fichier MP3

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <title>Objet MP3</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<style type="text/css" >
   /*object {position:absolute;top 20px}
  </style>
 </head>
 <body>
  <div>
   <h1>Lecture d'un fichier MP3 </h1>
   <!-- Première méthode height="40" width="300"-->
   <object ³ type="audio/mp3"  data="respire.mp3" id="mp3" standby="Respire"
    ➥ title="Respire" height="40" width="300">
    <p>OBJET MP3 NON LU.<br /> Vous ne pouvez pas entendre ce son : téléchargez
     ➥ <a href="http://www.apple.com">QuickTime</a>. Écouter <a href="respire.mp3">
     ➥ "Respire" en MP3</a></p>
   </object>
   <!-- Seconde méthode width="300" height="40"-->
    » <object classid='clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B' type="audio/mp3"
    ➥ title="Respire" codebase='http://www.apple.com/qtactivex/qtplugin.cab'
    ➥ ¿ width="300" ´ height="40">
    ² <param name="src" value="respire.mp3" />
   <param name="autoplay" value="false" />
    ¶ <param name="controller" value="true" />
   <param name="loop" value="false" />
   <p>OBJET MP3 NON LU.<br /> Vous ne pouvez pas entendre ce son : téléchargez
    ➥ <a href="http://www.apple.com">QuickTime</a>. Écouter <a href="respire.mp3">
    ➥ "Respire" en MP3</a></p>
   </object>
  </div>
 </body>
</html>
```

La figure 4-8 montre l'affichage obtenu dans Mozilla, navigateur qui ne reconnaît pas la seconde méthode. Dans ce cas, le contenu du paragraphe est affiché, et ainsi on peut cliquer sur le lien qui permet une lecture directe dans un logiciel comme Windows Media Player ou QuickTime.

Figure 4-8
Lecture d'un fichier MP3



Pour lire des fichiers sons de types différents, il faut simplement modifier les attributs `type` et `data`, en écrivant par exemple :

```
<object data="breeze.mid" type="audio/mid">
```

pour un fichier MIDI, ou encore :

```
<object data="logo.wav" type="audio/wav">
```

pour un fichier WAV.

L'insertion d'une applet Java

Les applets sont de petits programmes écrits en langage Java, qui peuvent s'exécuter dans une page web à condition bien sûr que le poste client soit doté des moyens de les exécuter, en l'occurrence de la console Java de Sun ou d'un contrôle ActiveX approprié, et que ceux-ci soient activés dans le navigateur. Les tâches réalisées par ces applets sont des plus variées et on en trouve un très grand nombre en téléchargement gratuit sur Internet.

Le code exécutable de l'applet est contenu dans un ou plusieurs fichiers qui portent l'extension `.class` et qui doivent être présents sur le serveur en même temps que la page XHTML qui les utilise. Pour inclure une applet, nous utilisons encore l'élément `<object>` et non plus `<applet>` désormais obsolète en XHTML. L'exemple 4-8 présente l'incorporation d'une applet (due à Biel Giel nai.net, merci pour lui !) affichant une pendule à aiguilles munie d'une trotteuse en temps réel avec la date du jour en complément.

Comme Explorer d'un côté et les autres navigateurs (Mozilla, FireFox, Opera et Netscape) de l'autre ne traitent pas ces objets de la même manière, et surtout que ces méthodes sont incompatibles, nous avons recours dans l'exemple 4-8 à une astuce consistant à inclure deux éléments `<object>` l'un dans l'autre, chacun utilisant une méthode différente.

De cette manière, l'applet est visible une seule fois dans tous les navigateurs. Pour les navigateurs autres que Explorer, dans le premier élément `<object>` (repère ³), nous précisons le type de code dans l'attribut `codetype` (repère ·) et le nom du fichier Java dans l'attribut `classid` (repère »). Les attributs `width` (repère ¿) et `height` (repère ´) précisent les dimensions de la zone de rendu de l'applet dans la page web. Généralement, chaque applet peut être personnalisée selon les intentions de son créateur en modifiant la valeur de certaines variables. Nous pouvons intervenir sur celles-ci par l'intermédiaire de l'élément `<param />` dont les attributs `name` et `value` correspondent respectivement aux noms des variables Java et à leur valeur. Les noms utilisés figurent soit dans le code source Java s'il est livré avec l'applet, soit dans un fichier texte qui explique sa mise en œuvre.

Dans l'exemple 4-8 nous intervenons sur la couleur de fond (paramètre `bgcolor` repères ² et ¸), sur le texte qui s'affiche sur le cadran de la pendule (paramètre `logoString`, repères ¶ et ¹) et sur le décalage horaire du lieu (paramètre `timezone` repères º et).

La méthode à utiliser spécifiquement pour Explorer fait appel à un module Java identifié dans l'attribut `classid` (repère ¾), le nom du fichier Java étant indiqué dans le paramètre nommé `code` (repère µ). Les autres paramètres (repères ¸, ¹ et) sont identiques aux précédents. L'utilisation dans l'exemple de couleurs de fond différentes pour les deux méthodes permet simplement de vérifier laquelle est employée par chaque navigateur.

Exemple 4-8. Insertion d'une applet Java

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <title>Applet Java</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
 </head>
 <body>
  <div>
  <h1>L'heure exacte</h1>
   <!-- Applet dans Netscape -->
    ³ <object · codetype="application/java" » classid="java:billsClock.class"
    ➥ ¿ width="400" ´ height="400">
     ² <param name="bgcolor" value="00FF00">
    <!-- couleur de fond hexa VERT -->
```

```
    ¶ <param name="logoString" value="XHTML"/>
    º <param name="timezone" value="-2"/>
  <!-- suite des paramètres -->
  <!-- ********************* -->
  <!--Applet dans Explorer-->
  <object ¾ classid="CLSID:8AD9C840-044E-11D1-B3E9-00805F499D93" width="400"
    ➥ height="400">
     µ <param name="code" value="billsClock.class">
     ¸ <param name="bgcolor" value="FF0000">
     ¹ <param name="logoString" value="XHTML"/>
       <param name="timezone" value="-2"/>
   <!--suite des paramètres -->
  </object>
 </object>
</div>
</body>
</html>
```

La figure 4-9 montre le résultat obtenu dans Explorer, mais il est identique dans les autres navigateurs, exception faite de la couleur de fond.

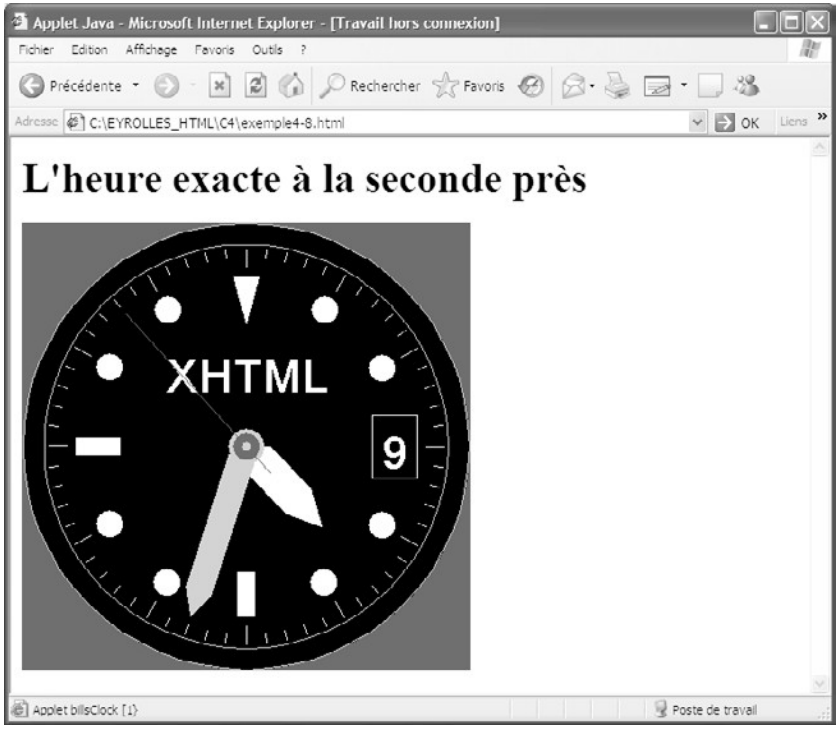


Figure 4-9
Insertion d'une applet dans une page

Exercices

Exercice 1 : Quel type d'images peut-on inclure avec l'élément <img /> ?

Exercice 2 : Dans quels cas et pourquoi a-t-on intérêt à choisir une image au format GIF plutôt qu'au format JPEG ou PNG ?

Exercice 3 : Quels sont les attributs indispensables dans l'élément <img /> ?

Exercice 4 : Incluez une image JPEG dans un paragraphe tout en faisant en sorte qu'elle conserve ses dimensions initiales.

Exercice 5 : Incluez une image GIF dans un titre en la dimensionnant à 50 % de sa taille normale.

Exercice 6 : Créez une image réactive comprenant deux zones sensibles. La première, de forme circulaire, dirige vers un autre site, quelconque, et la seconde, qui est un pentagone, permet l'affichage d'un message de bienvenue. Gérez les clics effectués en dehors de ces deux zones en affichant le message « Hors zone ».

Exercice 7 : Affichez dans une division une image de type TIFF, BMP ou autre format non courant sur le Web. Dimensionnez l'image à 80 %.

Exercice 8 : Créez un bouton dont la face est une image de votre choix. Le clic sur le bouton doit rediriger le visiteur vers le site www.funhtml.com

Exercice 9 : Récupérez ou fabriquez une animation Flash et incorporez-la dans un paragraphe. Créez un lien vers le site www.macromedia.com pour ceux qui ne peuvent pas la visualiser.

Exercice 10 : Insérez une vidéo dans un élément <button> sans utiliser d'éléments <param />

Exercice 11 : Insérez une vidéo dans un élément <button> en utilisant un contrôle ActiveX.

Exercice 12 : Créez une musique de fond à l'aide d'un fichier WAV.

Exercice 13 : Téléchargez une applet Java et installez ses fichiers sur le serveur. Incluez ensuite cette applet dans une page en définissant ses paramètres s'ils existent.

5

Créer des liens

Dans l'acronyme XHTML, on peut considérer que l'élément le plus important est le mot « Hypertext ». En effet, la création de documents contenant des liens hypertextes est la particularité qui a rendu le Web si populaire. Les liens hypertextes permettent de passer, d'un simple clic, sur un mot ou sur une image, d'une page à l'autre, qu'elle soit située sur le même serveur ou en n'importe quel point du réseau. Nous parlerons dans ce cas de lien externe (sous-entendu au document initial). Il est également possible de passer instantanément d'un point de la page à un autre situé dans la même page et identifié par un nom particulier. Nous parlerons alors de lien interne (encore sous-entendu au document initial). Tout contenu essentiel d'une page XHTML peut faire l'objet d'un lien hypertexte externe ou interne. Si l'origine d'un lien est très souvent un clic sur une partie d'un texte, nous pourrons tout aussi bien définir celle-ci sur une image, une partie donnée d'une image, un bouton ou encore créer cette redirection en utilisant un script JavaScript en réaction à d'autres événements que le traditionnel clic. Enfin, nous allons voir qu'un lien peut même permettre de déclencher l'exécution d'un script ou l'ouverture automatique de la messagerie électronique du visiteur pour envoyer un e-mail vers le site ou à l'attention de tout autre destinataire.

Les liens à partir d'un texte

L'élément XHTML primordial pour la création de liens est l'élément `<a>` dont le contenu, situé entre les balises `<a>` et `</a>`, est la partie visible, texte ou image, sensible au clic. Ses attributs permettent de définir la cible du lien et les moyens de le déclencher. Comme il s'agit d'un élément en ligne, il doit être inclus dans un bloc ou tout autre élément admettant comme contenu cet élément. Il peut donc être inclus quasiment en tout point d'une page, aussi bien dans un bloc `<div>` ou `<p>` que dans une liste, une cellule de tableau ou un formulaire.

Les tableaux 5-1 et 5-2 donnent respectivement la liste des éléments parents et enfants de l'élément <a> auxquels il faut se conformer pour créer un document conforme XHTML 1.1.

Tableau 5-1. Éléments parents de l'élément <a>

abbr | acronym | address | b | bdo | big | caption | cite | code | dd | del | dfnl div | dt | em | fieldset | h1 | h2 | h3 | h4 | h5 | h6 | i | ins | kbd | label | legend | li | object | p | pre | q | samp | small | span | strong | sub | sup | td | th | tt | var

Tableau 5-2. Éléments enfants de l'élément <a>

Texte | abbr | acronym | b | bdo | big | br | button | cite | code | del | dfn lem | i | img | input | ins | kbd | label | map | object | q | samp | script | select | small | span | strong | sub | sup | textarea | tt | var

Quand nous définissons un lien dont l'origine est par exemple un texte, celui-ci apparaît souligné, dans une couleur particulière définie par défaut dans le navigateur (généralement en bleu), et le curseur prend l'aspect d'une main pour signaler l'existence de ce lien. Ces conventions visuelles sont communes à tous les navigateurs et, même s'il est possible de les modifier à loisir avec des styles CSS comme nous le verrons au chapitre 12, il faut en user modérément, plus particulièrement pour le soulignement et l'aspect du curseur, car les internautes sont habitués à ces conventions bien établies.

Les liens externes

Les liens externes permettent de diriger le visiteur vers de nouveaux documents non contenus dans la page d'origine. Il peut s'agir de types de fichiers très divers. Les liens les plus usuels permettent de visualiser des documents courants directement dans le navigateur, tels que des pages XHTML ou des images. Mais il est possible de définir des liens vers des documents de type très divers comme des documents PDF, des fichiers compressés selon des formats divers comme ZIP, des animations Flash, des vidéos, des fichiers sons MP3 ou des fichiers MIDI.

Les liens vers des documents courants

Pour indiquer l'adresse URL de la cible d'un lien, il faut donner une valeur à l'attribut href de l'élément <a> Cette adresse peut être relative comme dans le code suivant :

```
<p>La page <a href= "pagecss.html" > CSS 2 </a></p>
```

Dans ce cas, l'aspect du texte est tel que celui représenté à la figure 5-1 et le document pagecss.html doit se trouver sur le serveur, dans le même dossier que la page en cours qui contient le lien. Dans le cas contraire, vous exposez les visiteurs à l'affichage de la page maudite des webmestres, nommée « Erreur 404 ».

L'adresse de la page cible peut aussi être absolue comme dans le code suivant :

```
<div>Visitez le site du <a href= "http://www.w3.org" >W3C </a></div>
```

L'élément `<a>` possède l'ensemble des attributs communs parmi lesquels nous noterons particulièrement `id` qui nous permet d'identifier le lien pour agir sur son contenu à l'aide d'un script JavaScript (voir l'exemple 5-1). L'attribut `title` doit être défini systématiquement car il est utile pour afficher un bref message permettant d'expliquer le contenu de la page cible du lien. De plus, il sera lu par les navigateurs oraux destinés aux non-voyants, ce qui améliore l'accessibilité du site. N'oublions pas non plus l'attribut `class` qui permet d'appliquer des styles spécifiques aux liens (voir le chapitre 12, *Le style du texte et des liens*). En plus de ces attributs communs, l'élément `<a>` possède un grand nombre d'attributs particuliers dont les définitions sont les suivantes :

- `accesskey` : il permet de créer un raccourci clavier pour ouvrir le document cible sans effectuer de clic sur le contenu de l'élément `<a>` Sa valeur doit être un caractère spécifique, de préférence une lettre. L'utilisateur accède alors à la cible en tapant le raccourci Alt+caractère. Cette possibilité doit être signalée au visiteur, soit en clair par un texte explicatif, soit en soulignant le caractère choisi, généralement dans un des mots du lien, comme le font les menus de nombreux logiciels. Il faut veiller à ne pas utiliser les mêmes raccourcis que ceux qui existent dans les navigateurs (comme Alt+F par exemple qui ouvre le menu Fichier) car ils agiraient sur le navigateur et non sur le lien.

  Exemple de code :

```
<div>Visitez le site du <a href= "http://www.w3.org" accesskey="W"
➥ title="W3C:raccourci Alt+W">W3C </a></div>
```

- `tabindex` : il définit un ordre d'accès au lien avec une tabulation, ce qui rend le lien actif si la page en comporte plusieurs. En écrivant le code suivant, il faudra que l'utilisateur tape deux fois sur la touche Tab pour rendre le lien actif, la redirection vers la page cible se faisant ensuite avec la touche Entrée :

```
<p>Les standards<a href="page2.html" tabindex="2">W3C</a></p>
```

- `charset` : il contient le code d'un jeu de caractères du type ISO-8859-1 (Latin 1) pour indiquer celui qui est utilisé dans le document cible. Nous ne l'utiliserons que si la cible est écrite par exemple dans un alphabet différent de celui de la page d'origine du lien.

- `type` : il indique si besoin est le type MIME du document cible. Nous l'utilisons uniquement si la cible n'est pas un document XHTML comme une image, un fichier son, vidéo ou un document PDF ou ZIP. Son utilisation permet de fournir au navigateur une information sur la manière d'afficher la cible, ce qui accélère son apparition. Nous pouvons écrire par exemple :

```
<li><a href="mapomme.tif" type="image/tiff">Ma photo</a></li>
```

- hreflang : il précise le code de langue du document cible s'il est différent de celui du document initial tel qu'il est indiqué par l'attribut xml:lang précisé par exemple dans l'élément <body>
- rel : il donne la relation de dépendance existant entre le document d'origine et le document cible. Cet attribut a déjà été utilisé dans l'élément <link>, et il peut prendre par exemple les valeurs index, next ou prev.
- rev : cet attribut indique la relation inverse du précédent, et donc le rapport existant entre le document cible et celui qui l'a appelé. Il prend le même type de valeurs que rel .
- shape: par défaut, toute la zone occupée à l'écran par le contenu de l'élément <a> est sensible au clic de l'utilisateur pour provoquer l'affichage de la cible. Cette situation peut être modifiée pour qu'une partie seulement du contenu soit sensible au clic. Cette possibilité est principalement utilisée quand le contenu est une image. L'attribut shape permet de définir la forme de cette zone. Il peut prendre l'une des valeurs suivantes :
  - rect pour un rectangle, c'est la valeur par défaut.
  - circle pour un disque.
  - poly pour un polygone.
  - default pour définir une adresse cible en cas de clic sur un emplacement situé en dehors des zones définies par les formes précédentes.

Il faut ensuite définir les coordonnées des points clés permettant de dessiner la forme choisie, à l'aide de l'attribut coords

- coords: il contient une suite de nombres qui sont les coordonnées de ces points clés. L'origine des coordonnées est l'angle supérieur gauche de la zone de contenu de l'élément <a> Pour chaque forme, il faut définir les coordonnées suivantes :
  - si shape="rect", nous devons donner dans cet ordre les coordonnées des sommets supérieur gauche et inférieur droit du rectangle. Par exemple :

```
<p><a href="meteo.fr" shape="rect" coords="50,20,100,60"> <img src="carte.gif"
alt="rectangle à cliquer" />Choix de la région</a></p>
```

  - si shape="circle", nous devons donner dans cet ordre les coordonnées du centre suivies du rayon. Par exemple :

```
<p><a href="meteo.fr" shape="circle" coords="100,150,50"> <img src="carte.gif"
alt="cercle à cliquer" />Choix de la région</a></p>
```

  - si shape="poly", nous devons donner les coordonnées de tous les sommets du polygone suivies des coordonnées du point de départ pour refermer la figure.

```
<p><a href="meteo.fr" shape="poly" coords="50,100,150,100,80,200,50,100">
<img src="carte.gif" alt="Triangle à cliquer" />Choix de la région</a></p>
```

Enfin, comme les autres éléments visuels pouvant recevoir le focus, l'élément <a> possède les attributs de gestion d'événements onfocus quand son contenu reçoit le focus (donc par défaut encadré par une bordure en pointillés), et onblur quand il perd le focus. Nous pouvons gérer ces événements en écrivant par exemple le code suivant :

```
<p>Le site du <a href="w3.org" onfocus="alert('Ne manquez pas le site du W3C')">
➥W3C</a></p>
```

Signalons enfin que l'utilisation des gestionnaires d'événements communs onmouseover et onmouseout peut se révéler intéressante pour les liens en permettant la création d'effets graphiques attrayants. En les associant à la méthode getElementById(), nous pouvons modifier le style du contenu de l'élément <a> quand le curseur survole ou quitte sa zone de contenu. En écrivant le code ci-après :

```
<p>Le site du <a id="lien1" href="http://w3.org"
➥ onmouseover="getElementById('lien1').style.color='red'"
➥ onmouseout="getElementById('lien1').style.color='black'">W3C</a></p>
```

le texte du lien devient rouge quand le curseur le survole, et noir quand il le quitte.

L'exemple 5-1 réalise la mise en œuvre des différents types de liens que nous venons d'envisager. Le premier crée un lien vers le site du W3C et utilise l'attribut onfocus pour afficher une boîte de dialogue (repère ³). Le deuxième lien définit les attributs accesskey, tabindex et title pour augmenter l'accessibilité (repère ·). Le troisième lien utilise les attributs onmouseover et onmouseout pour modifier le style du lien (repère »). Le suivant est effectué vers une image de type JPG qui s'affichera dans une nouvelle page (repère ¿). Le dernier est effectué vers un document au format texte brut qui s'affiche tel quel dans une nouvelle page (repère ´). Notons que, dans ces deux derniers cas, le contenu cible est affiché dans le navigateur mais il ne s'agit pas d'un document XHTML. Cela peut être vérifié en tentant d'afficher le code source de la page cible.

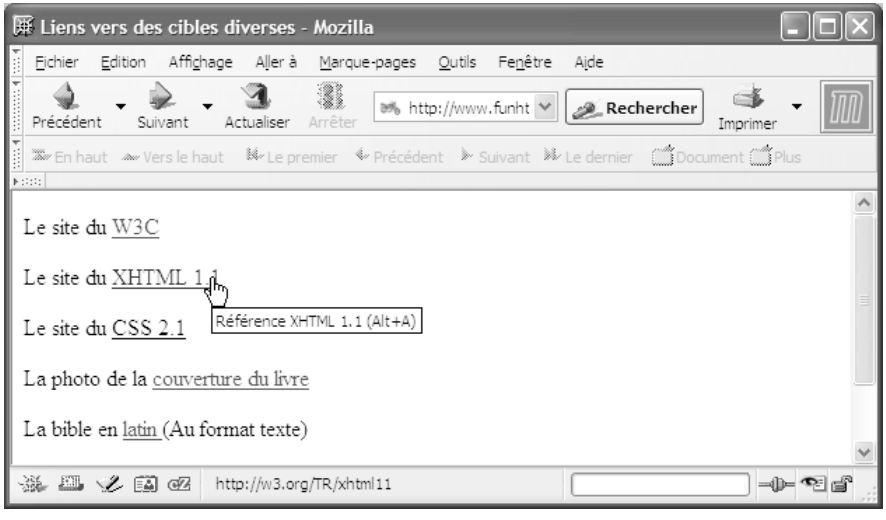
Exemple 5-1. Divers types de liens courants

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
➥ "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Liens vers des cibles diverses</title>
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <!-- Documents  XHTML -->
```

```
<p>Le site du <a id="lien1" href="http://w3.org"
  onfocus="alert('Ne manquez pas le site du W3C')">W3C</a></p>
<p>Le site du <a id="lien2" href="http://w3.org/TR/xhtml11" accesskey="A"
  tabindex="2" title="R&eacute;f&eacute;rence XHTML 1.1 (Alt+A)">
  XHTML 1.1 </a></p>
<p>Le site du <a id="lien3" href="http://www.w3.org/TR/CSS21/"
  onmouseover="getElementById('lien3').style.color='red'"
  onmouseout="getElementById('lien3').style.color='black'">CSS 2.1</a></p>
<!-- Image JPEG -->
<p>La photo de la <a id="lien4"
  href="http://www.funhtml.com/php5/couverture_PHP5.jpg" accesskey="E"
  tabindex="5" title="Couverture du livre (Alt+E)">couverture du livre</a></p>
<!-- DocumentTexte-->
<p>La bible en <a id="lien5" href="latin.txt" accesskey="G" tabindex="5"
  title="Couverture du livre (Alt+G)"> latin </a>(Au format texte)</p>
</body>
</html>
```

La figure 5-1 montre le résultat obtenu dans la page.

Figure 5-1
Différents types de liens



Les liens vers des documents particuliers

Quand leur attribut `href` pointe vers certains documents de types particuliers, les liens entraînent le démarrage de plug-ins spécialisés, propres au navigateur, ou de logiciels extérieurs qui lisent ces documents. L'exemple 5-2 contient de nombreux liens vers des documents très variés. La lecture de chacun d'eux fait appel à des modules ou des applications externes en général non installés par défaut sur les ordinateurs.

Exemple 5-2. Les liens vers des documents particuliers

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  ➥ "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Liens vers des documents multimédia</title>
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <!-- Document PDF -->
  <p>La documentation CSS 2.1 au format
   ➥ <a id="lien4" href="http://www.w3.org/TR/CSS21/css2.pdf" accesskey="A"
   ➥ tabindex="3" type="application/pdf" title="R&eacute;f&eacute;rence CSS 2.1
   ➥ (Alt+A)">PDF</a></p>
  <!-- Document Word -->
  <p>La bible en <a id="lien7" href="latin.doc" accesskey="B" tabindex="7"
   ➥ type="application/msword" title="Bible en latin (Alt+B)"> latin </a>
   ➥ (Au format Word)</p>
 <!-- Flash -->
  <p>Lire une animation <a id="lien8" href="lettres.swf" accesskey="D" tabindex="8"
   ➥ title="Animation Flash (Alt+D)"> Flash</a></p>
  <!-- Vidéo MPEG -->
  <p>Lire une Vidéo <a id="lien9" href="romy.mpg" accesskey="G" tabindex="9"
   ➥ type="video/mpeg" title="Romy la Star (Alt+G)"> MPEG</a></p>
  <!-- MP3 -->
  <p>Lire un fichier son au format<a id="lien10" href="respire.mp3" accesskey="H"
   ➥ tabindex="10" title="Mickey 3D: Respire (Alt+H)"> MP3</a></p>
 <!-- MIDI -->
  <p>Lire un fichier son au format<a id="lien11" href="breeze.mid" accesskey="J"
   ➥ tabindex="10" type="application/" title="Son midi (Alt+J)"> MIDI</a></p>
 <!-- Image TIFF -->
  <p>Lire une image au format<a id="lien12" href="romy.tif" accesskey="R"
   ➥ tabindex="10" type="image/tiff" title="Romy (Alt+R)"> TIFF</a></p>
<!-- Son WAV -->
  <p>Lire un son au format<a id="lien12" href="cigales.wav" accesskey="S"
   ➥ tabindex="10" type="audio/x-wav" title="Le chant des cigales (Alt+S)"> WAV</a>
   ➥ </p>
 </body>
</html>
```

Le premier lien est effectué vers un document PDF qui entraîne le démarrage du module spécialisé Acrobat Reader (repère ³). Le chargement de ce module (qui prend un temps assez long dans sa version 6, auquel il faut ajouter ensuite le temps de téléchargement du

fichier PDF lui-même) permet au navigateur de lire et d'afficher le document. Une nouvelle barre d'outils apparaît alors dans le navigateur. Elle permet de naviguer aisément dans le document. La figure 5-2 montre l'aspect de la fenêtre obtenue.

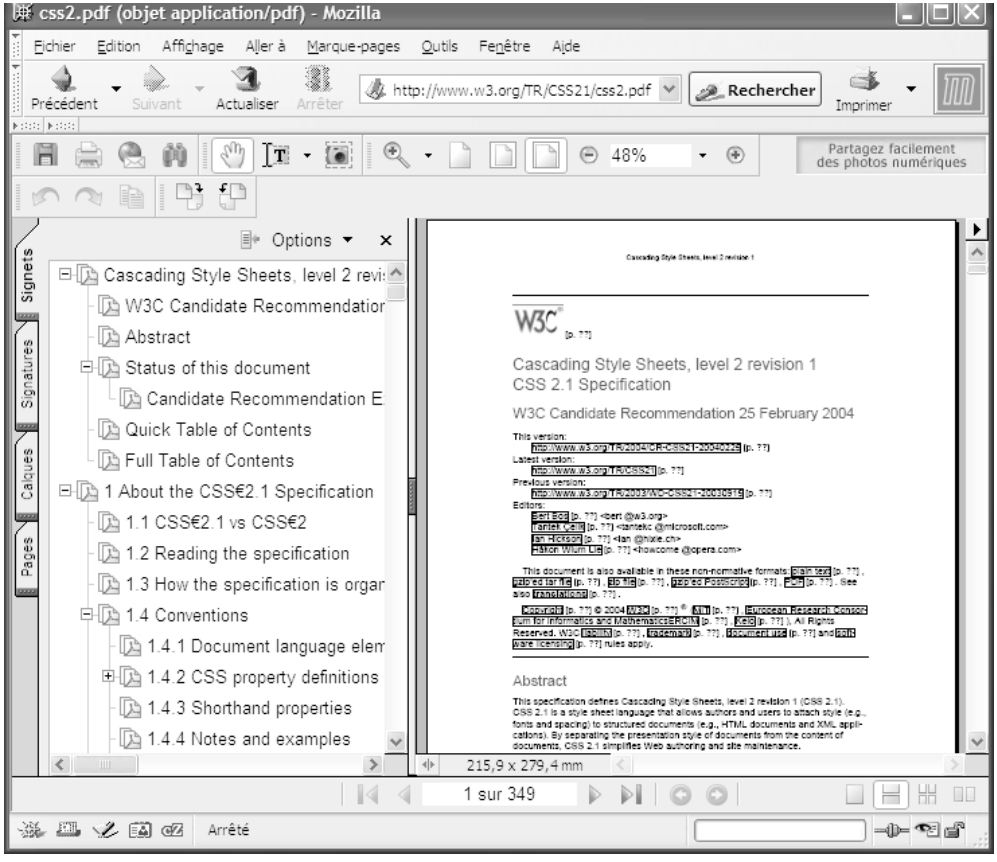


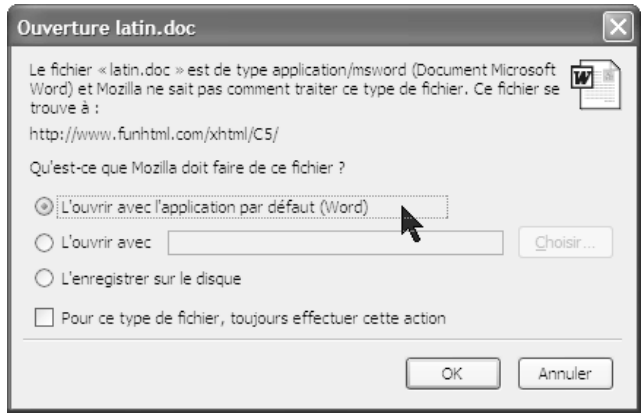
Figure 5-2

Visualisation d'un document PDF

Le deuxième lien est effectué vers un document Word (repère ·). L'activation de ce lien provoque l'apparition d'une fenêtre semblable à celle de la figure 5-3, qui demande au visiteur de choisir le programme à utiliser pour ouvrir le document, en suggérant une application. Si le visiteur clique sur la case à cocher de la dernière ligne, lors de son ouverture, le navigateur ouvre toujours la même application pour les fichiers Word sans afficher cette fenêtre de choix.

Le troisième lien dirige vers une animation Flash (repère ») qui est affichée dans une nouvelle page vide. Il faut cependant que le navigateur client soit muni du module de

Figure 5-3
Visualisation d'un document Word



lecture adéquat. Il est conseillé en règle générale de proposer un lien vers le site de téléchargement des plug-ins Flash (http://sdc.shockwave.com/shockwave/download/) pour ceux qui n'en disposeraient pas encore.

Les cinq liens suivants pointent vers des documents multimédias vidéo et sonores (repères ¿ , ´ , ² , ¶ et º). Leur activation va déclencher le démarrage d'une application externe comme le lecteur Windows Media ou Quick Time selon la configuration du poste client. La figure 5-4 montre la fenêtre de l'application Windows Media ouverte d'un clic sur le lien MP3.

Figure 5-4
Le lecteur Windows Media

Les liens ciblés : les ancres

Quand le contenu d'une page est volumineux, l'utilisateur ne peut pas en avoir une vision globale. Il est alors souhaitable de lui proposer une table des matières ou un menu composé de liens internes vers les différentes sections de la page. Il pourra ainsi accéder directement au point de son choix sans faire défiler la page. De même, si le lien est externe, il est possible d'accéder en un point particulier de la page cible. Chaque point cible de la page doit être signalé par un lien particulier, appelé ancre nommée, lequel est créé à l'aide de l'élément `<a>` muni simplement d'un attribut `id` dont la valeur doit être unique dans la page. Notons que l'ancre peut aussi se trouver dans un autre document. L'attribut `href` ne doit pas nécessairement être utilisé dans une ancre car celle-ci peut n'être que la cible du lien. Nous pouvons par exemple écrire le code suivant pour le contenu d'une page :

```
<p><a id="para1">Paragraphe 1</a>Un document XHTML doit avoir la structure…</p>
<p><a id="para2">Paragraphe 2</a>Une feuille de style CSS comporte…</p>
```

Chaque paragraphe devient alors la cible d'un lien. Nous devons ensuite créer le lien d'origine pour lequel un clic permet l'affichage du paragraphe cible en haut de la fenêtre du navigateur. L'élément `<a>` origine du lien doit contenir un attribut `href` dont la valeur est celle de l'attribut `id` de la cible, précédé du caractère dièse (#). Nous disposons par exemple du code suivant pour créer le lien vers le premier paragraphe :

```
<p><a href="#para1">Vers le paragraphe 1</a></p>
```

Si l'ancre est située dans un autre document XHTML, l'attribut `href` contient l'adresse relative ou absolue du document, suivie du caractère dièse, puis de l'identifiant de l'ancre. Pour une adresse relative, nous pouvons écrire :

```
<p><a href="page2.html#para1">Paragraphe 1</a></p>
```

Et, pour une adresse absolue, le code suivant :

```
<p><a href="http://www.funhtml.com/page2.html#para1">Paragraphe 1</a></p>
```

Pour créer un menu sous forme de liste, situé en haut de la page, et qui permet d'accéder directement à chaque partie de la page, nous pouvons écrire le code suivant :

```
<ul>
 <li><a href="#sujet1"> XHTML</a></li>
 <li><a href="#sujet2"> CSS</a></li>
 <li><a href="#sujet3"> JavaScript</a></li>
</ul>
```

Pour permettre des allers et retours rapides entre les liens origines et les ancres cibles, il faut utiliser conjointement les attributs `href` et `id` dans chacun des éléments `<a>` En reprenant l'exemple de notre menu, nous aurions le code de l'exemple 5-2 qui, à partir d'un menu contenant trois liens (repères ³ , · et »), permet l'accès aux différentes sections du texte contenu dans des éléments `<div>` (repères ¿ , ² et º). Avec les ancres qui possèdent un attribut `href` (repères ´ , ¶ et ¾), on peut revenir au menu à partir des différents paragraphes.

Exemple 5-3. Système de navigation dans une page unique

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les ancres</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <h1>Vos sujets préférés</h1>
  <ul>
   <li><a href="#sujet1" id="menu1"> XHTML</a></li>
   <li><a href="#sujet2" id="menu2"> CSS</a></li>
   <li><a href="#sujet3" id="menu3"> JavaScript</a></li>
  </ul>
<hr />
<!-- Première section -->
 <div>
<h1>XHTML</h1><a  id="sujet1" href="#menu1">Retour au vers le menu</a><br />
Quand le contenu d'une page est assez long, l'utilisateur ne peut pas en avoir une
➥ vision globale. Il est alors possible de lui proposer une table des matières ou
➥ un menu composé de liens vers les différentes sections de la page. Il pourra
➥ alors accéder directement au point qui lui convient sans avoir à faire défiler
➥ la page… </div><br /><hr />
<!-- Deuxième section -->
<div>
<h1>CSS</h1><a  id="sujet2" href="#menu2">Retour au vers le menu</a><br />
Quand le contenu d'une page est assez long, l'utilisateur ne peut pas en avoir une
➥ vision globale. Il est alors possible de lui proposer une table des matières ou
➥ un menu composé de liens vers les différentes sections de la page. Il pourra
➥ alors accéder directement au point qui lui convient sans avoir à faire défiler
➥ la page…</div><br /><hr />
<!-- Troisième section -->
<div>
<h1>JavaScript</h1><a  id="sujet3" href="#menu3">Retour au vers le menu</a><br />
Quand le contenu d'une page est assez long, l'utilisateur ne peut pas en avoir une
➥ vision globale. Il est alors possible de lui proposer une table des matières ou
➥ un menu composé de liens vers les différentes sections de la page. Il pourra
➥ alors accéder directement au point qui lui convient sans avoir à faire défiler
➥ la page… </div><br /><hr />
</body>
</html>
```

La figure 5-5 donne l'aspect de la page obtenue avec des paragraphes courts pour alléger le code, mais ce système convient bien évidemment à des textes longs.

Figure 5-5

Navigation dans une page unique

Chacun des liens de cette page est donc l'origine d'un lien interne vers une ancre nommée – par exemple le lien nommé « XHTML » (repère ³) conduit au premier paragraphe identifié par l'attribut id de valeur sujet1 (repère ´) – mais, réciproquement, le lien nommé « sujet1 » (repère ´) permet de remonter au menu « XHTML » identifié par l'attribut id de valeur menu1 (repère ³), et donc de le réafficher si la page est trop longue. De même, le lien nommé « CSS » du menu (repère ·) conduit au deuxième paragraphe (repère ¶) et le lien nommé « JavaScript » (repère ») mène au troisième paragraphe (repère ¾). Réciproquement, les liens incorporés dans chaque division permettent de remonter au menu en tête de page.

Emplacement des attributs id

Pour permettre le retour à l'ensemble du menu, court ici, il est également possible de supprimer les attributs id de chacun des liens de la liste et de définir un seul attribut pour cet élément <ul> avec la valeur menu1. Tous les liens situés dans les différents paragraphes ont alors un attribut href ayant la valeur #menu1. Ils conduisent alors tous au même point de la page.

L'exemple 5-4 permet d'illustrer les possibilités offertes en créant un système de navigation complet entre trois pages du même site. La page d'accueil nommée index5-4.html affiche un menu permettant d'accéder directement aux différents chapitres des pages nommées xhtml.html et css.html. Chacune de ces pages comporte également le même menu, ce qui permet une navigation croisée entre toutes les pages sans utiliser le bouton Précédent du navigateur pour revenir à la page d'accueil, et également d'atteindre n'importe quel chapitre de n'importe quelle page.

Exemple 5-4. Un système de navigation entre plusieurs pages

Le fichier index5-4.html

La page d'accueil contient deux titres de niveau 1 (repères ³ et ´) qui contiennent chacun trois liens (repères · , » et ¿) vers les chapitres 1, 2 et 3 situés dans les pages xhtml.html et css.html (repères ² , ¶ et º). Chacun de ces chapitres est contenu dans un paragraphe des pages cibles.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html  xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Navigation entre plusieurs pages</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <h1 id="menuxhtml">XHTML
   <a  href="xhtml.html#chap1"> Chapitre 1 </a> ** ·
   <a href="xhtml.html#chap2"> Chapitre 2 </a> ** »
   <a href="xhtml.html#chap3"> Chapitre 3 </a>  ¿
  </h1><hr />
  <h1 id="menucss">CSS´2
   <a  href="css.html#chap1"> Chapitre 1 </a> ** ²
   <a  href="css.html#chap2"> Chapitre 2 </a> ** ¶
   <a href="css.html#chap3"> Chapitre 3 </a>  º
  </h1><hr />
 </body>
</html>
```

Le fichier xhtml.html

Les pages xhtml.html et css.html ont une structure identique. On y retrouve le même menu que dans la page d'accueil (repères ³ et ·), auquel s'ajoute un lien vers le menu général de la page d'accueil (repère »). Chaque page contient ensuite trois paragraphes correspondant aux trois chapitres de chaque sujet (repères ¿ , ´ et ²).

Pour faciliter la navigation, chaque chapitre renferme par ailleurs un lien vers le menu de sa propre page (repères ¶ , º et ¾).

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html  xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>La page de XHTML 1.1</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <h1 id="menuxhtml">XHTML
   <a  href="xhtml.html#chap1"> Chapitre 1 </a> **
   <a href="xhtml.html#chap2"> Chapitre 2 </a> **
   <a href="xhtml.html#chap3"> Chapitre 3 </a>
  </h1><hr />
  <h1 id="menucss">CSS 2
   <a  href="css.html#chap1"> Chapitre 1 </a> **
   <a  href="css.html#chap2"> Chapitre 2 </a> **
   <a href="css.html#chap3"> Chapitre 3 </a>
  </h1><hr />
  <h1>
   <a  href="index5-3.html#menuxhtml"> Accueil </a>»
  </h1><hr />
<!-- Contenu de la page -->
  <h2>Chapitre 1: Introduction</h2>¿
   <p><a id="chap1" href="#menuxhtml">Menu¶</a><br />
    Pourquoi utiliser XHTML...In principio creavit Deus caelum et terram terra autem
      ➥ erat inanis et vacua et tenebrae super faciem abyssi et spiritus Dei
      ➥ ferebatur super aquas dixitque Deus fiat lux et facta est lux </p>
  <h2>Chapitre 2: La structure</h2>´
   <p><a id="chap2" href="#menuxhtml">Menuº</a><br />
    La structure d'un document XHTML...In principio creavit Deus caelum et terram
      ➥ terra autem erat inanis et vacua et tenebrae super faciem abyssi et spiritus
      ➥ Dei ferebatur super aquas dixitque Deus fiat lux et facta est lux
   </p>
  <h2>Chapitre 3: Le texte</h2>²
   <p><a id="chap3" href="#menuxhtml">Menu¾</a><br />
    Pour structurer le texte...In principio creavit Deus caelum et terram terra
      ➥  autem erat inanis et vacua et tenebrae super faciem abyssi et spiritus Dei
      ➥  ferebatur super aquas dixitque Deus fiat lux et facta est lux
   </p>
 </body>
</html>
```

Le fichier css.html

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>La page de XHTML 1.1</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <h1 id="menuxhtml">XHTML
   <a  href="xhtml.html#chap1"> Chapitre 1 </a> **
   <a href="xhtml.html#chap2"> Chapitre 2 </a> **
   <a href="xhtml.html#chap3"> Chapitre 3 </a>
  </h1><hr />
  <h1 id="menucss">CSS 2
   <a  href="css.html#chap1"> Chapitre 1 </a> **
   <a  href="css.html#chap2"> Chapitre 2 </a> **
   <a href="css.html#chap3"> Chapitre 3 </a>
  </h1><hr />
  <h1>
   <a  href="index5-3.html#menuxhtml"> Accueil </a>»
  </h1><hr />
<!-- Contenu de la page -->
  <h2>Chapitre 1: Introduction à CSS</h2>
  <p><a id="chap1" href="#menuxhtml">Menu</a><br />Pourquoi utiliser CSS 2...
     In principio creavit Deus caelum et terram terra autem erat inanis et vacua
     et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
     dixitque Deus fiat lux et facta est lux
  </p>
  <h2>Chapitre 2: Les sélecteurs CSS</h2>
  <p><a id="chap2" href="#menuxhtml">Menu</a><br />Les sélecteurs CSS...
     In principio creavit Deus caelum et terram terra autem erat inanis et vacua
     et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
     dixitque Deus fiat lux et facta est lux
  </p>
  <h2>Chapitre 3: Créer des styles</h2>
  <p><a id="chap3" href="#menuxhtml">Menu</a><br />Créer des styles...
     In principio creavit Deus caelum et terram terra autem erat inanis et vacua
     et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
     dixitque Deus fiat lux et facta est lux
  </p>
 </body>
</html>
```

Ce système complet permet donc en chaque point de n'importe quelle page d'accéder rapidement aux autres points des différentes pages du site. Les figures 5-6 et 5-7 montrent respectivement le menu de la page d'accueil et la page `xhtml.html` (la page `css.html` a le même aspect).

Figure 5-6
La page d'accueil du système de navigation

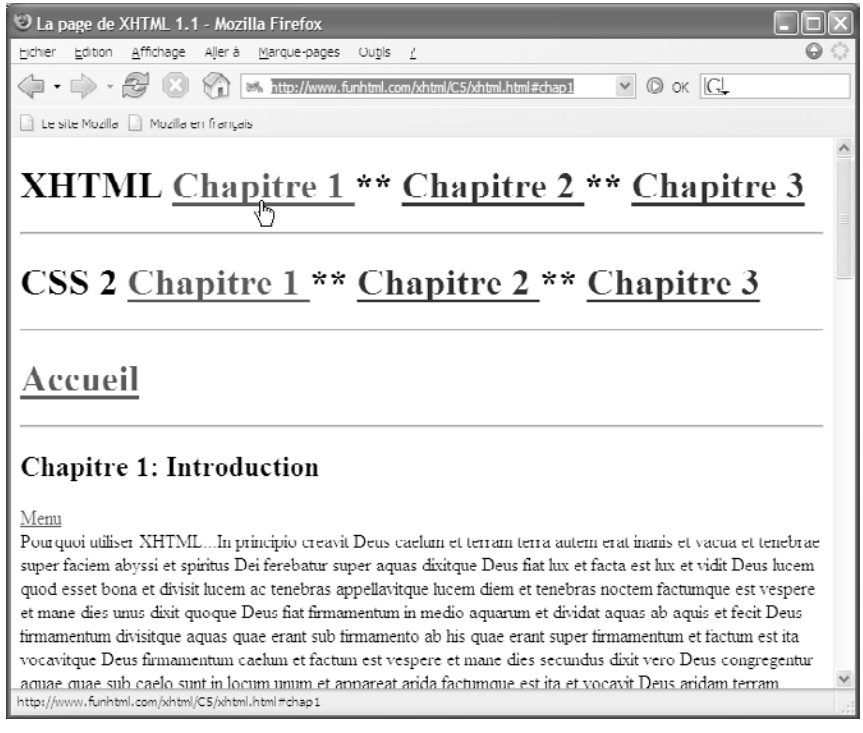


Figure 5-7
La page du XHTML

Liens à partir d'éléments graphiques

Lien à partir d'une image ou d'un bouton

Le contenu de l'élément <a> s'il est généralement composé de texte, peut également inclure des éléments graphiques, tels que des images, des boutons ou des contenus multimédias (voir le tableau 5-2).

On utilise les boutons comme origine de lien en incorporant un élément <button> entre les balises <a> et . L'attribut type de l'élément <button> doit avoir la valeur button, et le contenu de cet élément est le texte visible sur le bouton. C'est l'attribut href de <a> qui, comme précédemment, donne l'adresse du document cible du lien.

L'exemple 5-4 crée un ensemble de boutons utilisés comme déclencheur de liens vers des sites externes. Le code de l'exemple 5-4 fonctionne parfaitement dans les navigateurs les plus avancés (Mozilla, FireFox, Netscape) mais dans Internet Explorer (même avec la version 6) les boutons ne réagissent pas au clic. Pour pallier ce souci de conformité, il faut ajouter les gestionnaires d'événements onclick de l'élément <button> de la manière suivante, en appelant la propriété JavaScript window.location dont la valeur est l'adresse de la page vers laquelle le visiteur va être redirigé, c'est-à-dire la même valeur que celle de l'attribut href de l'élément <a> :

```
<button type="button" onclick="window.location = 'http://www.w3.org/TR/xhtml11'">
➥ XHTML </button>
```

Le premier lien (repère ³) utilise ce code. Les deux suivants ne sont dotés que de l'attribut href pour l'élément <a> (repères · et »). Nous pouvons même nous passer de l'élément <a> et utiliser seulement un bouton en guise de lien mais il faut créer la redirection avec du code JavaScript (repère ¿). Cette forme paraît plus simple mais la forme initiale me semble meilleure car plus conforme au standard XHTML et qu'il n'est pas positif de nous plier au manque de conformité d'un certain navigateur. Notons également que, pour plus simple qu'elle paraisse, elle nécessite l'emploi de JavaScript. Il n'est pas difficile d'imaginer ce qui peut se passer si l'utilisateur a choisi de bloquer l'exécution de JavaScript dans son navigateur : aucun lien de ce type ne fonctionne plus.

Exemple 5-5. Navigation utilisant des boutons

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
   <title> Liens et boutons </title>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  </head>
  <body>
  <h1>XHTML 1.1 et CSS 2.1</h1>
   <div>Visitez les sites
   <a id="lien1" href="http://www.w3.org/TR/xhtml11">
    <button type="button" onclick="window.location = '
     ➥ http://www.w3.org/TR/xhtml11'"> XHTML    </button>³
```

```
  </a>   
  <a id="lien2" href="http://www.w3.org/TR/CSS21/">
   <button type="button"> CSS 2.1 </button>
  </a>   
  <a id="lien3" href="http://www.w3.org">
   <button type="button" value="W3C"> Le site W3C </button>
  </a> »
  <button type="button" onclick = "window.location= ‘
     ➥ http://www.editions-eyrolles.com’"> Éditions Eyrolles </button>  ¿
 </div>
 </body>
</html>
```

Figure 5-8
Création de liens à partir de boutons

Dans son état courant, une image n'est pas sensible au clic pour provoquer un lien hypertexte, à moins de gérer l'attribut `onclick` comme nous venons de le faire pour l'élément `<button>`. Pour qu'elle devienne l'origine d'un lien, il suffit de l'inclure dans un élément `<a>` et de définir l'attribut `href` de cet élément de la même manière que pour un lien ordinaire. L'image sera alors entourée d'une bordure, généralement en bleu, pour signaler qu'il s'agit d'un lien. L'exemple 5-6 crée un système de navigation dont les origines des liens sont des images représentant des couvertures de livre de votre éditeur préféré. Chaque clic sur une image dirige le visiteur vers la page dans laquelle se trouve une image grand format de la même couverture et une description détaillée du livre. Chaque élément `<a>` (repères ³ , » , ´ et ¶) contient une image (repères · , ¿ , ² et º). L'attribut `href` de chaque élément `<a>` renferme l'adresse de la page cible du lien. L'élément `<a>` ne contenant aucun texte, c'est l'attribut `title` de chaque élément `<img />` qui peut donner une information sur la cible du lien quand le curseur survole l'image (ici, l'information est écourtée afin d'alléger le code).

Exemple 5-6. Création de lien à partir d'images

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <title> Liens et images </title>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
 </head>
 <body>
  <h1>De bonnes lectures recommandées</h1>
  <!-- Ensemble des liens créés à partir d'images miniatures -->
  <p>
   <a href="php5.html" > ³
     <img src="../images/couvphp5.gif" alt="PHP 5 Cours et exercices" width="100"
       ➥ height="130" title="PHP 5 Cours et exercices" /> ·
   </a>  
   <a href="objet.html">  »
     <img src="../images/couvobjet.gif" alt="L'orienté Objet" width="100"
       ➥ height="130" title="L'orienté objet" />      ¿
   </a>  
   <a href="reseaux.html"> ´
    <img src="../images/couvreseaux.gif" alt="Réseaux et télécoms"  width="100"
      ➥ height="130" title="Réseaux et télécoms"/>²
   </a>  
   <a href="uml.html"> ¶
    <img src="../images/couvuml2.gif" alt="UML 2 par la pratique"  width="100"
      ➥ height="130" title="UML 2 par la pratique"/>  º
   </a>
  </p>
  <hr />
 </body>
</html>
```

La figure 5-9 montre le résultat obtenu.

Figure 5-9
Création de liens à partir d'images

Comme nous l'avons fait pour un texte, il est possible de définir une ancre sur une image en définissant l'attribut `id` de l'élément `<a>`. Cette technique peut être utilisée par exemple pour diriger le visiteur vers une image particulière en grand format (soit l'ancre), située dans la même page ou dans une autre à partir d'une série d'images miniatures (les origines des liens).

Créer plusieurs liens sur la même image

Nous avons déjà vu au chapitre précédent comment créer des zones sensibles au clic sur une image pour déclencher des scripts quand l'utilisateur clique ou positionne simplement le curseur sur l'une des zones. En utilisant les mêmes éléments XHTML, nous pouvons définir des zones sensibles au clic sur une même image pour créer un lien différent pour chaque zone. L'ensemble des zones sensibles est défini dans une carte créée grâce à l'élément `<map>` qui contient autant d'éléments `<area />` que nous voulons créer de zones. La structure du code XHTML de création d'une carte de liens est la suivante :

```
<div>
 <map id="carte1">
  <area href="URL de la cible 1" alt="Cible 1" shape="mot-clé"
   ➥ coords="coordonnées" />
  <area href="URL de la cible 2" alt="Cible 2" shape="mot-clé"
   ➥ coords="coordonnées" />
  <!-- Suite des éléments <area> -->
 </map>
</div>
```

L'élément `<map>` doit avoir un attribut `id` qui contient l'identificateur de la carte. Précédé du caractère dièse (#), celui-ci peut être utilisé par plusieurs images différentes en étant affecté à l'attribut `usemap` de l'élément `<img />`. Chaque élément `<area />` contient l'URL du document cible du lien, définie dans son attribut `href` comme pour l'élément `<a>`. Comme pour une image, l'attribut `alt` de chaque élément `<area />` est obligatoire et fournit un texte de substitution si l'image ne peut pas être affichée, ou pour les navigateurs non graphiques. L'attribut `shape` définit la forme de la zone sensible au clic. Il prend les valeurs `rect`, `circle`, `poly` pour créer respectivement un rectangle, un disque un polygone, ou la valeur `default` qui permet de gérer tous les clics effectués en dehors des zones sensibles définies avec les éléments `<area />`. La définition des formes est faite en donnant les coordonnées en pixels des points clés de la forme dans l'attribut `coords`. L'origine des coordonnées est l'angle supérieur gauche de l'image.

- Si `shape="rect"`, l'attribut `coords` s'écrit `coords="Xhg,Yhg,Xbd,Ybd"`. Les valeurs `(Xhg,Yhg)` et `(Xbd,Ybd)` sont respectivement les coordonnées des sommets supérieur gauche et inférieur droit du rectangle.
- Si `shape="circle"`, l'attribut `coords` s'écrit `coords="Xc,Yc,R"`, `(Xc,Yc)` étant les coordonnées du centre et R le rayon du disque.
- Si `shape="poly"`, l'attribut `coords` contient la suite des coordonnées de chaque sommet et se termine par la répétition des coordonnées du premier point pour fermer le polygone.

L'utilisation de l'attribut `title` pour chaque élément `<area />` permet de définir un texte explicatif différent qui s'affiche quelques instants si le curseur reste sur la zone sensible.

Pour chaque élément `<area />`, il est utile de définir les attributs `accesskey` et `tabindex`, que nous avons déjà rencontrés, pour créer un raccourci clavier et un ordre de tabulation pour rendre chaque zone active. Comme un lien `<a>`, chaque zone sensible peut être définie comme une ancre. Dans ce cas, il faut lui donner un attribut `id`, omettre l'attribut `href` et facultativement préciser cette omission en définissant l'attribut `nohref` avec la valeur booléenne unique `nohref`. Comme tous les éléments pouvant recevoir le focus, l'élément `<area />` possède les attributs gestionnaires d'événements `onfocus` quand la zone reçoit le focus au moyen de la souris, d'un raccourci clavier ou d'une tabulation, et `onblur` quand elle le perd par les mêmes moyens. Il est alors possible de déclencher l'exécution d'un script JavaScript lorsque ces événements surviennent. Nous pouvons également déclencher des scripts pour créer des effets visuels avec les gestionnaires d'événements `onmouseover` et `onmouseout` quand le curseur survole ou quitte une zone sensible.

Dans l'exemple 5-7, nous créons une navigation à partir de l'image d'une carte de France pour laquelle nous définissons un ensemble de zones sensibles à l'aide des éléments `<map>` et `<area />`. La carte des zones est identifiée par l'attribut `id` de l'élément `<map>` sous le nom `regions` (repère ·). La première zone correspondant à la région parisienne est circulaire (repère »,), la deuxième pour la région Pays de la Loire est un polygone à quatre côtés (repère ¿) et la dernière pour la région Centre est un rectangle (repère ´). Le dernier élément `<area>` ayant l'attribut `shape` avec la valeur `default` permet de diriger les visiteurs vers la page d'accueil en cas de clic sur les autres parties de la carte (repère ²). La carte des zones sensibles est utilisée par l'image de la carte de France (repère ¶). À cet effet, l'identifiant de la carte est donc donné comme valeur de l'attribut `usemap` de l'élément `<img />`.

Exemple 5-7. Liens à partir d'une image

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">³
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
 <title>Liens et image sensible aux clics</title>
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
 <div>
  <h1>Les sites des régions</h1>
  <!-- Validé W3C XHTML 1.0 avec # dans usemap : Fonctionne dans
    ➥ Opera,Mozilla,FireFox,Explorer-->
  <map id="regions" name="regions">
   <area href="http://www.paris.fr" alt="Région parisienne"
     ➥ title="Région parisienne" shape="circle" coords="180,98,16" /> »
   <area href="http://www.nantes.fr" alt="Pays de Loire" title="Pays de Loire"
     ➥ shape="poly" coords="76,146,95,110,138,130,94,182,76,146"¿/>
```

```
  <area href="http://www.orleans.fr" alt="Région Centre" title="Région Centre"
    ➥ shape="rect" coords="142,118,188,180" />
  <area shape="default" href="http://www.funhtml.com" alt="Page d'accueil"
    ➥ title="La page d'accueil"/>
 </map>
 <img usemap="#regions" src="../images/france.gif" alt="Carte des régions"
  ➥ width="344" height="336" />¶
</div>
</body>
</html>
```

Vous n'avez pas manqué de remarquer que cet exemple n'utilise pas la DTD XHTML 1.1 mais la DTD XHTML 1.0 strict (repère ³). Nous y sommes contraint pour que la page soit validée par le W3C. En effet, son service de validation signale une erreur de conformité quand on utilise le caractère dièse (#) dans l'attribut `usemap` alors que le code de l'exemple est validé avec la DTD 1.0 strict, qui est théoriquement quasi similaire. Il s'agit probablement d'un bogue non éclairci à ce jour malgré de nombreux tests.

La figure 5-10 illustre le résultat obtenu, dans lequel on peut remarquer que l'utilisation de l'attribut `title` permet l'affichage d'un message pour chaque région.

Figure 5-10
Création de zones sensibles dans une image

Ouverture de la cible dans une nouvelle fenêtre

La création de liens externes comme ceux de l'exemple 5-7 présente l'inconvénient de provoquer la fermeture de la page de votre site afin d'en ouvrir une autre. Pour y remédier et laisser votre page visible, vous pouvez choisir d'afficher le document cible dans une nouvelle fenêtre en utilisant, comme nous l'avons déjà fait par ailleurs, la méthode JavaScript `window.open()` dont le premier paramètre est l'adresse URL de la cible, et dont les deux suivants sont respectivement la largeur et la hauteur de la fenêtre souhaitée, exprimés en pixels. Il faut alors l'associer à l'attribut `onclick` de chaque élément `<area />` pour obtenir l'effet désiré pour chaque lien. L'attribut `href` de cet élément n'est alors plus utile et doit être supprimé, faute de quoi la page cible s'ouvre dans deux fenêtres différentes, l'ancienne et une nouvelle.

Nous pouvons écrire par exemple le code suivant :

```
<map id="regions" name="regions">
  <area alt="Région parisienne" title="Région parisienne" shape="circle"
    ➥ coords="180,98,16" onclick="window.open('http://www.paris.fr/',800,600)" />
</map>
```

Dans ce cas, la page cible `www.paris.fr` sera visible dans une nouvelle fenêtre de 800 sur 600 pixels.

L'attribut target

Dans les DTD HTML 4 et en XHTML 1.0 Transitional, l'élément `<a>` possède un attribut `target` qui permet de désigner la page cible d'un lien. Il peut prendre les valeurs prédéfinies `_blank`, `_parent`, `_self` et `_top` pour afficher la cible, respectivement, dans une nouvelle fenêtre, la fenêtre parent de la fenêtre en cours, la fenêtre elle-même et la fenêtre de plus haut niveau. Les trois dernières valeurs sont surtout utilisées dans le cas de création de cadres (voir le chapitre 8, *Créer des cadres*).

Liens déclenchant une action

Lien déclenchant l'envoi d'un e-mail

Un site qui se dit à l'écoute de ses visiteurs doit leur permettre d'entrer en contact avec son webmestre afin qu'ils envoient leurs observations ou questions. Pour leur faciliter la tâche, il ne suffit pas d'indiquer une adresse e-mail dans chaque page. On doit créer un type de lien particulier provoquant, en un clic, l'ouverture automatique du logiciel de courrier préféré du visiteur avec comme destinataire l'adresse que l'auteur du site aura choisi dans son code. Pour réaliser cette opération, il suffit que l'attribut `href` du lien soit composé du nom de protocole `mailto:` suivi de l'adresse e-mail du contact.

En associant les éléments `<address>` et `<a>` nous pouvons écrire par exemple le code suivant :

```
<address>
 <a href="mailto:webmaster@funhtml.com"> Contactez nous! </a>
</address>
```

L'élément <address> peut bien sûr être remplacé par un autre élément de type bloc, comme <p> ou . Il est de plus possible de faciliter la tâche du visiteur en remplissant à sa place des champs dans son logiciel de courrier. Pour définir le sujet du courrier, dans l'attribut href, il faut faire suivre l'adresse e-mail par un point d'interrogation (?) puis du mot-clé subject= et du texte choisi pour l'objet du message sans guillemets. Nous pouvons définir par exemple l'objet de l'e-mail de la manière suivante :

```
<p>
Demande de <a href="mailto:xhtml@funhtml.com?subject=Demande de renseignements">
➥ documentation</a>
</p>
```

Nous pouvons encore ajouter un ou plusieurs destinataires en copie visible. Dans ce cas, le destinataire d'une copie de l'e-mail sera informé qu'il ne reçoit qu'une copie d'un e-mail et aura connaissance du destinataire principal et des autres destinataires mis en copie. Il faut pour cela faire suivre l'adresse du destinataire principal du caractère ?, puis du code cc=, suivi de l'adresse du destinataire mis en copie. Pour ajouter plusieurs destinataires en copie, il faut séparer chaque adresse de la précédente par un caractère point-virgule (;). Pour ajouter un destinataire, nous pouvons écrire :

```
<p>
Réclamation de
<a href="mailto:xhtml@funhtml.com?cc=php5@funhtml.com"> commande non reçue</a>
</p>
```

Si nous voulons ajouter un destinataire sans que son adresse e-mail apparaisse, il faut l'ajouter en copie cachée en utilisant le code bcc à la place de cc. Là aussi, nous pouvons indiquer plusieurs adresses en les séparant par un point-virgule (;).

Toutes les options précédentes peuvent être utilisées simultanément en les séparant toutes les unes des autres par l'entité de caractère correspondant au caractère esperluette (&), soit l'entité &. Notez que le caractère & lui-même est interdit dans l'attribut href et qu'il faut le remplacer par l'entité correspondante.

L'exemple 5-8 crée une liste de liens permettant au visiteur de demander l'envoi d'une documentation sur différents types de services offerts par un site d'assurance. Il comporte trois liens contenus chacun dans un paragraphe (repères ³, · et »). Ces liens déclenchent l'envoi d'un e-mail à une adresse différente selon le choix effectué par le visiteur. Pour tous ces liens, nous avons défini un objet de message personnalisé différent (avec subject=), une adresse en copie simple et une adresse en copie cachée vers deux services de l'entreprise.

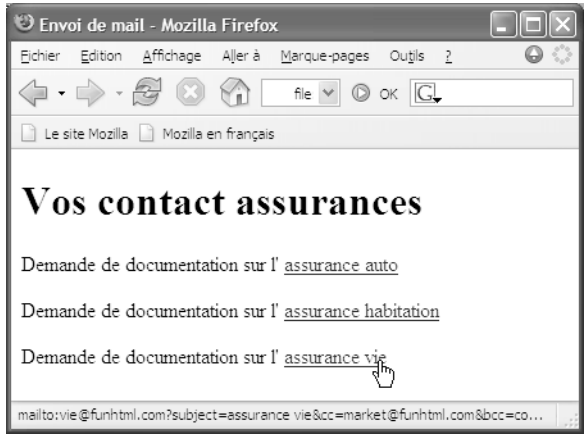
Exemple 5-8. Envoi d'e-mail à partir d'un lien

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
 <title>Envoi d'e-mail</title>
```

```
 <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 <style type="text/css" title="">
 </style>
</head>
<body>
 <h1>Vos contact assurances</h1>
 <p>Demande de documentation sur l'
  <a href="mailto:auto@funhtml.com?subject=assurance auto&amp;
    ➥ cc=market@funhtml.com&amp;bcc=commercial@funhtml.com">assurance auto</a>
 </p>
 <p>Demande de documentation sur l'
  <a href="mailto:habitation@funhtml.com?subject=assurance habitation&amp;
    ➥ cc=market@funhtml.com&amp;bcc=commercial@funhtml.com">assurance habitation<
    ➥ /a>
 </p>
 <p>Demande de documentation sur l'
  <a href="mailto:vie@funhtml.com?subject=assurance vie&amp;
    ➥ cc=market@funhtml.com&amp;bcc=commercial@funhtml.com">assurance vie</a>
 </p>
</body>
</html>
```

La figure 5-11 présente le résultat obtenu. Notons que le contenu de l'attribut `href` est affiché comme les autres types de liens dans la barre d'état du navigateur, quand le curseur survole le lien.

Figure 5-11
Création de liens vers des e-mails



Les liens utilisant le protocole `mailto:` peuvent, comme tous les autres, contenir une image. Un clic sur celle-ci provoque donc l'ouverture du logiciel de courrier. Cette possibilité peut être exploitée pour une demande de renseignements dans un site de petites annonces, par exemple après un clic sur la photographie de l'article qui intéresse un visiteur. La référence de l'article peut alors être insérée dans l'objet de l'e-mail, ainsi que l'adresse du destinataire.

Les liens déclenchant un script JavaScript

Un lien peut également être utilisé pour déclencher l'exécution d'un script JavaScript sans utiliser un gestionnaire d'événements tel que `onclick` comme nous l'avons vu précédemment. Pour réaliser cette opération, il faut que l'attribut `href` de l'élément `<a>` contienne le nom de protocole `javascript:`, suivi du code que l'on désire voir s'exécuter en cas de clic sur le lien. Ce code peut être l'appel d'une fonction qui doit alors être écrite dans un élément `<script>` situé soit dans l'en-tête de la page (l'élément `<head>`), soit dans le corps du document de cette même page, ou encore dans un fichier externe ayant l'extension `.js` incorporé par un élément `<link>` (voir le chapitre 2).

Pour créer ce type de lien, nous écrivons par exemple le modèle suivant :

```
<p>Cliquez pour <a href="javascript:code JavaScript">exécuter le script</a></p>
```

Si le code JavaScript est court, il peut être écrit directement à la suite du mot-clé `javascript:`. Mais s'il est plus long, il est conseillé de définir une fonction dans un élément `<script>` et de faire suivre le mot-clé de l'appel de cette fonction.

L'exemple 5-9 utilise ce type de lien pour proposer le déclenchement de plusieurs fonctions JavaScript définies dans l'en-tête du document (repère ³). Le premier lien permet d'afficher à la demande la date et l'heure complète dans une boîte de dialogue (repère ·) par l'appel de la fonction `ladate()` (repère »). Le deuxième permet de modifier la couleur de fond de la division contenant les liens (repère ¿) en appelant la fonction `fondrouge()` qui attribue un style à la division dont l'attribut `id` vaut `divis` (repère ´). Enfin, le troisième lien permet à l'utilisateur de modifier la taille de la police du texte de l'élément `<div>` dans lequel il se trouve (repère ²) en appelant la fonction `grandtexte()` (repère ¶).

Exemple 5-9. Création de liens JavaScript

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>³
  <title> Liens vers des scripts JavaScript </title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <script type="text/javascript">
  <!--
  function ladate()  »
  {
   jour= new Date;
   alert(jour);
  }
  function fondrouge()  ´
  {
   col="#F00";
   document.getElementById('divis').style.backgroundColor=col;
  }
  function grandtexte()  ¶
  {
```

```
  document.getElementById('divis').style.fontSize='20px';
 }
   -->
 </script>
</head>
<body id="page">
 <div id="divis">
  <h1>Lancer des scripts JAVASCRIPT</h1>
  <a href="javascript:ladate()">Afficher la date</a><br />
  <a href="javascript:fondrouge() ">Créer un fond rouge pour cette division
    </a><br /> ¿
  <a href="javascript:grandtexte() ">Afficher le texte avec des caractères
    de 20 Pixels</a><br /> ²
 </div>
</body>
</html>
```

La figure 5-12 montre l'état initial de la page et celui qu'elle a après l'activation des trois liens.

Figure 5-12
Création de liens JavaScript

Exercices

Exercice 1 : Peut-on inclure directement un élément `<a>` dans les éléments `<body>` `<address>` `<pre>` ou `<form>` ?

Exercice 2. Écrivez la phrase : « Le langage XHTML permet la création de pages structurées, CSS 2 des présentations variées. ». Créez des liens vers les sites http://www.funhtml.com/xhtml et http://www.w3.org/TR/CSS21/ pour les mots « XHTML » et « CSS 2 ».

Exercice 3. Déclenchez l'ouverture d'une boîte de dialogue JavaScript en cas de clic sur un lien textuel.

Exercice 4. Créez un lien vers un document PDF (par exemple : http://www.w3.org/TR/CSS21/css2.pdf) puis vers un fichier MP3. Indiquez le type du document cible en utilisant l'attribut correct.

Exercice 5. Créez plusieurs paragraphes dans une page. Chacun d'eux contient un titre et un texte quelconque assez long. Chaque titre doit avoir un identifiant et contenir une ancre. En haut de la page, ajoutez un titre `<h1>` contenant les noms des différents paragraphes, chacun étant inclus dans un lien. Gérez ensuite les attributs `href` pour permettre une navigation réciproque entre ce menu et chacun des paragraphes.

Exercice 6. À partir de l'exercice précédent, remplacez le titre qui sert de menu par des boutons. Un clic sur chaque bouton doit permettre la même navigation vers les paragraphes.

Exercice 7. Récupérez les logos des différentes chaînes de télévision sur leur site respectif. Créez une page incluant ces logos, puis des liens vers le site de chaque chaîne à partir de ces logos sans utiliser l'attribut `onclick`.

Exercice 8. Même question que dans l'exercice 7, mais sans utiliser l'élément `<a>`

Exercice 9. À partir de l'exercice 7, modifiez le code de façon à ce que chaque cible d'un lien s'ouvre dans une nouvelle fenêtre sans que la page d'origine ne se ferme.

Exercice 10. À partir d'une carte d'Europe (récupérable par exemple sur le site : http://europa.eu.int/abc/maps/index_fr.htm), créez des zones sensibles pour la France (en forme de cercle), l'Allemagne (en forme de rectangle) et la Grande-Bretagne (en forme de polygone), et dirigez le visiteur vers un site quelconque de chaque pays (par exemple, celui de amazon.com avec les extensions .fr, .de et .co.uk). Le clic en dehors de ces zones doit être géré, c'est-à-dire qu'il doit diriger le visiteur vers un site par défaut (par exemple, amazon.com).

Exercice 11. Réutilisez l'exemple précédent de façon à ce que chaque page cible s'affiche dans une nouvelle fenêtre de dimensions 640 par 480 pixels.

Exercice 12. Créez des liens à partir de la carte d'Europe précédente afin que chaque clic sur une zone déclenche l'envoi d'un e-mail à votre adresse. Chaque e-mail a pour objet le nom du pays concerné. Si vous possédez plusieurs adresses, mettez la deuxième en copie.

Exercice 13. Créez deux boutons dans une page. Le premier contient le texte « Fond rouge/Texte blanc », et le second « Fond noir/Texte jaune ». En vous inspirant de l'exemple 5-9, provoquez le changement des couleurs de fond et de texte en cas de clic sur les boutons.

6

Créer des tableaux

Dans une page XHTML, il est possible de réaliser une présentation d'informations de type textuelle ou graphique sous la forme de tableaux. De la même manière que dans un tableur Excel ou OpenOffice, un tableau créé en XHTML est composé de lignes et de colonnes. À l'intersection de ces dernières se trouve une cellule du tableau dont le contenu peut être constitué aussi bien d'éléments de type bloc (titres, paragraphes, divisions, listes), d'éléments en ligne tels que des images, ou encore d'éléments de formulaire. Nous verrons qu'à partir d'une grille de base, il est possible de fusionner plusieurs cellules voisines, ce qui permet de créer des tableaux asymétriques très variés. Les tableaux ont souvent été utilisés pour élaborer des mises en page de documents sur plusieurs colonnes en lieu et place des cadres (voir le chapitre 8). Même si nous étudions dans ce chapitre le procédé permettant de réaliser des présentations variées, il faut savoir que cette tendance est actuellement abandonnée au profit de l'utilisation des propriétés CSS de positionnement qui permettent de procéder aux mêmes types de réalisations d'une manière plus élégante et moins rigide. Nous retrouvons ici la tendance permanente en XHTML de séparation du contenu et de la présentation. Les tableaux n'en continuent pas moins d'être des éléments essentiels d'une page XHTML pour la présentation des données.

La structure générale d'un tableau

L'élément essentiel dans la création de tableaux est `<table>`, car il s'agit du conteneur renfermant tous les autres éléments de tableau. En lui-même, il ne fournit donc aucun résultat visuel et ne prend d'importance que par ses éléments enfants. Comme il est de type bloc, il peut être inclut directement dans le corps du document `<body>`. Il peut aussi

être inclut dans des éléments XHTML divers dont la liste est présentée dans le tableau 6-1.

Tableau 6-1. Les éléments parents de l'élément <table>

blockquote, body, button, dd, del, div, fieldset, form, ins, li, map, noscript, object, td, th

Nous devons créer sa structure ligne par ligne. Chaque ligne est déclarée par un élément <tr> (pour *table row*). La création de chaque cellule d'une ligne est déclarée par un élément <td>(pour *table data*) pour les cellules standards du tableau ou <th>(pour *table head*) pour les cellules qui jouent le rôle d'en-tête de colonne ou de ligne. Ces éléments sont les seuls enfants autorisés pour l'élément <tr>. La priorité est en effet donnée aux lignes sur les colonnes, priorité que nous retrouverons dans la définition des styles des tableaux au chapitre 14, *Le style des tableaux*. Pour un tableau régulier, nous obtenons donc une structure du code XHTML dans laquelle il y a autant d'éléments <tr> inclus dans <table>que de lignes et d'éléments <td>ou <th>inclus dans les différents éléments <tr> que de colonnes dans chaque ligne. La diversité du contenu d'un tableau n'apparaît que dans le contenu des éléments <td>et <th>dont le tableau 6-2 donne la liste des éléments enfants.

Tableau 6-2. Les éléments enfants des éléments <td> et <th>

Texte, a, abbr, acronym, address, b, bdo, big, blockquote, br, button, cite, code, del, dfn, div, dl, em, fieldset, form, h1, h2, h3, h4, h5, h6, hr, i, img, input, ins, kbd, label, map, noscript, object, ol, p, pre, q, samp, script, select, small, span, strong, sub, sup, table, textarea, tt, ul, var

La structure d'un tableau élémentaire régulier (contenant autant de cellules dans chaque ligne) est donc celle présentée dans l'exemple 6-1. Faute de précision supplémentaire, la largeur du tableau est fonction du contenu des cellules le composant. Dans ces conditions, si le contenu d'une cellule est important, elle s'élargit automatiquement jusqu'à ce que le tableau occupe toute la largeur de la page. Si toutes les cellules ont ce type de contenu, le partage de la largeur est fait équitablement entre elles et leur contenu occupe plusieurs lignes. Pour terminer la présentation d'un tableau, nous pouvons lui attribuer un titre général qui doit être contenu dans l'élément <caption> lui-même inclus dans <table> Cet élément doit être le premier à apparaître dans <table> Son contenu, qui apparaît par défaut au-dessus du tableau, peut être constitué d'éléments divers dont la liste figure dans le tableau 6-3. Cette position peut être modifiée en recourant à un style CSS (voir la propriété caption-side).

Tableau 6-3. Les éléments enfants de l'élément <caption>

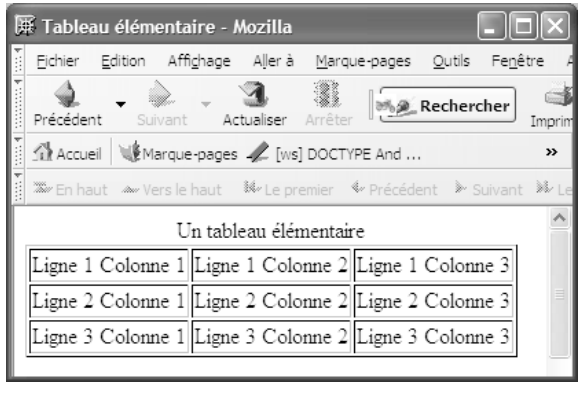
Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var

Exemple 6-1. Un tableau élémentaire

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Tableau élémentaire</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body >
  <table border="1">
   <caption>Un tableau &#233;l&#233;mentaire</caption>
   <tr>
     <td> Ligne 1 Colonne 1 </td><td> Ligne 1 Colonne 2 </td><td> Ligne 1 Colonne 3
        ➥ </td>
   </tr>
   <tr>
     <td> Ligne 2 Colonne 1 </td><td> Ligne 2 Colonne 2 </td><td> Ligne 2 Colonne 3
        ➥ </td>
   </tr>
   <tr>
     <td> Ligne 3 Colonne 1 </td><td> Ligne 3 Colonne 2 </td><td> Ligne 3 Colonne 3
        ➥ </td>
   </tr>
  </table>
 </body>
</html>
```

La figure 6-1 présente le résultat obtenu pour ce code.

Figure 6-1
Un tableau élémentaire



Les attributs de l'élément <table>

L'élément <table> possède l'ensemble des attributs communs, les plus couramment utilisés étant title, qui permet d'expliciter son contenu général, et class pour lui appliquer des styles, et bien sûr l'attribut id pour l'identifier de manière unique. Il possède également les attributs suivants, permettant d'affiner son organisation ou sa présentation par défaut :

- border="N" : il définit la largeur des bordures externes et internes qui délimitent le tableau et les cellules. Sa valeur s'exprime exclusivement en nombre de pixels et la valeur 0 est admise pour cacher ses bordures. L'unité px, pour pixel, ne doit pas être indiquée car elle est implicite. Nous pourrons par la suite affiner la taille des bordures du tableau, les lignes et des cellules à l'aide de styles CSS (voir le chapitre 14).
- width="N px | N%" : il définit la largeur totale du tableau dans la page. La possibilité de définir une largeur relative en pourcentage est très pratique pour adapter le tableau à l'écran du visiteur.
- cellpadding="N px | N%" : il définit la largeur de l'espacement entre le contenu d'une cellule et sa bordure.
- cellspacing="N px | N%" : il définit l'espacement entre les bordures de chaque cellule. Pour que cet attribut soit pris en compte, il faut que l'attribut border ait été par ailleurs défini avec une valeur non nulle. Si on omet de définir une valeur pour l'attribut cellspacing sa valeur est la même que celle de border.
- frame : il permet de diversifier l'affichage des bordures externes du tableau en n'affichant qu'une partie d'entre elles. Il peut prendre les valeurs suivantes :
  - void : supprime toutes les bordures extérieures ;
  - above : ne laisse subsister que la bordure supérieure ;
  - below : ne laisse que la bordure inférieure ;
  - lhs : ne laisse que la bordure gauche ;
  - rhs : ne laisse que la bordure droite ;
  - hsides : ne laisse que les bordures horizontales haute et basse ;
  - vsides : ne laisse que les bordures verticales gauche et droite ;
  - box ou border : ne laisse que les quatre bordures externes.
- rules : il définit l'affichage des bordures internes du tableau, situées entre les cellules. Il prend les valeurs suivantes :
  - none : supprime toutes les bordures internes ;
  - rows : ne laisse que les bordures horizontales ;
  - cols : ne laisse que les bordures verticales ;
  - all : affiche toutes les bordures. C'est le comportement par défaut ;

  - groups : n'affiche que les bordures délimitant les groupes de lignes ou de colonnes définis par les éléments <col /> et <colgroup>, <tbody>, <thead> et <tfoot>, dont nous allons voir les rôles respectifs ci-après.

- summary : il contient un texte décrivant plus complètement le contenu du tableau que l'élément title. De plus, ce texte n'est pas utilisé dans les navigateurs visuels habituels, mais uniquement dans les navigateurs oraux destinés aux non-voyants.

L'exemple 6-2 reprend le contenu de l'exemple 6-1 en ajoutant à l'élément <table> une partie des attributs que nous venons de décrire. Nous y définissons une bordure de 8 pixels (repère ³), un espacement entre les cellules de 5 pixels (repère ·), un espacement entre le contenu des cellules et leurs bordures de 20 pixels (repère »). La largeur totale du tableau est égale à 70 % de celle de son conteneur (repère ¿). Ici, il s'agit de l'élément parent <body> le pourcentage est donc calculé par rapport à la largeur de la fenêtre du navigateur, mais ce n'est pas nécessairement toujours le cas, si par exemple le tableau est inclut dans une division dont la largeur est elle-même fixée à une valeur inférieure à celle de la fenêtre. Enfin, l'attribut frame permet de n'afficher que les bordures horizontales du tableau (repère ´).

Exemple 6-2. Les attributs de l'élément <table>

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Tableau élémentaire utilisant les attributs de table</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body >
  <caption><strong>Un tableau &#233;l&#233;mentaire</strong></caption>
  <table border="8" ³ cellspacing="5px" · cellpadding="20" » width="70%"¿
   ➥ frame="hsides" ´ >
   <tr>
     <td> Ligne 1 Colonne 1 </td><td> Ligne 1 Colonne 2 </td><td> Ligne 1 Colonne 3
        ➥ </td>
   </tr>
   <tr>
     <td> Ligne 2 Colonne 1 </td><td> Ligne 2 Colonne 2 </td><td> Ligne 2 Colonne 3
        ➥ </td>
   </tr>
   <tr>
     <td> Ligne 3 Colonne 1 </td><td> Ligne 3 Colonne 2 </td><td> Ligne 3 Colonne 3
        ➥ </td>
   </tr>
  </table>
 </body>
</html>
```

La figure 6-2 montre les notables différences d'aspect obtenues en utilisant ces attributs.

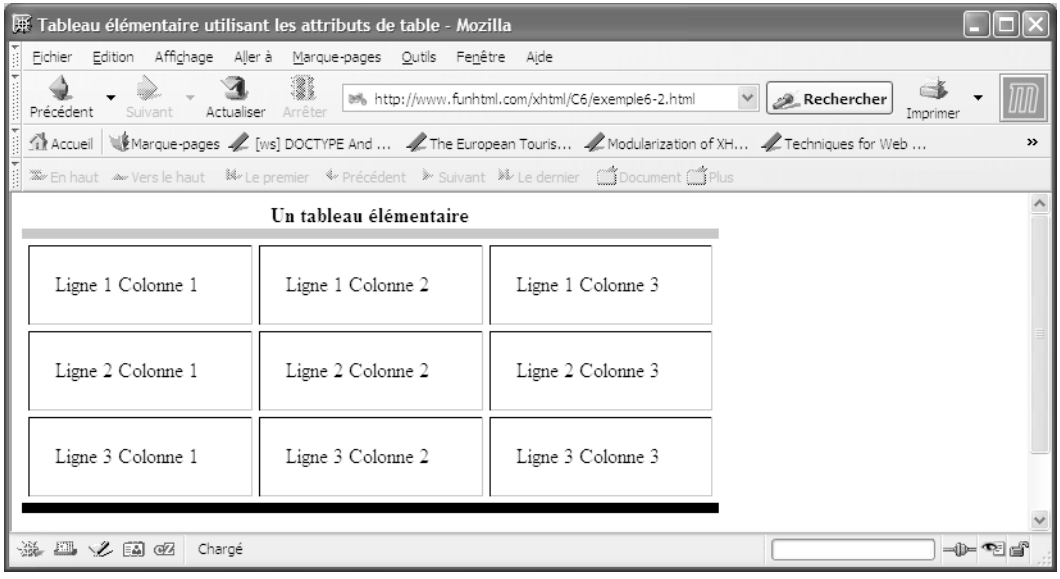


Figure 6-2
Les effets des attributs de l'élément <table>

Créer des groupes de lignes et de colonnes

Pour mieux structurer les données d'un tableau, nous pouvons y créer des groupes de lignes ou de colonnes. Cette structuration permet de mettre en évidence différentes parties du tableau et peut, dans ce cas, se révéler particulièrement utile en vue d'y appliquer des styles personnalisés.

Les groupes de lignes

Nous pouvons créer des groupes de lignes pour opérer une structuration horizontale du tableau. Il s'agit dans ce cas de réaliser un en-tête, un ou plusieurs groupes de lignes contenant les données proprement dites, et éventuellement un pied de tableau.

Un en-tête est créé à l'aide de l'élément <thead> Il contient autant d'éléments <tr> que de lignes nécessaires à l'en-tête, chacun contenant ensuite autant d'éléments <th> (au lieu de <td>) que nous avons besoin de colonnes. Chaque élément <th> contient généralement le titre d'une colonne. Par défaut, le contenu de ces éléments est mis en évidence en étant affiché en caractères gras et centré dans la cellule.

Quand le tableau est d'une hauteur supérieure à celle de la fenêtre du navigateur, il est utile de répéter ces titres en pied de tableau. Un pied de tableau peut aussi contenir les

totaux de chaque colonne dans les tableaux de données numériques. Pour créer ce pied de tableau nous écrivons l'élément <tfoot>, contenant comme <thead> les éléments <tr> qui, à leur tour, incluent des éléments <th>. Notons que la position de ces éléments dans le code n'influe pas sur leur position dans le tableau, et qu'ils doivent être écrits à l'intérieur de l'élément <table> avant la définition des lignes de données du tableau créées avec l'élément <tbody>. De plus, ils ne doivent apparaître chacun qu'une seule fois dans le tableau.

Entre l'en-tête et le pied de tableau se trouve le corps du tableau avec ses données. Il est créé en utilisant l'élément <tbody> qui contient un élément <tr> par ligne, puis des éléments <td> pour chaque cellule. Quand il est nécessaire de faire ressortir plusieurs groupes de données dans le corps du tableau, il est envisageable d'insérer plusieurs éléments <tbody> dans le même tableau. Cela permet d'appliquer un style différent pour chaque groupe.

En supplément de l'ensemble des attributs communs, les éléments <thead>, <tfoot> et <tbody> possèdent les attributs suivants qui permettent de définir la position du contenu de leurs cellules :

- align ="left | center | right | justify | char" : il définit l'alignement horizontal du contenu respectivement à gauche, au centre, à droite ou justifié dans la cellule. La valeur char permet l'alignement sur un caractère donné qui doit être précisé dans l'attribut char.
- valign ="top | middle | bottom | baseline" : il définit l'alignement vertical du contenu respectivement sur le haut, le milieu ou le bas de la cellule. La valeur baseline crée un alignement sur la ligne de base du texte (la ligne de support de pieds des lettres minuscules).
- char ="un_caractère" : il indique le caractère sur lequel doit être réalisé l'alignement des nombres décimaux si l'attribut align vaut char.
- charoff ="N px | N %" : il donne le décalage du caractère d'alignement défini dans char. Les attributs char et charoff ne sont actuellement pas pris en compte par les navigateurs, même les plus récents.

Nous pouvons définir l'alignement ligne par ligne ou même pour chaque cellule individuellement car les éléments <tr>, <td> et <th> possèdent les mêmes attributs align et valign. La structure d'un tableau comprenant des en-têtes, pied de page et corps de tableau est donc semblable à celle présentée dans l'exemple 6-3. Nous définissons dans l'élément <table> les attributs frame et rules pour n'afficher que les bordures externes du tableau et les lignes de séparation horizontales entre les différents groupes (repère ·), ainsi que l'espacement entre les cellules avec cellspacing et la largeur totale du tableau avec width. Il contient un en-tête et un pied de tableau qui incluent les titres de colonnes dans des éléments <th> (repères » et ¿). Viennent ensuite deux éléments <tbody> (repères ´ et ¾) dont le rôle est de définir des groupes de lignes auxquelles il est alors possible d'appliquer des styles différents. Le premier groupe contient trois lignes (repères ² , ¶ et º) et le second deux lignes (repères µ et ¸). Ces styles, que nous ne détaillerons

pas ici, sont définis dans l'élément <head> (repère ³) et nous permettent de bien distinguer les deux groupes comme nous pouvons le constater figure 6-3.

Exemple 6-3. Un tableau avec en-tête et pied de tableau

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html  xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Tableau avec en-tête et pied </title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" >
 .rouge{background-color:yellow;}
 .bleu{background-color:gray;color:#FFF;}
  table{border-color:red;}
 </style>
 </head>
 <body >
  <h1>Statistiques des contr&ocirc;les radars sur autoroute A7</h1>
  <table border="5" frame="box" rules="groups" cellspacing="5" width="90%">
   <thead><tr><th> Vitesse mesurée en km/h </th><th> Nombre de Véhicules </th>
    <th> Sanction opérées</th></tr></thead>
   <tfoot><tr><th> Vitesse mesurée en km/h </th><th> Nombre de Véhicules </th>
    <th> Sanction opérées</th></tr></tfoot>
   <tbody class="rouge" align="center">
    <tr>
     <td> de 90 à 110  km/h </td><td> 325 </td><td> NON </td>
    </tr>
    <tr>
     <td> de 110 à 130 km/h </td><td> 564 </td><td> NON </td>
    </tr>
    <tr>
     <td> de 130 à 140 km/h </td><td> 323 </td><td> NON </td>
    </tr>
   </tbody>
   <tbody class="bleu" align="center">
    <tr>
     <td> de 140 à 160 km/h </td><td> 223 </td><td> Amende </td>
    </tr>
    <tr>
     <td> de 160 à 190 km/h </td><td> 87 </td><td> Retrait </td>
    </tr>
   </tbody>
  </table>
 </body>
 </html>
```

La figure 6-3 présente le résultat obtenu grâce à ce code. On notera la séparation nette des groupes qui y est opérée à l'aide des couleurs et des bordures horizontales.

Figure 6-3

Un tableau avec en-tête, pied et deux groupes de données

L'élément <thead> permet de créer des titres de colonnes mais nous pouvons aussi envisager de structurer le tableau pour que les titres soient en tête de ligne et non plus de colonnes. Pour cela, le premier élément inclut dans chaque ligne créée avec <tr> doit être <th> au lieu de <td> Les éléments <thead> et <tfoot> n'ont alors plus d'utilité. L'exemple 6-4 reprend les données de l'exemple précédent en les réorganisant en lignes. Le tableau ne comprend plus que trois lignes (repères ³ , » et ´) dont le premier élément inclut est un titre de ligne incorporé dans <th> (repères · , ¿ et ²). La structuration des données est donc maintenant verticale. Nous pourrions appliquer des styles différents pour la première colonne, qui correspond aux éléments <th> et les colonnes suivantes, qui correspondent aux éléments <td>

Exemple 6-4. Un tableau avec des titres de lignes

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html  xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Tableau avec en-tête, pied et corps de données </title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body >
  <h1>Statistiques des contrôles radars sur l'autoroute A7</h1>
```

```
<table border="1" cellspacing="3" cellpadding="10">
 <tbody>
  <tr align="center">
    <th> Vitesse mesurée en km/h </th>
   <td> de 90 à 110  km/h </td>
   <td> de 110 à 130 km/h </td>
   <td> de 130 à 140 km/h </td>
   <td> de 140 à 160 km/h </td>
   <td> de 160 à 190 km/h </td>
  </tr>
  <tr align="center">
    <th> Nombre de véhicules </th>
   <td> 325 </td><td> 564 </td><td> 323 </td><td> 223 </td><td> 87 </td>
  </tr>
  <tr align="center">
    <th> Sanctions opérées</th>
   <td>NON</td><td>NON</td><td>NON</td><td>Amende</td><td>Retrait</td>
  </tr>
 </tbody>
</table>
</body>
</html>
```

La figure 6-4 présente la structure visuelle du tableau muni de titres de lignes.

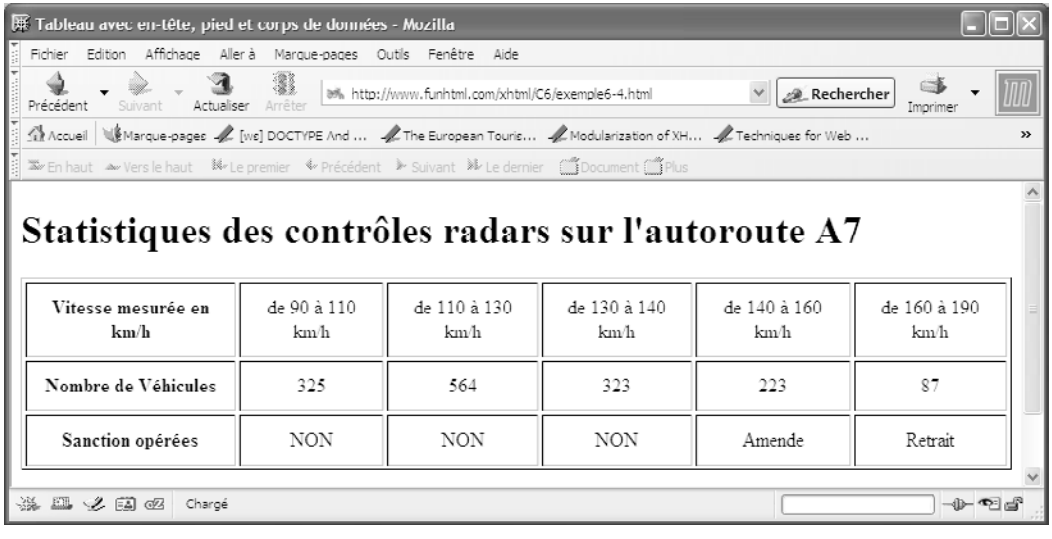


Figure 6-4

Un tableau avec des titres de lignes

Alignement du contenu des cellules

Les attributs `align` et `valign` des éléments `<table>`, `<thead>`, `<tfoot>`, `<tbody>`, `<tr>`, `<td>` et `<th>` permettent de définir l'alignement de leurs différents contenus. Comme tous ces éléments sont emboîtés les uns dans les autres, la définition de ces attributs à des niveaux différents implique des règles de priorité pour déterminer laquelle des valeurs doit l'emporter. Ces règles sont les suivantes :

- L'alignement défini dans un élément inclut dans `<td>` ou `<th>` l'emporte sur celui de son parent.
- L'alignement défini dans `<td>` ou `<th>` l'emporte sur celui de `<tr>`.
- L'alignement défini dans `<tr>` l'emporte sur ceux de `<thead>`, `<tfoot>` ou `<tbody>`
- L'alignement défini dans `<thead>`, `<tfoot>` ou `<tbody>` l'emporte sur celui de `<table>`

L'exemple 6-5 nous permet de vérifier les effets des attributs d'alignement du contenu des cellules. Dans l'élément `<tbody>` l'alignement est défini à gauche (repère ³). En première ligne, il est défini à droite dans l'élément `<tr>` (repère ·) et doit l'emporter sur la valeur précédente, mais la première cellule contient une définition d'alignement justifié (repère »), et c'est cette définition qui l'emporte sur celles de ses éléments parents. Dans le second élément `<td>`, l'alignement n'est pas défini explicitement (repère ¿), et c'est celui de son élément parent `<tr>` qui est appliqué et donc le texte est aligné à droite. La première cellule de la deuxième ligne a un contenu défini comme centré (repère ´) et le contenu de la seconde, n'ayant pas d'attribut `align` (repère ²), est aligné selon la valeur de son parent `<tr>` qui lui-même hérite de l'alignement défini dans l'élément `<tbody>` (repère ³). Nous retrouverons souvent ces notions d'héritage dans les styles CSS.

Exemple 6-5. Alignement du contenu des cellules

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Alignement du contenu des cellules</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <table border="3" width="100%" cellpadding="12px" title="Alignement du texte">
   <caption><big>Alignement du contenu des cellules</big></caption>
   <tbody align="left">  ³
    <tr align="right">    ·
     <td align="justify"      »  ><p><big>Texte justifié</big><br />
          ➥ In principio creavit Deus caelum et terram terra autem erat inanis et vacua
          ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas</p>
     </td>
     <td>  ¿  <p><big>Texte aligné à droite</big><br />
          ➥ In principio creavit Deus caelum et terram terra autem erat inanis et vacua
          ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas </p>
```

```
    </td>
   </tr>
   <tr>
    <td align="center"      ><p><big>Texte centré</big><br />
        In principio creavit Deus caelum et terram terra autem erat inanis et vacua
        et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas </p>
    </td>
    <td>  ²  <p><big>Texte aligné à gauche</big><br />
        In principio creavit Deus caelum et terram terra autem erat inanis et vacua
        et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas </p>
    </td>
   </tr>
  </tbody>
 </table>
</body>
</html>
```

La figure 6-5 montre les alignements obtenus dans Mozilla pour le code de l'exemple 6-5. L'alignement « justifié » n'est pas pris en compte par Explorer (pour compenser ce manque, voir la propriété CSS `text-align`).

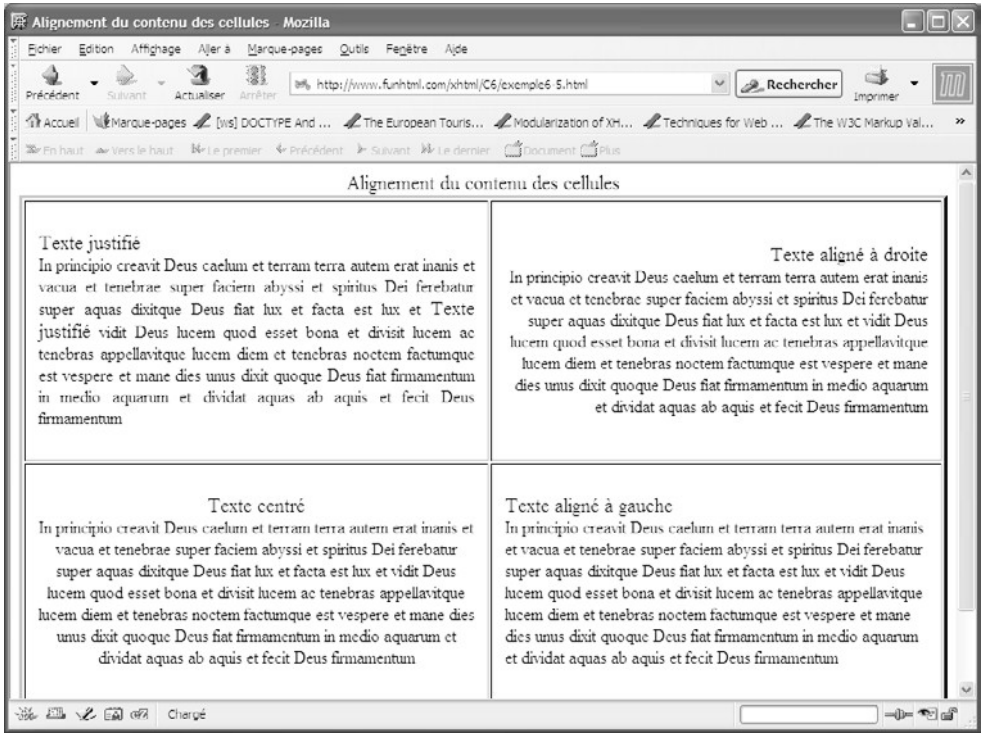


Figure 6-5

L'alignement du contenu des cellules

Les groupes de colonnes

Pour structurer l'information contenue dans un tableau, nous pouvons également créer des groupes de colonnes qui peuvent être matérialisés à l'écran en affichant uniquement leurs bordures, et non plus celles de toutes les cellules, en définissant les attributs `frame` et `rules` de l'élément `<table>`. On crée des groupes de colonnes au moyen des éléments `<colgroup>` et `<col />`.

Ces éléments possèdent, outre les attributs communs et notamment `id` qu'il est utile de définir pour chaque groupe, les attributs spécifiques suivants :

- `span="N"` : il définit le nombre de colonnes du groupe. Sa valeur par défaut est `1`.
- `width="N px | N %"` : définit la largeur de chaque colonne du groupe (et non pas de l'ensemble). Si elle est donnée en pourcentage, celui-ci se rapporte à la dimension de l'élément parent `<table>`.

En recourant à ces éléments, on ne crée pas les colonnes elles-mêmes mais on les associe en un groupe sémantique qui peut contenir une ou plusieurs colonnes. Il faut impérativement définir les groupes dans l'élément `<table>` avant les éléments `<tbody>`, `<thead>` et `<tfoot>`. On distingue deux manières de les mettre en œuvre.

Quand il est utilisé seul, l'élément `<colgroup>` permet de créer un groupe de colonnes identiques. On se sert de son attribut `span` pour indiquer le nombre de colonnes du groupe et de `width` pour déterminer la largeur commune de toutes les colonnes. L'exemple 6-6 en donne une illustration. Le tableau créé permet de procéder à l'affichage du récapitulatif d'une commande de livres. Le premier élément `<colgroup>` ne contient qu'une seule colonne dont l'en-tête est la date du livre. Sa largeur est définie par l'attribut `width` à 10 % de celle du tableau (repère ³). L'attribut `span` a ici explicitement la valeur `1`, mais sa définition n'est pas indispensable car il s'agit de la valeur par défaut. Le second groupe, qui utilise encore l'élément `<colgroup>` seul, définit deux colonnes identiques de largeurs égales à 25 % de celle du tableau, en définissant l'attribut `span` à la valeur `2` (repère ·). Elles contiennent le titre et le nom de l'auteur.

Pour créer un groupe de colonnes de dimensions différentes, il faut associer les éléments `<colgroup>` et `<col />`. Le premier définit le groupe, son attribut `id` permettant de l'identifier pour lui appliquer un style par exemple. Avec l'élément `<col />`, qui est le seul enfant possible du précédent, on définit chaque colonne ou même chaque sous-groupe de colonnes identiques. Chacun de ces sous-groupes contient alors N colonnes en définissant l'attribut `span="N"`. Leur largeur commune est indiquée en pixel ou en pourcentage dans l'attribut `width`. Pour créer plusieurs colonnes ayant chacune des caractéristiques différentes, nous devons inclure autant d'éléments `<col />` que nous désirons obtenir de sous-groupes et définir leurs attributs `span` et `width` de la même façon.

La création de sous-groupes est illustrée par l'exemple 6-6, dans lequel le troisième élément `<colgroup>` (repère ») contient deux éléments `<col />`. Le premier définit une colonne de largeur égale à 10 % de celle du tableau (repère ¸), et le second deux colonnes de largeur égales à 15 % chacune (repère ´).

Les dimensions en pourcentage faisant référence à la largeur totale du tableau, il faut prendre la précaution de vérifier que la somme de toutes les largeurs ne dépasse pas 100 %, sinon le navigateur risque d'afficher des résultats inattendus.

Le tableau contient donc trois groupes et un total de six colonnes dont les contenus sont donnés dans les éléments <tr> et <td> (repères ¶ , ° et ¾) inclus dans un élément <tbody> (repère ²).

Exemple 6-6. Création de groupes de colonnes

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Groupement de colonnes</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
<style type="text/css" title="">
.gras{font-weight:bold; background-color: red;}
.prix{background-color: yellow;}
.date{background-color:#AAA;color:blue;}
</style>
 </head>
 <body>
  <table border="5"   width="100%"  rules="groups" summary="Facture de livres"
   ➥ cellpadding="5">
   <caption><big>Facture de votre commande de livres</big></caption>
   <!-- Groupe 1 : Dates -->
    <colgroup id="date" width="10%" span="1" align="center" class="date">³
    </colgroup>
    <!-- Groupe 2 : Titre et Auteur -->
    <colgroup  id="titre" span="2" width="25%" align="left" class="gras">     ·
    </colgroup>
    <!-- Groupe 3: Quantité, Prix unitaire, Prix total -->
    <colgroup id="prix" span="3" align="right" class="prix">     »
     <col  width="10%" />  ¿
     <col span="2" width="15%"  />  ´
    </colgroup>
   <!—En-tête et pied du tableau -->
   <thead><tr><th>Date </th><th>Titre </th> <th>Auteur</th> <th>Quantité </th>
    ➥ <th>Prix Unitaire </th> <th>Prix Total </th> </tr> </thead>
   <tfoot><tr><th>Date </th> <th>Désignation </th> <th>Auteur</th> <th>Quantité
    ➥ </th> <th>Prix Unitaire </th> <th>Prix Total </th> </tr></tfoot>
   <!-- Données du tableau -->
   <tbody> ²
    <tr>  ¶
```

```
    <td>29/05/2005</td><td>XHTML Design</td><td>Jeffrey Zeldman</td><td>3</td>
      ➥ <td>32.00 &amp;euro;</td><td>96.00 &amp;euro;</td>
   </tr>
   <tr>
    <td>15/06/2005</td><td>CSS 2 </td><td>Raphael Goetter</td><td>2</td>
      ➥ <td>30.00 &amp;euro;</td><td>60.00 &amp;euro;</td>
   </tr>
   <tr>
    <td>10/09/2005</td><td>XHTML et CSS </td><td>Jean Engels</td><td>5</td>
      ➥ <td>29.90 &amp;euro;</td><td>149.50 &amp;euro;</td>
   </tr>
  </tbody>
 </table>
</body>
</html>
```

Pour matérialiser les groupes, des styles différents sont définis pour chacun d'eux. La figure 6-6 montre le résultat obtenu en tenant compte de ces styles. Les séparations entre les différents groupes y apparaissent nettement

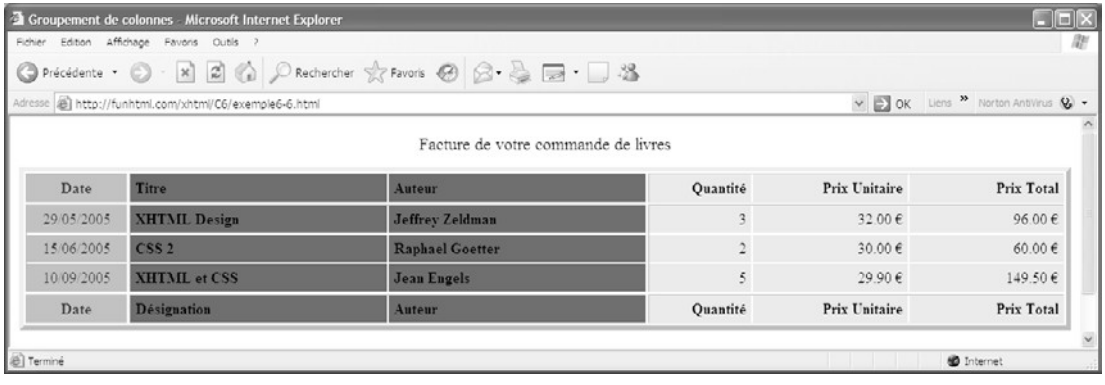


Figure 6-6
Les groupes de colonnes

Créer des tableaux irréguliers

Les tableaux que nous venons de créer ont tous une structure régulière, toutes les lignes et toutes les colonnes ayant le même nombre de cellules. Ce type de présentation peut se révéler contraignant, particulièrement quand le contenu de plusieurs cellules adjacentes est varié, puisqu'elles peuvent contenir aussi bien du texte que des éléments de type bloc ou de type en ligne comme des images occupant une grande surface. Pour créer des tableaux à structure irrégulière, il faut fusionner des cellules voisines comme nous pouvons

le faire dans un tableur. On procède à cette fusion en utilisant les attributs colspan et rowspan des éléments <td> et <th>.

Comme la création d'une mise en page de site, celle de tableaux irréguliers doit être précédée d'un travail sur le papier. Il faut tracer au préalable un quadrillage basé sur le plus grand nombre de lignes et de colonnes virtuelles nécessaires dans le tableau final. Il faut ensuite tracer les cellules qui vont être fusionnées pour obtenir l'aspect désiré. Il faudra également vérifier que le comptage final des cellules des colonnes et des lignes est bien constant, sachant que, dans ce décompte, la fusion de deux colonnes ou lignes compte pour deux cellules, même si elles n'en forment plus qu'une visuellement.

Ce type d'organisation dans le travail préparatoire est nécessaire pour que l'écriture du code soit facile à réaliser.

Fusion de colonnes

On procède à la fusion de cellules d'une même ligne en définissant la valeur de l'attribut colspan d'un élément <td> ou <th> avec un entier, lequel indique le nombre de cellules à fusionner en partant de la gauche. La syntaxe à suivre pour fusionner N cellules est la suivante :

```
<td colspan="N"> Contenu de la cellule</td>
```

Si nous considérons le code de l'exemple 6-7 et le résultat obtenu à la figure 6-7, nous constatons que le tableau a potentiellement cinq colonnes, comme le montre la ligne deux créée avec cinq éléments <td> (repère ¿).

Dans la première ligne (repère ³), les cellules des colonnes un et deux sont fusionnées ainsi que celles des cellules quatre et cinq en écrivant colspan="2" (repères · et ¿). Entre les deux se trouve une cellule ordinaire (repère »). Il y a bien un total de cinq colonnes virtuelles (2 + 1 + 2).

La deuxième ligne (repère ´) qui ne réalise aucune fusion a cinq colonnes définies par autant d'éléments <td> (repères ² à µ), la première et la dernière s'alignant bien avec celles de même position dans la ligne précédente.

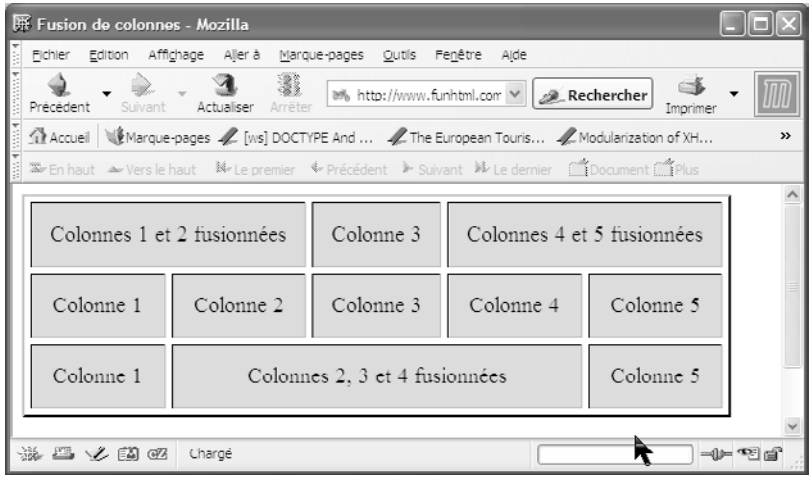
Dans la troisième ligne (repère ¸), l'attribut colspan est utilisé dans le deuxième élément <td> (repère) et donc les cellules des colonnes 2, 3 et 4 sont fusionnées en une seule. La première et la dernière cellules sont normales (repères ¹ et). Il y a également un total de cinq colonnes virtuelles dans cette ligne (1 + 3 + 1).

Exemple 6-7. Fusion de colonnes

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Fusion de colonnes</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
```

```
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
<style type="text/css" title="">
 td {background-color:lightblue;}
</style>
 </head>
 <body>
  <table border="2" cellspacing="5" cellpadding="15">
   <tr align="center">
        <td colspan="2"><big> Colonnes 1 et 2 fusionnées </big></td>
        <td><big>  Colonne 3 </big></td>
        <td colspan="2"><big> Colonnes 4 et 5 fusionnées </big></td>
   </tr>
   <tr align="center">
        <td><big> Colonne 1 </big></td>
        <td><big> Colonne 2 </big></td>
        <td><big> Colonne 3 </big></td>
        <td><big> Colonne 4 </big></td>
        <td><big> Colonne 5 </big></td>
   </tr>
   <tr align="center">
        <td><big> Colonne 1</big></td>
        <td colspan="3"><big> Colonnes 2, 3 et 4 fusionnées </big></td>
        <td><big> Colonne 5</big></td>
   </tr>
  </table>
 </body>
</html>
```

Figure 6-7
La fusion des colonnes



Un cas particulier de fusion de colonnes

Dans certains cas de mise en page de tableau, la notion précédente de colonnes virtuelles prend tout son sens car, dans aucune ligne, nous ne voyons apparaître ce nombre de colonnes, mais uniquement un des diviseurs de ce nombre. Supposons par exemple que nous souhaitions réaliser le tableau représenté à la figure 6-8. Il est composé d'une colonne à la première ligne, de deux à la deuxième, et ainsi de suite jusqu'à la quatrième ligne.

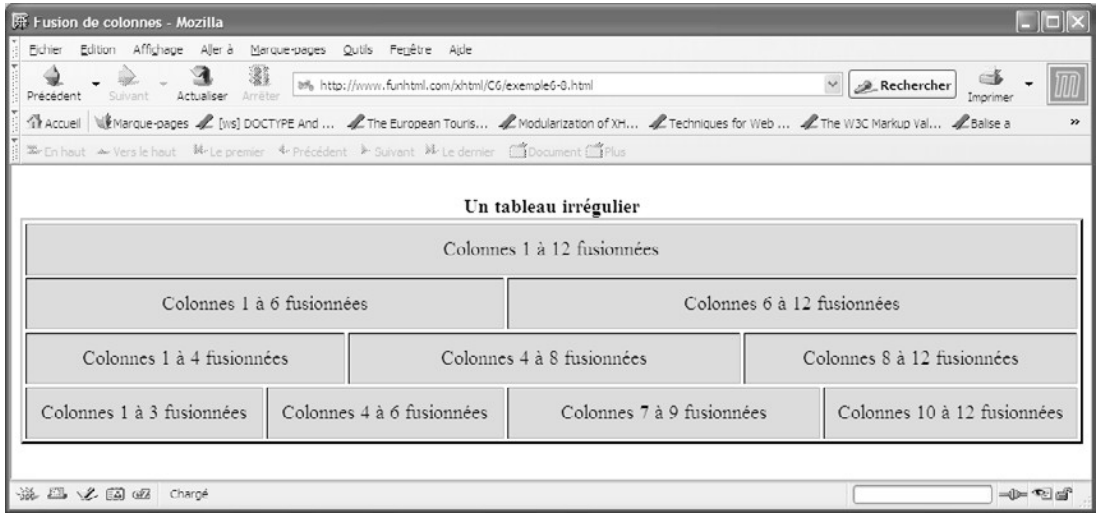


Figure 6-8

Un tableau très irrégulier

Il faut que notre tableau ait virtuellement douze colonnes (soit le plus petit multiple commun à 2, 3 et 4) pour pouvoir obtenir des divisions de ce type. L'exemple 6-8 permet de créer ce tableau. Dans la première ligne, nous définissons l'attribut `colspan` avec la valeur `12` pour fusionner nos douze cellules virtuelles afin d'obtenir une cellule unique (repère ³). Dans la deuxième, cet attribut prend la valeur `6` pour obtenir deux colonnes (repères · et »). La troisième ligne opère la fusion des colonnes par quatre en écrivant `colspan="4"` pour en obtenir trois visuellement (repères ¿ , ´ et ²), et la dernière fusion est faite par groupe de trois colonnes avec le code `colspan="3"` pour en obtenir quatre (repères ¶ , º , ¾ et µ).

Exemple 6-8. Fusion des cellules virtuelles

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
```

```
  <title>Fusion de colonnes</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
<style type="text/css" title="">
 td {background-color:lightblue;}
</style>
 </head>
 <body>
  <table border="2" cellspacing="2" width="90%">
   <!-- Ligne 1 -->
   <tr align="center">
    <td colspan="12"><big> Colonnes 1 à 12 fusionnées</td>
   </tr>
   <!-- Ligne 2 -->
   <tr align="center">
    <td colspan="6"><big> Colonnes 1 à 6 fusionnées</td>
    <td colspan="6"><big> Colonnes 6 à 12 fusionnées</td»
   </tr>
   <!-- Ligne 3 -->
  <tr align="center">
    <td colspan="4"><big> Colonnes 1 à 4 fusionnées</td>
    <td colspan="4"><big> Colonnes 4 à 8 fusionnées</td>
    <td colspan="4"><big> Colonnes 8 à 12 fusionnées</td>
   </tr>
   <!-- Ligne 4 -->
   <tr align="center">
    <td colspan="3"><big> Colonnes 1 à 3 fusionnées </td>¶
    <td colspan="3"><big> Colonnes 4 à 6 fusionnées </td>°
    <td colspan="3"><big> Colonnes 7 à 9 fusionnées </td>¾
    <td colspan="3"><big> Colonnes 10 à 12 fusionnées </td>╣
   </tr>
  </table>
 </body>
</html>
```

Cet exemple montre l'importance de la préparation graphique qui doit être réalisée avant le codage XHTML afin de créer des fusions opérationnelles de manière rigoureuse. En observant bien la figure 6-8, nous pouvons remarquer que les colonnes de la même ligne n'ont pas toujours la même largeur. Ce problème pourra être résolu en appliquant les propriétés CSS de dimensionnement aux différentes cellules du tableau.

La fusion de lignes

La fusion de cellules situées dans les lignes adjacentes peut être définie à l'aide de l'attribut `rowspan` des éléments `<td>` et `<th>`. Il a pour valeur le nombre de cellules à fusionner. La fusion doit être déclarée dans la cellule la plus haute. Sa syntaxe est donc la suivante :

```
<td rowspan="N"> Contenu de la cellule</td>
```

Ici encore, la phase de conception sur le papier doit permettre d'effectuer un décompte des cellules virtuelles, sachant que nous devons en retrouver un nombre égal pour chaque colonne. Comme nous l'avons déjà mentionné, la création des tableaux donne la priorité aux lignes sur les colonnes et est faite ligne après ligne en incluant des éléments `<tr>`. La fusion de lignes demande une plus grande attention que celle des colonnes. En effet, le fait d'écrire par exemple `rowspan="3"` dans un élément `<td>` implique que, dans les deux éléments `<tr>` qui suivent, il doit y avoir une définition de cellule de moins par rapport au nombre total de colonnes. Dans le même élément `<td>` ou `<th>` il est possible d'inclure simultanément les attributs `colspan` et `rowspan` pour réaliser la fusion de lignes et de colonnes.

Pour créer par exemple le tableau représenté à la figure 6-9, il faut écrire le code de l'exemple 6-9.

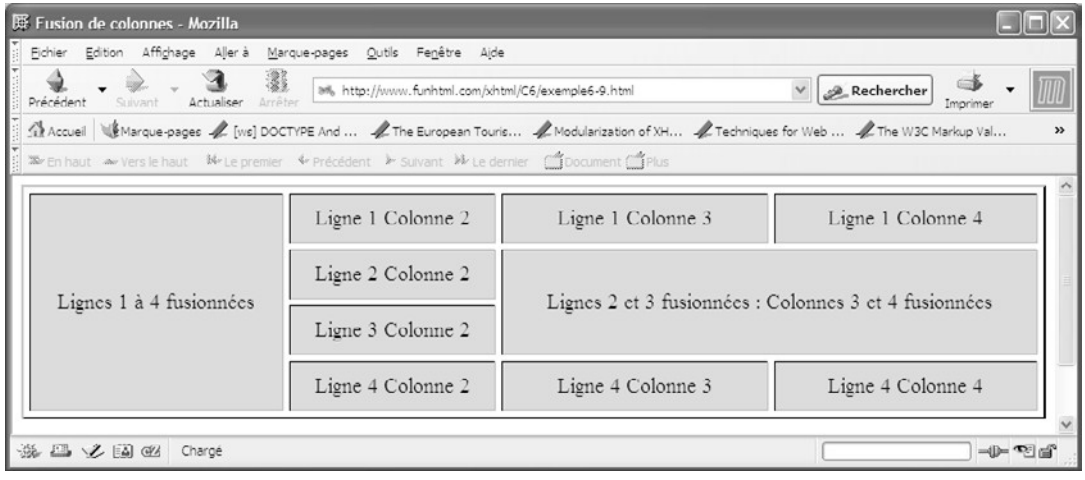


Figure 6-9

La fusion de cellules de lignes adjacentes

Exemple 6-9. Fusion de lignes

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html  xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Fusion de colonnes</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
<style type="text/css" title="">
 td {background-color:lightblue;}
</style>
```

```
</head>
<body>
 <table border="2" cellspacing="5" width="100%" cellpadding="10" >
  <!-- Ligne 1 -->
  <tr align="center">
      <td rowspan="4"><big> Lignes 1 à 4 fusionnées</big></td>
   <td><big>Ligne 1 Colonne 2</big></td>
   <td><big>Ligne 1 Colonne 3</big></td>
   <td><big>Ligne 1 Colonne 4</big></td>
  </tr>
  <!-- Ligne 2 -->
  <tr align="center">
   <td ><big>Ligne 2 Colonne 2</big></td>
      <td colspan="2" rowspan="2"><big>Lignes 2 et 3 fusionnées : Colonnes 3 et 4
          fusionnées</big></td>
  </tr>
  <!-- Ligne 3 -->
  <tr align="center">
   <td><big>Ligne 3 Colonne 2</big></td>
  </tr>
  <!-- Ligne 4 -->
  <tr align="center">
   <td><big>Ligne 4 Colonne 2 </big></td>
   <td><big>Ligne 4 Colonne 3</big></td>
   <td><big>Ligne 4 Colonne 4 </big></td>
  </tr>
 </table>
</body>
</html>
```

Dans la première ligne du tableau (repère ³), nous créons la fusion des quatre cellules de la première colonne (repère ·) puis nous créons les trois colonnes suivantes de cette ligne (repères » , ¿ et ´). À ce stade, nous savons donc déjà que les éléments <tr> suivants ne doivent plus contenir qu'un maximum de trois éléments <td>

Dans la deuxième ligne (repère ²), nous créons d'abord une cellule normale qui correspond à la deuxième colonne (repère ¶), puis la suivante qui est la fusion des colonnes 3 et 4 avec le code colspan="2" et la fusion des lignes 2 et 3 avec le code rowspan="2" (repère º). La cellule obtenue contient donc quatre cellules virtuelles. La troisième ligne (repère ¾) ne contient donc plus qu'un seul élément <td> (repère µ), les autres cellules de la même ligne ayant été déclarées implicitement dans les éléments <tr> et <td> précédents. Quant à la quatrième ligne (repère ¸), elle contient trois éléments <td> correspondant aux colonnes 2, 3 et 4 (repères ¹ , et).

Imbrication de tableaux

Chaque élément <td>ou <th>inclut dans une ligne peut lui-même contenir un élément <table>comme nous l'avons mentionné au tableau 6-2. Cela permet d'inclure un nouveau tableau dans une cellule d'un premier tableau, ce qui entraîne une imbrication de tableau.

Le tableau créé à l'exemple 6-10 illustre cette possibilité nous permettant d'afficher un calendrier, représenté à la figure 6-10. Ce dernier comporte quatre cellules virtuelles principales, réparties en deux groupes (repère ³), celles de la première colonne étant fusionnées pour n'en former qu'une contenant l'année dans un élément <th>(repère ·). La deuxième cellule de la première ligne contient le nom du mois dans un second élément <th>(repère »). La deuxième cellule de la deuxième ligne va contenir le calendrier proprement dit. Celui-ci est créé dans un nouveau tableau, constitué de sept lignes et de sept colonnes. Un second élément <table>est donc inclut dans la dernière cellule du premier tableau (repère ¿). La première ligne de ce tableau contient le nom des jours (repère ´), et les suivantes les dates (repères ², ¶, º, ¾, µ et ¸).

Exemple 6-10. Imbrication de tableaux

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html  xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Tableaux inclus l'un dans l'autre</title>
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <!-- Premier tableau -->
  <table border="4">
   <caption><strong> Calendrier Octobre 2005 </strong></caption>
   <colgroup span="1" width="15%"></colgroup>
   <colgroup span="1" width="85%"></colgroup>
   <tbody>
   <tr align="center">
     <th rowspan="2" ><h1> 2 <br /> 0 <br /> 0 <br /> 5 <br /> </h1></th>
     <th valign="middle"><h2>OCTOBRE</h2></th>
   </tr>
   <tr align="center">
    <td>
     <!-- Deuxième tableau -->
     <table border="5">
      <tbody>
      <tr align="center">
        <th>Lun</th>
        <th>Mar</th>
        <th>Mer</th>
        <th>Jeu</th>
        <th>Ven</th>
```

```
    <th>Sam</th>
    <th>Dim</th>
  </tr>
<tr align="center">
  <td> </td>
  <td> </td>
  <td> </td>
  <td> </td>
  <td> </td>
  <td>1</td>
  <td>2</td>
</tr>
<tr align="center">
  <td>3</td>
  <td>4</td>
  <td>5</td>
  <td>6</td>
  <td>7</td>
  <td>8</td>
  <td>9</td>
</tr>
<tr align="center">
  <td>10</td>
  <td>11</td>
  <td>12</td>
  <td>13</td>
  <td>14</td>
  <td>15</td>
  <td>16</td>
</tr>
<tr align="center">
  <td>17</td>
  <td>18</td>
  <td>19</td>
  <td>20</td>
  <td>21</td>
  <td>22</td>
  <td>23</td>
</tr>
<tr align="center">
    <td>24</td>
  <td>25</td>
  <td>26</td>
  <td>27</td>
  <td>28</td>
  <td>29</td>
  <td>30</td>
</tr>
<tr align="center">
```

```
      <td>31</td>
        <td> </td>
        <td> </td>
        <td> </td>
        <td> </td>
        <td> </td>
        <td></td>
      </tr>
      </tbody>
      </table>
     </td>
    </tr>
   </tbody>
  </table>
 </body>
</html>
```

La figure 6-10 montre le résultat obtenu.

Figure 6-10
Un calendrier créé par inclusion de tableaux

Organisation d'une page à l'aide de tableaux

Dans les années 1990, la tendance en matière de création de sites était de structurer les pages au moyen de tableaux utilisés en tant que grille contenant les différentes parties de la page. L'utilisation de tableaux couplée à la définition de leur largeur en pourcentage à

l'aide de l'attribut `width` peut permettre d'obtenir un tableau qui occupe par exemple toute la largeur d'une page, quelle que soit la définition de l'écran du poste client. Afin de rendre les pages plus attractives et que le visiteur ne perçoive pas la grille générée par l'élément `<table>`, on utilise souvent comme astuce la définition de son attribut `border` à la valeur `0`, ce qui rend les bordures des cellules invisibles. De nombreux sites utilisent encore ces principes. À titre d'expérience, rendez-vous sur le site `www.amazon.fr` et affichez-en le code source (Affichage>Code source dans Mozilla par exemple), puis enregistrez-le et ouvrez ce fichier dans votre éditeur XHTML. Cherchez ensuite tous les éléments `<table>` (Search>Find dans EditPlus par exemple) et remplacez la valeur `0` de l'attribut `border` par la valeur `1` ou `2`. Vous pourrez constater, outre la non-conformité du code source à XHTML, l'usage abusif des tableaux dans cette page (pas moins de 43 éléments `<table>` !) uniquement dans le but d'organiser la présentation de la page. Tentez ensuite les mêmes opérations sur le site `www.eyrolles.com`, strictement conforme à la DTD XHTML 1.0, et comparez le nombre d'éléments `<table>`. Vous n'en trouverez qu'un seul. Il s'agit bien là de deux politiques radicalement différentes, reconnaissons-le. Chacun choisira la sienne, mais je l'espère en connaissance de cause, après la lecture de ce livre, où vous constaterez dans la deuxième partie qu'il est parfaitement possible de se passer de la mise en page systématique à l'aide de tableaux. Les figures 6-11 et 6-12 montrent la page d'accueil du site `www.amazon.fr` sans bordures de tableaux et avec une bordure de 1 pixel ensuite, ce qui met en évidence tous les tableaux inclus dans la page.

Figure 6-11

Le site www.amazon.fr sans bordures de tableaux

Figure 6-12

Le site www.amazon.fr avec des bordures de tableaux

Nous allons cependant présenter cette possibilité de travail en soulignant que l'utilisation conjointe de divisions et de propriétés CSS de positionnement et de dimensionnement présente des solutions aux problèmes de mise en page.

La présentation choisie est simple et divise la page en quatre zones distinctes, à savoir un bandeau contenant le titre général du site, une zone de menu située sur la gauche, une zone principale affichant le contenu au centre et enfin une zone de liens utiles située à droite. L'aspect final est présenté à la figure 6-13. Le tableau incluant tous ces éléments est plutôt simple car il ne contient que six cellules virtuelles. Les éléments <colgroup> et <col /> permettent de dimensionner les trois colonnes de largeurs respectives de 15 % pour le menu, 70 % pour le contenu et 15 % pour les liens (repères ³ , · , » et ¿). Ces largeurs sont données en pourcentage de celle du tableau qui occupe toute la largeur de la fenêtre du navigateur car l'attribut width de l'élément <table> a la valeur 100 %

Les trois cellules de la première ligne (repère ´) sont fusionnées pour n'en former qu'une seule, contenant le titre du site dans un élément <h1> (repère ²).

La deuxième ligne (repère ¶) contient les trois cellules principales (repères º , µ et). La cellule de gauche du menu (repère º) comporte une liste à puces des liens vers les différentes pages du site (repère ¾), qui abordent les sujets préférés des webmestres.

La cellule centrale (repère µ) affiche un titre général (repère ¸), deux sous-titres suivis chacun d'une division incluant le contenu rédactionnel de la page d'accueil (repères¹ et puis et).

La cellule de droite (repère) contient une liste de liens utiles.

Exemple 6-11. Mise en page à l'aide de tableau

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
 <title>Mise en page à l'aide de tableaux</title>
  <meta name="Author" content="Jean ENGELS" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <table border="0" width="100%" cellspacing="15px" cellpadding="5">
  <!--Définition de largeurs des colonnes  -->
  <colgroup>³
   <col span="1" width="15%" />·
   <col span="1" width="70%" />»
   <col span="1" width="15%" />¿
  </colgroup>
  <!-- Première ligne -->
  <tr align="center">
   <td colspan="3"><h1>XHTML 1.1 et CSS 2.1</h1><hr /></td>
  </tr>
  <!-- Deuxième ligne-->
  <tr valign="top">
  <!-- Première colonne -->
   <td>
    <ul>
     <li><h2><a id="xhtml" href="xhtml.html" tabindex="1" accesskey="X" title="
        La page XHTML 1.1">XHTML 1.1 </a></h2></li>
     <li><h2><a id="css" href="css.html" tabindex="2" accesskey="A" title="
        La page CSS 2.1">CSS 2.1 </a></h2></li>
     <li><h2><a id="js" href="javascript.html" tabindex="3" accesskey="J" title="
        La page JavaScript">JavaScript </a></h2></li>
     <li><h2><a id="php" href="php.html" tabindex="4" accesskey="P" title="
        La page PHP 5">PHP 5 </a></h2></li>
     <li><h2><a id="mysql" href="mysql.html" tabindex="5" accesskey="B" title="
        La page MySQL 5">MySQL 5 </a></h2></li>
     <li><h2><a id="sqlite" href="sqlite.html" tabindex="6" accesskey="S" title="
        La page SQLite">SQLite </a></h2></li>
```

```
    <li><h2><a id="xml" href="xml.html" tabindex="7" accesskey="D" title="
      ➥ La page XML">XML </a></h2></li>
   </ul>
  </td>
  <!-- Deuxième colonne -->
  <td align="justify">
   <h2>Utilisez les standards du Web </h2>
   <h3>XHTML 1.1</h3>
   <div> L'adoption des standards du Web devient un préalable à la création de
     ➥ sites modernes et accessibles à tous avec XHTML et CSS. L'adoption des
     ➥ standards du Web devient un préalable à la création de sites modernes et
     ➥ accessibles à tous. L'adoption des standards du Web devient un préalable
     ➥ à la création de  sites modernes et accessibles à tous . . .
   </div>
   <h3>CSS 2.1</h3>
   <div> L'adoption des standards du Web devient un préalable à la création de
     ➥ sites modernes et accessibles à tous L'adoption des standards du Web devient
     ➥ un préalable à la création de sites modernes et accessibles à tous.
     ➥ L'adoption des standards du Web devient un préalable à la création de sites
     ➥ modernes et accessibles à tous . . .
   </div>
   </td>
   <!-- Troisième colonne -->
   <td align="left">
    <a href="mailto:xhtml@funhtml.com">Contact </a><br /><br />
    <a href="funhtml.com/xhtml"> Référence XHTML 1.1 </a><br /><br />
    <a href="funhtml.com/css"> Référence CSS 2.1 </a><br /><br />
    <a href="w3.org"> Le site du W3C</a><br /><br />
    <a href="eyrolles.com"> La librairie Eyrolles</a><br /><br />
   </td>
  </tr>
 </table>
</body>
</html>
```

La figure 6-13 donne le résultat obtenu pour cette mise en page. Dans un but pédagogique, nous n'avons envisagé qu'une mise en page simple ne comportant que quatre zones, mais sur le même principe il est évidemment possible de créer davantage de cellules et d'opérer des regroupements de colonnes plus complexes.

Nous pouvons dès à présent constater que l'aspect visuel de la page est des plus pauvres. Le code de cette page est en effet purement XHTML et remplit bien la fonction qui lui est assignée de structurer l'information. Les seules informations de redimensionnement sont faites dans l'attribut `width` des éléments `<table>` et `<col />`. En respectant uniquement les recommandations XHTML 1.1, il est impossible de faire mieux. En effet, aucun attribut ne permet de définir une couleur pour le fond ou le texte, ni la taille des caractères, et

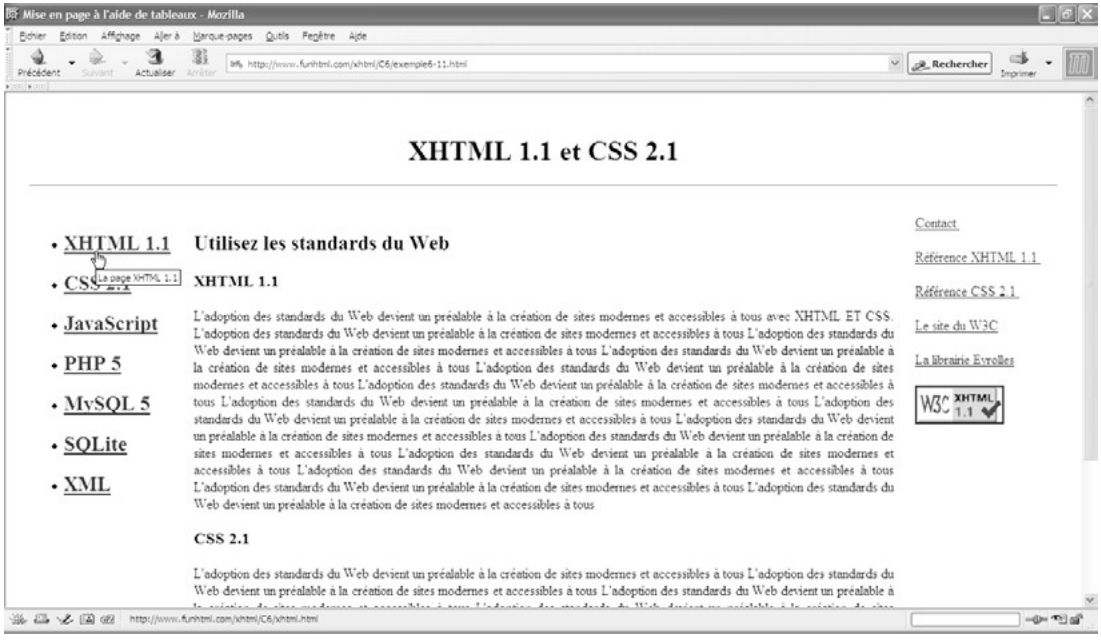


Figure 6-13

Mise en page à l'aide d'un tableau

chacun d'entre nous, maintenant convaincu de l'utilité de séparer le contenu et la présentation, ne peut que s'en réjouir car la structuration du contenu de la page est bien réalisée en quatre zones distinctes.

Une présentation plus esthétique ne peut être obtenue qu'en utilisant des styles CSS appliqués aux différents éléments du code XHTML de cet exemple. Nous verrons par la suite comment appliquer des styles aux éléments de tableau ou à leurs contenus (voir la deuxième partie). De plus, le type de présentation de la page en trois colonnes pourra être obtenu sans même avoir à utiliser de tableau, simplement en positionnant les différentes divisions de la page (voir le chapitre 13) d'une manière plus souple et moins lourde au niveau du code.

En conclusion, il n'est pas dans mes intentions de déconseiller systématiquement l'emploi des tableaux, mais plutôt de faire comprendre que leur emploi doit être réservé à des situations spécifiques comme la structuration de données numériques pour lesquelles ils sont particulièrement destinés. Si vous n'avez pas l'idée d'écrire une lettre de motivation et votre CV avec un tableur Excel ou OpenOffice (même si c'est potentiellement réalisable), vous n'utiliserez pas non plus un tableau pour structurer vos pages web en emboîtant de multiples tableaux les uns dans les autres.

Exercices

Exercice 1 : Citez tous les éléments spécialisés utilisables pour la création des tableaux et leur ordre d'apparition dans le code XHTML.

Exercice 2 : Peut-on inclure un tableau dans les éléments <body><form>ou <td>?

Exercice 3 : Créez un tableau composé de quatre lignes et de cinq colonnes.

Exercice 4 : Créez un tableau de 600 pixels de large contenant cinq lignes et trois colonnes avec des bordures de 4 pixels, un espacement entre cellules de 10 pixels. Seules les bordures verticales doivent s'afficher.

Exercice 5 : Construisez un tableau statistique ayant un en-tête, un pied de page et deux corps de données. N'affichez que les bordures horizontales des groupes de données. Centrez le contenu des cellules de données et alignez à gauche les titres de l'en-tête et du pied de page.

Exercice 6 : Créez un tableau composé de sept colonnes et de cinq lignes avec des bordures verticales de groupes. Les colonnes sont organisées en groupe. Le premier groupe a deux colonnes de 100 pixels chacune, le deuxième trois colonnes de 150 pixels et le troisième deux colonnes de 50 pixels et une de 100 pixels.

Exercice 7 : Créez le tableau suivant en fusionnant les colonnes.

<table>
<tr><td>1234</td><td colspan="2"></td><td></td></tr>
<tr><td>567</td><td colspan="2"></td><td></td></tr>
<tr><td>8910</td><td></td><td colspan="2"></td></tr>
<tr><td colspan="2">11</td><td>12</td><td>13</td><td>14</td></tr>
</table>

Exercice 8 : Créez le tableau suivant en fusionnant les lignes.

<table>
<tr><td>1</td><td rowspan="4">2</td><td rowspan="2">3</td><td>45</td><td></td></tr>
<tr><td rowspan="2">6</td><td>7</td><td rowspan="3">8</td></tr>
<tr><td>9</td><td rowspan="2">10</td></tr>
<tr><td>11</td><td>12</td></tr>
</table>

Exercice 9 : Créez un tableau selon le modèle suivant.

<table>
<tr><td colspan="2">123</td><td colspan="2"></td><td></td></tr>
<tr><td>4</td><td rowspan="3">5</td><td>678</td><td></td><td></td></tr>
<tr><td rowspan="3">9</td><td>10</td><td>11</td><td>12</td></tr>
<tr><td>13</td><td colspan="2" rowspan="2">14</td></tr>
<tr><td>15</td><td>16</td></tr>
</table>

Exercice 10 : Créez un tableau selon le modèle suivant.

1234			
5	6		7
8			9
10	11	12	13

Exercice 11 : Créez un tableau selon le modèle suivant.

1			
2	34		5
	67		
	8		

Exercice 12 : Créez un tableau selon le modèle suivant en utilisant la notion de cellules virtuelles.

12345						
678			9		101112	
13						
14		15			16	
17	18	19	20	21	22	23

Exercice 13 : À partir d'un tableau de deux lignes et de deux colonnes, fusionnez les deux cellules inférieures. Insérez dans les deux cellules supérieures des tableaux de deux lignes et cinq colonnes, puis dans la cellule inférieure un tableau de trois lignes et six colonnes selon le modèle suivant.

12345				
678910				

ABCDE				
FGHIJ				

abcdef					
ghijkl					
mnopqr					

7

Créer des formulaires

S'inscrire dans une mailing-list, laisser son avis dans un livre d'or, saisir un mot-clé dans un moteur de recherche ou passer une commande en ligne, toutes ces actions aujourd'hui devenues très courantes pour nombre d'entre nous mais ne sont possibles que grâce à l'existence des formulaires insérés dans une page web. Tout échange de données entre un visiteur (le poste client) et l'ordinateur hébergeant le site (le serveur), opéré via le protocole HTTP, se fait donc via les saisies effectuées dans un formulaire. L'utilisateur peut entrer des textes ou des mots de passe, opérer des choix grâce à des boutons radio, de cases à cocher ou des listes de sélection. Il peut également effectuer le transfert de ses propres fichiers vers le serveur, par exemple sur les sites proposant le tirage de photographies numériques par correspondance. Les formulaires sont donc présents dans un très grand nombre de sites. Nous allons aborder dans ce chapitre les éléments XHTML qui permettent de créer la structure et les différents composants des formulaires. Nous étudierons également quelques notions relatives à la récupération des données côté serveur à l'aide de scripts PHP.

Structure d'un formulaire

Les éléments constitutifs d'un formulaire doivent être contenus entre les balises `<form>` et `</form>` Comme il s'agit d'un élément de niveau bloc, il peut être inclus directement dans l'élément `<body>` Le tableau 7-1 présente la liste exhaustive de ses éléments parents. L'élément `<form>` peut contenir entre autres des éléments de titre, des blocs (mais jamais `<form>`), des listes et l'élément `<fieldset>` pour structurer son contenu. Aucun des composants de formulaire que nous allons étudier dans ce chapitre ne peut être inclus directement dans l'élément `<form>` et il faut d'abord y inclure un autre élément.

Le tableau 7-2 présente la liste de tous ses éléments enfants.

Tableau 7-1. Les éléments parents de l'élément <form>

blockquote, body, dd, del, div, fieldset, ins, li, map, noscript, object, td, th

Tableau 7-2. Les éléments enfants de l'élément <form>

address, blockquote, del, div, dl, fieldset, h1, h2, h3, h4, h5, h6, hr, ins, noscript, ol, p, pre, script, table, ul

L'élément <fieldset> est très souvent inclus dans un formulaire. Il a pour vocation de délimiter les groupes de composants actifs du formulaire. D'un point de vue graphique, chaque groupe de composants est délimité par défaut, dans les navigateurs, par une bordure mince dans laquelle il est possible d'intégrer un titre pour chaque groupe. Chaque titre est le contenu d'un élément <legend> inclus dans <fieldset>. L'effet obtenu pour un formulaire à deux groupes est présenté à la figure 7-1. Chaque groupe peut contenir des titres, des blocs, des listes et des éléments de type en ligne parmi lesquels figurent ceux qui créent les composants actifs propres aux formulaires. L'élément <fieldset> peut avoir comme contenu une grande variété d'éléments dont la liste est donnée au tableau 7-3.

Tableau 7-3. Les éléments enfants de l'élément <fieldset>

Texte, a, abbr, acronym, address, b, bdo, big, blockquote, br, button, cite, code, del, dfn, div, dl, em, fieldset, form, h1, h2, h3, h4, h5, h6, hr, i, img, input, ins, kbd, label, legend, map, noscript, object, ol, p, pre, q, samp, script, select, small, span, strong, sub, sup, table, textarea, tt, ul, var

Le code XHTML de la structure d'un formulaire est donc de la forme présentée dans l'exemple 7-1.

Exemple 7-1. Structure d'un formulaire

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Formulaire de base</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <form method="post" action="traitement.php">
   <fieldset>
```

```
    <legend>Formulaire type</legend>
    <!-- Composants du formulaire -->
   </fieldset>
  </form>
 </body>
</html>
```

Les attributs de l'élément <form>

L'élément `<form>` possède l'ensemble des attributs communs (`id`, `title`, `class`, `xml:lang`, `dir` et les gestionnaires d'événements de base). Nous ne manquerons pas d'utiliser systématiquement l'attribut `id` pour identifier le formulaire dans une page qui peut en compter plusieurs, ainsi que l'attribut `title` pour améliorer l'accessibilité du site.

En supplément de ses attributs communs désormais bien connus, l'élément `<form>` possède les attributs propres suivants :

- `action` : cet attribut est obligatoire car il désigne le fichier qui est chargé de traiter les données du formulaire côté serveur. Je recommande particulièrement l'emploi du langage PHP pour écrire les scripts de traitement de ces données, d'autres solutions, JSP et ASP.Net par exemple, peuvent également être envisagées. Le code du script de traitement des données peut être inclus dans un fichier séparé de celui qui contient le code XHTML mais il peut être inclus dans ce même fichier qui doit alors avoir une extension appropriée à la place de `.html` (par exemple, `.php` pour le langage PHP ou `.aspx` pour ASP.Net). De plus, dans ce cas, comme nous l'avons indiqué au chapitre 2, la déclaration XML doit être supprimée. La structure initiale de la page est alors celle de l'exemple 2-2. Le contenu de l'attribut `action` doit donc être une URL, qui peut être relative, de la forme :

```
<form action="traitement.php">
```

Dans ce cas, le fichier désigné doit être présent dans le même répertoire que le fichier XHTML. L'URL peut également être absolue, que le fichier soit présent sur le même serveur, ou même sur un autre serveur. Elle est alors de la forme suivante :

```
<form action="http://www.monsite.com/scripts/traitement.php">
```

Traitement par le fichier lui-même

Si le script de traitement est incorporé dans le fichier XHTML, nous pouvons utiliser du code PHP pour désigner le fichier lui-même. En cas changement de nom du fichier, il n'y a ainsi rien à modifier. Pour réaliser cette opération, il convient d'écrire le code suivant :

```
<form action="<?= $_SERVER['PHP_SELF'] ?>">
```

L'attribut `action` peut aussi avoir la valeur `mailto:` qui provoque l'envoi des données (leur nom et leur valeur) à une adresse e-mail qu'il faut préciser à la suite du mot-clé `mailto:`, par exemple :

```
<form method="post" action="mailto:xhtml@funhtml.com">
```

Pour que la transmission vers une adresse e-mail ait lieu, il faut que l'attribut `method` soit utilisé et qu'il ait la valeur `post`. La soumission du formulaire provoque le démarrage du logiciel de courrier sur le poste du visiteur, et celui-ci doit approuver l'envoi comme celui d'un e-mail habituel. Cette méthode interdit tout traitement informatique des données sur le serveur et nous ne l'utiliserons pas systématiquement ici. Elle peut cependant avoir l'avantage de rendre possible la transmission instantanée d'informations à une personne et ce, quel que soit l'endroit où elle se trouve. Cela n'est pas à négliger pour créer un contact rapide entre un client et un commercial itinérant par exemple, s'il est doté d'un terminal portable pouvant recevoir des e-mails.

- `method` : il détermine la méthode d'envoi des données vers le serveur. Il peut prendre les deux valeurs `get` ou `post`. La méthode `get` est celle qui est utilisée par défaut (si nous ne définissons pas explicitement l'attribut `method`). Elle présente l'inconvénient d'ajouter les données du formulaire sous la forme `nom=valeur` à la suite de l'URL définie dans l'attribut `action`. Si, par exemple, le formulaire contient deux champs de saisie, le premier nommé `prenom` dans lequel est saisie la valeur `jean` et un second nommé `nom` avec la saisie `engels`, l'URL affichée dans la barre d'adresse du navigateur sera la suivante :

```
http://www.funhtml.com/fichier.php?prenom=jean&nom=engels
```

Remarquons que les données sont précédées du point d'interrogation (`?`) et séparées l'une de l'autre par le caractère esperluette (`&`). Les données transmises sont donc visibles par l'utilisateur, mais peuvent aussi être plus facilement interceptées par des personnes mal intentionnées. De plus, la quantité de caractères de l'URL est limitée, ce qui pose un problème pour l'envoi de commentaires longs par exemple. En revanche, il est possible de contourner cet inconvénient en passant directement des données à un script, en les ajoutant à une URL. Elles sont alors traitées par le fichier désigné dans l'URL et peuvent déclencher l'affichage d'une page particulière créée dynamiquement à partir de ces informations. En tapant par exemple l'adresse suivante :

```
http://www.eyrolles.com/Informatique/Recherche/index.php?q=php+5&themes=INF
```

l'utilisateur se trouve sur une page, comme s'il avait effectué une recherche avec le mot-clé `php 5` et la rubrique `INF` dans la page d'accueil du site `www.eyrolles.com`.

La seconde valeur de l'attribut `method` est `post`. Elle ne présente pas l'inconvénient de la méthode `get` car les données transmises sont invisibles dans l'URL, et leur longueur n'est plus limitée. C'est donc celle que nous recommandons dans la plupart des cas.

- `id` : nous y revenons ici car il permet d'identifier le formulaire pour pouvoir accéder à ses composants à partir d'un script JavaScript avec par exemple la méthode `getElementById(id)` que nous avons déjà utilisée.

- enctype : il détermine le type d'encodage de données transmises vers le serveur. Sa valeur par défaut est application/x-www-form-urlencoded elle est applicable dans la plupart des cas, sauf pour le transfert de fichiers du poste client vers le serveur. Dans ce cas, l'attribut doit prendre la valeur multipart/form-data . Quand nous utilisons la valeur mailto: pour l'attribut action, la valeur de enctype doit être text/plain ou text/html, ce qui dépendra de l'encodage de l'e-mail (texte brut ou XHTML).
- accept : il contient un ou plusieurs types MIME de la forme text/html , image/jpeg qui correspond au type des fichiers acceptés pour les opérations de transfert de fichiers vers le serveur. Cela évite d'effectuer un contrôle du type des fichiers transférés côté serveur. Chaque type indiqué doit être séparé du précédent par une virgule. Les fichiers de type non listé dans cet attribut seraient ainsi refusés.
- accept-charset : il contient un ou plusieurs jeux de caractères admis pour les saisies effectuées dans les champs du formulaire. Par défaut, la valeur est la même que celle définie dans la déclaration XML ou l'attribut <meta /> (voir le chapitre 2).
- onsubmit : il permet de gérer l'événement qui survient quand l'utilisateur clique sur le bouton d'envoi des données de type submit (voir l'élément <input />). Il peut être utilisé pour effectuer un contrôle de validité des saisies, par exemple vérifier qu'un code postal comporte cinq chiffres, un numéro de téléphone dix chiffres, ou la syntaxe d'une adresse e-mail.
- onreset : permet de gérer l'événement qui survient quand l'utilisateur efface en bloc toutes les saisies réalisées en cliquant sur le bouton de type reset (voir les sections suivantes).

Les composants communs

Quel que soit le rôle que l'on assigne à un formulaire, questionnaire ou bon de commande par exemple, il doit comporter certains éléments obligatoires et au minimum un bouton d'envoi.

Bouton d'envoi et de réinitialisation

Le bouton d'envoi, généralement nommés Envoi ou OK, est indispensable car il est le seul qui, après un clic, peut déclencher l'envoi des données vers le serveur ou vers un e-mail. La manière la plus courante de créer un bouton d'envoi est d'utiliser l'élément <input/> que nous allons retrouver dans tous les paragraphes suivants car ses effets sont très divers. C'est son attribut type qui permet de lui assigner une fonction particulière. Pour un bouton d'envoi (on dit aussi de soumission du formulaire), l'attribut type prend la valeur submit L'élément <input /> étant vide, c'est le contenu de l'attribut value qui permet de déterminer le texte du bouton. Sa valeur est récupérée côté serveur au même titre que les données du formulaire. Il est possible de créer plusieurs boutons d'envoi et de gérer différemment les données en fonction du bouton choisi. Cet élément possède l'ensemble des attributs communs et nous ne manquerons pas

d'utiliser systématiquement id et title. En supplément de id, nous pouvons définir l'attribut name qui sert également à identifier le bouton d'envoi dans les scripts JavaScript ou PHP, en lui donnant la même valeur. Pour améliorer l'accessibilité, nous pouvons aussi définir les attributs accesskey et tabindex. Le code créant un bouton d'envoi courant peut donc être le suivant :

```
<form>
 <fieldset>
  <input type="submit" value="Envoi" name="soumission" id="soumission" tabindex="5"
   accesskey="E" title="Bouton d'envoi"/>
 </fieldset>
</form>
```

La figure 7-1 (repère ³) montre le résultat obtenu par ce code.

L'effet visuel obtenu est assez spartiate et il sera possible d'ajouter des couleurs de fond et de texte ainsi que des bordures pour ces boutons en écrivant des styles CSS (voir les propriétés color et background-color au chapitre 10 et border au chapitre 11). Pour créer un bouton d'envoi, il est également possible de recourir à l'élément <button>. Pour ce faire, on définit son attribut type avec la valeur submit. Son attribut value contient la valeur associée au bouton. Comme l'élément <input />, il possède les attributs name accesskey tabindex. Le texte visible sur le bouton est le contenu de l'élément et non pas la valeur de l'attribut value. Il peut également s'agir d'une image, ce qui enrichit beaucoup les possibilités graphiques par rapport à l'élément <input />. Pour créer ce type de bouton d'envoi, nous écrivons le code suivant :

```
<fieldset>
 <button type= "submit" value="Envoi2"  name="soumission2" id="soumission2"
  accesskey="B" title="Bouton d'envoi"> Envoi2
 </button>
</fieldset>
```

La figure 7-1 (repère ·) montre un bouton d'envoi dont le contenu est textuel. Pour créer un bouton utilisant le même élément avec une image pour contenu, nous pouvons écrire le code suivant :

```
<fieldset>
 <button type="submit" value="Envoi3" name="soumission3" id="soumission3"
  accesskey="C" title="Bouton d'envoi">
  <img src="france2.gif" height="50" width="50" alt="France"/>
 </button>
</fieldset>
```

La figure 7-1 (repère ») montre ce type de bouton contenant une image.

Quand ces effets décoratifs sont insuffisants, il est envisageable de créer des boutons d'envoi avec des images. La première des possibilités est fournie par l'élément <input /> dont l'attribut type prend la valeur image

Le fichier image, de type GIF, JPEG ou PNG, doit être référencé avec l'attribut src comme il en va pour une image créée avec l'élément <img /> Le bouton a les dimensions de l'image et celles-ci ne peuvent pas être modifiées avec les attributs de l'élément <input />. Comme il s'agit avant tout d'une image, nous disposons de l'élément alt qui contient un texte de substitution qui s'affiche à la place de l'image si elle n'est pas trouvée dans les navigateurs uniquement textuels. Comme pour les images sensibles au clic et divisées en zones, il est possible d'utiliser une carte définissant ces zones avec des éléments <map> et <area /> Nous devons alors référencer cette carte à l'aide de l'attribut usemap qui contient la valeur de l'attribut id de l'élément <map> précédé du caractère dièse (#). Les attributs id, title, accesskey et tabindex sont également utilisés de la même façon que pour un bouton de type submit S'il est employé, l'attribut value ne fournit plus le texte visible, mais une valeur récupérable par les scripts. Pour créer un bouton à partir d'une image, nous pouvons écrire le code suivant :

```
<form>
 <fieldset>
  <input type="image" src="../images/france.gif" value="Envoi3" name="soumission3"
   ➥ id="soumission3" tabindex="6" accesskey="A" title="Bouton d'envoi image"
   ➥ onclick="submit()"/>
 </fieldset>
</form>
```

La figure 7-1 (repère ¿) montre un exemple de ce type de bouton graphique.

Ce type de bouton graphique a la particularité de permettre la transmission au serveur non seulement des données saisies dans le formulaire, mais aussi des coordonnées du point sur lequel a été effectué le clic dans l'image. Ces coordonnées (exprimées en pixel) sont transmises dans les variables nom.x ou nom_x selon le serveur (pour l'abscisse) et nom.y ou nom_y (pour l'ordonnée) dans lesquelles le préfixe nom représente la valeur de l'attribut name du bouton. L'origine des coordonnées est le sommet supérieur gauche de l'image.

On associe souvent à un bouton d'envoi un bouton de réinitialisation. Il n'efface pas le contenu de toutes les zones de saisies mais remet le formulaire dans son état initial, y compris les valeurs par défaut qui ont pu y être définies. Un tel bouton est le plus souvent créé à l'aide de l'élément <input /> dont l'attribut type a la valeur reset. Comme les boutons d'envoi, c'est l'attribut value qui contient le texte visible sur le bouton. Les autres attributs sont identiques à ceux des boutons d'envoi. Nous pouvons créer un bouton d'envoi avec le code suivant :

```
<form>
 <fieldset>
  <input type="reset" value="Effacement" name="efface"/>
 <fieldset/>
<form/>
```

L'utilisation de l'élément <button> est possible pour créer des boutons de réinitialisation en définissant son attribut type grâce à la valeur reset. En incluant une image comme pour les boutons d'envoi précédents, nous pouvons aussi créer des boutons graphiques. Cependant, l'élément <input /> avec un attribut type ayant la valeur image ne peut pas être utilisé pour créer un bouton de réinitialisation.

L'exemple 7-2 résume la création de ces différents types de boutons.

Exemple 7-2. Création de boutons d'envoi et de réinitialisation

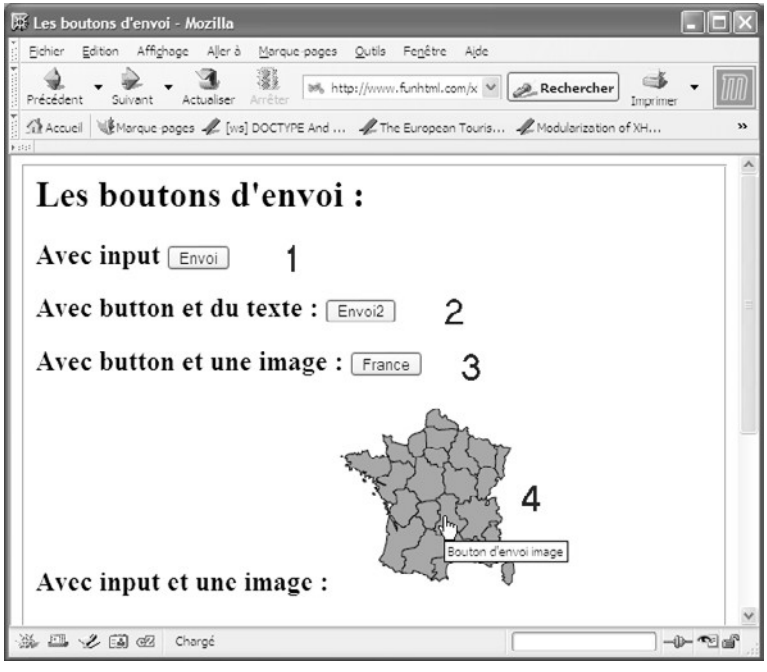
```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les boutons d'envoi</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <form action="exemple7-2post.php" method="post" name="formul1">
   <fieldset>
    <h1>Les boutons d'envoi : </h1>
    <h2>Avec input <input type="submit" value="Envoi" name="soumission" id=
     ➥ "soumission" tabindex="5" accesskey="E" title="Bouton d'envoi" /></h2>
    <h2>Avec button et du texte : <button type= "submit" value="Envoi2"
     ➥ name="soumission2" id="soumission2" accesskey="B" title="Bouton d'envoi">
     ➥ Envoi2</button>
    </h2>
    <h2>Avec button et une image :
     <button type="submit" value="Envoi3"  name="soumission3" id="soumission3"
      ➥ accesskey="C" title="Bouton d'envoi">
     <img src= "../images/france2.gif " height= "50" width= "50" alt= "France"/>
     </button>  »
    </h2>
    <h2>Avec input et une image : <input type="image" src="../images/
     ➥ france.gif"value="Envoi4" name="soumission4" id="soumission4" tabindex="6"
     ➥ accesskey="D" title="Bouton d'envoi image" onclick="submit()"/>¿
    </h2>
    <h1>Les boutons de réinitialisation : </h1>
    <h2>Avec input : <input type="reset" value="Effacement" name="efface"/></h2>
    <h2>Avec button et du texte : <button type= "reset" value="efface2"
     ➥ name="efface2" id="efface2" accesskey="G" title="Bouton d'effacement">
     ➥ Effacement</button></h2>
   <h2>Avec button et une image : <button type= "reset" value="efface3"
     ➥ name="efface3" id="efface3" accesskey="H" title="Bouton d'effacement">
```

```
    <img src= "../images/france2.gif " height= "50" width= "50" alt= "France"/>
    </button> ¶
    </h2>
    </fieldset>
  </form>
 </body>
</html>
```

La figure 7-1 représente tous les types de boutons d'envoi, les boutons de réinitialisation ayant le même aspect.

Figure 7-1

Les boutons textuels et graphiques



Les composants de saisie de texte

Le texte constitue le plus souvent la part la plus importante des saisies effectuées dans un formulaire. Selon les besoins, il peut s'agir d'un texte court tenant sur une ligne, mais il est également possible d'envisager la saisie de textes beaucoup plus longs.

La saisie de texte uniligne

Un formulaire permet le plus souvent la saisie de texte, par exemple pour indiquer son nom ou son adresse e-mail. Chacun a pu rencontrer ce type de composants, ne serait-ce que pour entrer un mot-clé dans un moteur de recherche comme Google. Comme dans

l'ancestral langage Basic, dans lequel une entrée de l'utilisateur se fait par l'instruction input, en XHTML un grand nombre de champs de saisie sont créés avec l'élément <input />. C'est son attribut type qui détermine la catégorie de champ qui est obtenue. Pour un champ de saisie de texte ne comprenant qu'une ligne, l'attribut type prend la valeur text. Comme il est nécessaire de préciser à l'utilisateur le type d'information attendue, il faut introduire le libellé de l'information dans un élément <label>. Ce dernier peut contenir à son tour d'autres éléments qui peuvent permettre de structurer son contenu. Le tableau 7-4 présente la liste de ces éléments enfants.

Tableau 7-4. Les éléments enfants de <label>

Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var

Pour avertir l'internaute qu'il doit saisir son nom, nous écrivons par exemple :

```
<form>
 <label>Votre nom :</label><input type= "text" />
<form>
```

L'élément <input /> possède l'ensemble des attributs communs. Nous pouvons en améliorer la gestion en utilisant ses attributs particuliers.

- name="texte" : il attribue un nom à la zone de saisie, ce qui permet de récupérer dans une variable la valeur saisie sur le serveur. En utilisant PHP pour récupérer les données saisies dans le formulaire, si l'attribut name a la valeur nom le nom indiqué par l'internaute est contenu dans la variable $_POST["nom"] si l'attribut method de l'élément <form> a la valeur post ou dans la variable $_GET["nom"] s'il a la valeur get. En JavaScript, la valeur saisie est accessible dans la variable document.forms[1].nom.value si la page ne contient qu'un seul formulaire (remplacez forms[1] par forms[2] pour le deuxième formulaire éventuel, et ainsi de suite).
- size="N" : il permet de fixer la longueur visible de la zone de texte à N caractères, ce qui n'empêche pas des saisies plus longues.
- maxlength="N" : il permet de limiter le texte saisi à N caractères. Au-delà de ce nombre, les frappes effectuées au clavier sont inopérantes. Cet attribut peut permettre de mettre en adéquation la longueur d'une donnée avec celle du champ d'une base de données dans laquelle elle doit être enregistrée.
- value="texte" : il définit un texte par défaut qui est affiché dans la zone de texte tant que l'utilisateur n'en a pas saisi un autre. C'est cette valeur qui est transmise au serveur si l'internaute ne modifie rien dans le champ texte. Comme dans l'exemple ci-après :

```
<input type="text" name="pays" maxlenght="25" value="France" />
```

Si rien n'est changé dans la zone, c'est la valeur France qui est envoyée au serveur.

Effacement de la valeur par défaut

Pour des raisons d'ergonomie, il est préférable que le texte par défaut défini à l'aide de l'attribut value s'efface tout seul au moment où l'utilisateur clique dessus car cela lui évite d'avoir à le faire lui-même.

Il suffit pour cela d'utiliser une instruction JavaScript très simple :

Pour réagir à l'événement clic :

```
<input type="text" name="prenom" value="Votre prénom" maxlength="25"
➥ onclick="this.value='"' /> />
```

Pour que le texte s'efface dès que la zone reçoit le focus (au moyen de la touche de tabulation par exemple) :

```
<input type="text" name="adresse" value="Votre adresse" maxlength="60"
➥ onfocus="this.value=" " />
```

- disabled="disabled" : cet attribut, qui prend la valeur booléenne unique disabled, rend la zone de saisie inactive, empêchant ainsi toute saisie.
- readonly="readonly" permet d'utiliser une zone de saisie pour afficher une information. Celle-ci ne peut donc pas être modifiée, mais est en lecture seule.
- onchange="Script" : ce gestionnaire d'événements permet de déclencher un script JavaScript quand la valeur par défaut contenue dans l'attribut value est modifiée. Le code est exécuté soit quand l'utilisateur passe à une autre zone de saisie, soit lors de l'envoi. Nous avons par exemple le code suivant :

```
<input type="text" name="pays" value="Votre pays" maxlength="20"
➥ onchange="alert('Modification opérée')" />
```

Comme pour les boutons d'envoi, il est possible d'utiliser les attributs accesskey, tabindex, pour améliorer l'accessibilité, et onfocus ou onblur pour gérer l'attribution ou la perte du focus à la zone de texte. L'exemple 7-3 crée plusieurs zones de saisie de texte (repères ´ , ¶ et ¾). Elles sont toutes précédées d'un libellé contenu dans un élément <label> (repères · , ¿ , ² et º). Le formulaire se termine comme il se doit par un bouton d'envoi (repère µ). Le traitement des données, réalisé en PHP, est confié dans l'attribut action au fichier externe nommé exemple 7-4.php dont le code est présenté plus bas. Il permet simplement d'afficher dans une nouvelle page, l'ensemble des données saisies après l'envoi du formulaire.

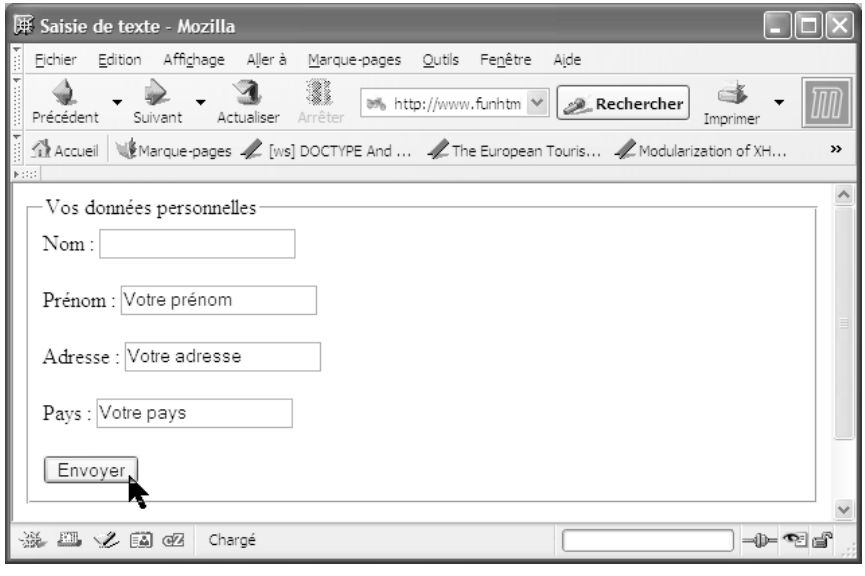
Exemple 7-3. Les champs de saisie de texte

```
 <?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
 <title>Saisie de texte</title>
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
```

```
</head>
<body>
 <form method="post" action="exemple7-4.php"³>
 <fieldset>
  <legend>Vos données personnelles</legend>
   <label>Nom : </label> ·  <input type="text" name="nom"  maxlength="25"  />
    ➥ <br /><br /> »
   <label>Prénom : </label> ¿  <input type="text" name="prenom" value="Votre prénom"
    ➥ maxlength="25" onclick="this.value='“' /><br /><br />
   <label>Adresse : </label> ²  <input type="text" name="adresse" value=
    ➥ "Votre adresse" maxlength="60" onfocus="this.value='“' /><br /><br /> ¶
   <label>Pays : </label> °  <input type="text" name="pays" value="Votre pays"
    ➥ maxlength="20"  onchange="alert('Modification opérée')" /><br /><br /> ¾
   <input type="submit" name="envoi" value="Envoyer"/>↵
  </fieldset>
 </form>
</body>
</html>
```

La figure 7-2 montre l'aspect visuel du formulaire obtenu. Remarquons que sa présentation n'est pas du meilleur effet, en raison des longueurs variables des contenus des éléments <label> qui entraînent le décalage horizontal des zones de texte. Mais une fois de plus, il ne s'agit que de créer une structure, et c'est bien ce qui est réalisé, et notre travail en XHTML est donc bien fait. Pour améliorer cet aspect visuel, nous pourrons utiliser un tableau comme nous le verrons à la fin de ce chapitre ou encore appliquer des styles CSS aux différents éléments (voir le chapitre 14).

Figure 7-2
Les zones de saisie de texte uniligne



Le code de l'exemple 7-4 permet simplement l'affichage du nom de la variable correspondant à l'attribut `name` de chaque composant, et la valeur que lui a donnée le visiteur. Il permet aussi de réaliser des tests si vous disposez d'un serveur PHP.

Exemple 7-4. Le code PHP de traitement des données

```
<?php
echo "<h1>Lecture des données </h1>";
foreach($_POST as $cle=>$valeur)
{
  echo "$cle : $valeur <br />";
}
?>
```

La saisie de mot de passe

Les champs de saisie de mot de passe sont quasi identiques aux champs de saisie de texte. Ils ne comportent qu'une seule ligne et sont créés avec le même élément `<input />`. La différenciation entre ces deux champs réside dans la valeur de l'attribut `type` qui prend la valeur `password` au lieu de `text`. Pour l'utilisateur le champ a le même aspect visuel, mais quand il tape son mot de passe, les caractères qu'il utilise ne sont pas affichés dans la zone et sont remplacés par un astérisque (`*`), ce qui le protège des regards indiscrets. On retrouve ce type de champ dans tous les sites d'accès réservé, comme les pages d'accueil des webmails qui permettent la lecture des courriers électroniques, sans faire appel à un logiciel comme Outlook ou Mozilla. Les attributs sont les mêmes que pour un champ de texte. Nous définissons de préférence les attributs `size="N"` pour limiter la taille de la zone visible à N caractères et `maxlenght="N"` pour limiter le mot de passe à N caractères. Notez encore la différence entre ces deux attributs car `size` n'empêche pas la saisie d'un nombre supérieur de caractères à la taille visible du champ alors que `maxlenght` bloque la saisie à exactement N caractères, les frappes suivantes étant ignorées quelle que soit la largeur du champ. Dans les navigateurs modernes, la soumission d'un formulaire contenant au moins un champ de saisie de mot de passe entraîne l'apparition d'un message d'alerte demandant au visiteur s'il veut mémoriser son mot de passe pour ne pas avoir à le retaper lors d'une prochaine visite sur le même site. La figure 7-3 montre le modèle de fenêtre obtenue dans Mozilla.

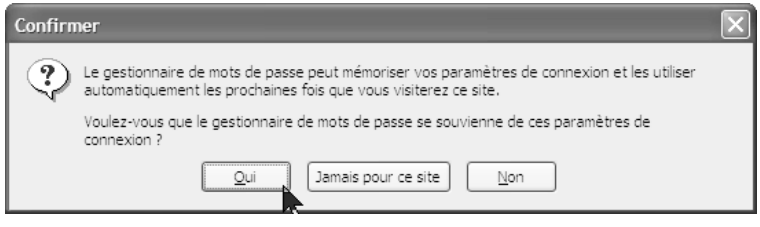


Figure 7-3
Enregistrement de mot de passe par le navigateur

L'exemple 7-5 crée un formulaire contenant simplement un champ de texte pour saisir un login (repère ·) et un champ password (repère ¿) pour le mot de passe, chacun étant précédé d'un label explicatif (repères ³ et »).

Exemple 7-5. Saisie de mot de passe

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
 <title>Saisie de texte</title>
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <form method="post" action="exemple7-4.php" >
   <fieldset>
    <legend><big>Accès réservé: donner vos login et mot de passe</big></legend>
    <label> Login : </label>  ³
    <input type="text" name="nom"  maxlength="8"  />·
    <label> Mot de passe : </label>  »
    <input type="password" name="code"  maxlength="8" /><br /¿
    <input type="submit"  value="Envoyer"/>
   </fieldset>
  </form>
 </body>
</html>
```

La figure 7-4 montre le formulaire obtenu.

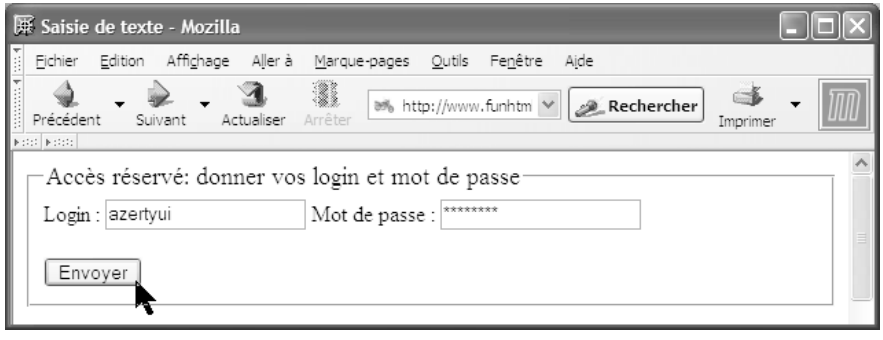


Figure 7-4
Un formulaire de saisie de mot de passe

La saisie de texte long

L'élément <input /> ne permet la saisie que sur une seule ligne avec peu de visibilité pour les textes longs. Pour permettre à un visiteur la saisie de textes beaucoup plus longs, comme des commentaires élogieux sur votre site par exemple, ou d'une manière plus actuelle dans les blogs, nous disposons d'un élément spécialisé. Comme l'élément <input /> doté de l'attribut type="text", l'élément <textarea> crée un champ de saisie de texte sur plusieurs lignes. Contrairement au précédent, ce n'est pas un élément vide et son contenu n'est autre que le texte saisi par le visiteur. Il est possible de déterminer un contenu pour l'élément dans le code de la page XHTML. Ce texte sera alors visible dans la zone de saisie et permettra de donner une information à l'utilisateur.

Les dimensions de la zone de saisie doivent obligatoirement être définies. Pour cela, il faut utiliser les attributs suivants :

- cols="N" qui fixe la largeur de la zone à N caractères. Le retour à la ligne est automatique dans la zone.
- rows="N" qui fixe la hauteur à N lignes. Il faut distinguer la hauteur visible et le nombre de lignes que l'on peut saisir. En effet, le visiteur peut écrire autant de lignes qu'il le souhaite, quelle que soit la hauteur visible de la zone. Quand le texte dépasse la capacité de la zone, une barre de défilement vertical apparaît automatiquement. L'attribut name s'il n'est pas réellement obligatoire, est fortement conseillé pour pouvoir identifier le champ côté serveur et lire son contenu dans une variable (en PHP, dans les variables $_POST["nom-du-champ"] ou $_GET["nom-du-champ"] selon la méthode d'envoi des données).

Comme pour les zones de saisie uniligne, il est recommandé d'effacer automatiquement le contenu par défaut de l'élément au moyen d'un script JavaScript en réponse à l'événement onfocus en écrivant le code suivant :

```
onfocus="this.value=''"
```

L'attribut onblur est également associé à l'élément pour gérer la perte du focus.

En plus des attributs communs, l'élément <textarea> possède les attributs gestionnaires d'événements particuliers, à savoir onselect et onchange. Le premier permet de gérer l'événement qui survient quand l'utilisateur sélectionne une partie du texte contenu dans l'élément. Le second correspond à l'événement survenant quand le texte contenu dans l'élément a été modifié. Cependant, pour gérer cet événement, il faut noter qu'il ne se produit réellement que quand la zone a perdu le focus et pas à l'instant même où son contenu est modifié. Pour rendre plus aisée l'accessibilité, nous n'oublierons pas d'utiliser systématiquement les attributs accesskey et tabindex. L'élément <textarea> possède enfin les attributs disabled et readonly dont la définition avec les valeurs respectives disabled et readonly rend le champ texte inactif et le transforme en une simple zone d'affichage d'informations. Cette faculté est souvent utilisée pour afficher les termes d'un contrat ou les conditions de vente sur un site de e-commerce.

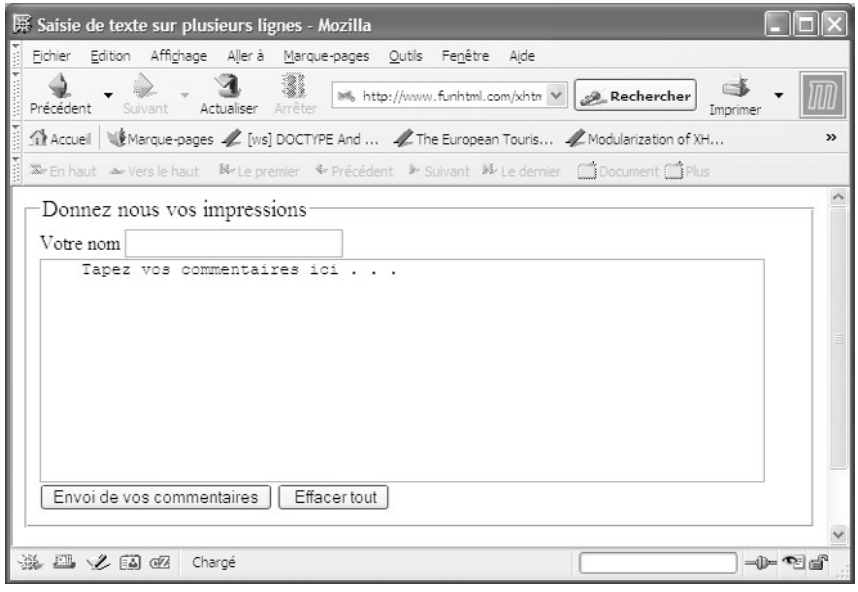
Le code de l'exemple 7-6 crée un formulaire complet de saisie permettant à un visiteur de faire des commentaires après s'être identifié avec un login. Le premier champ est une zone de texte uniligne dans laquelle il doit saisir son nom (repère ³). Le deuxième est créé à l'aide de l'élément <textarea> et permet la saisie des commentaires proprement dits (repère ¿). Ses dimensions sont fixées à 70 caractères en largeur et 10 lignes en hauteur (repères · et »). Enfin, nous retrouvons les indispensables boutons submit et reset (repères ´ et ²). Chaque composant est précédé d'un élément <label> de structuration de l'information au lieu d'un simple texte, qui permettra d'améliorer la présentation du formulaire à l'aide de CSS appliquées séparément aux éléments <label> d'une part et <input /> ou <textarea> d'autre part. À titre indicatif, le code de l'exemple 7-4, étudié précédemment, contient le code PHP permettant la récupération des valeurs du formulaire, et les afficher dans une nouvelle page.

Exemple 7-6. Création de zone de saisie de texte sur plusieurs lignes

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Saisie de texte sur plusieurs lignes</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <form action="exemple7-4.php" method="post">
   <fieldset>
    <legend><big>Donnez-nous vos impressions</big></legend>
    <label>Votre nom </label><input type="text" name="nom" size="25"
     ➥ tabindex="1" /><br /> ³
    <label>Vos commentaires</label>
    <textarea name="commentaires" cols="70" rows="10"» onfocus="this.value=''"
     ➥ tabindex="2">¿
    Tapez vos commentaires ici…
    </textarea><br />
    <input type="submit" value="Envoi de vos commentaires" />´
    <input type="reset" value=" Effacer tout" /><br /> ²
   </fieldset>
  </form>
 </body>
</html>
```

La figure 7-5 montre le formulaire obtenu.

Figure 7-5
Formulaire de saisie de données textuelles



Les boutons radio et les cases à cocher

Les zones de texte permettent à l'internaute de saisir du texte, ce qui offre une infinité de réponses possibles. Cela correspond bien à la saisie du nom ou de l'adresse e-mail. Pour les données dont les réponses sont prévisibles et en nombre limité, nous pouvons inclure des éléments de formulaire spéciaux, nommés boutons radio et cases à cocher. Un bouton radio se présente sous la forme d'un cercle dans lequel apparaît un disque noir quand l'utilisateur clique dessus pour effectuer un choix. Les boutons radio sont typiquement utilisés pour présenter plusieurs choix dont la réponse est unique (par exemple, un choix entre un bouton libellé Homme et un bouton libellé Femme). Pour créer un bouton radio, nous utilisons encore une fois l'élément `<input />` avec un attribut `type` qui prend la valeur `radio`. L'ensemble des boutons radio avec lesquels on peut opérer un choix donné constitue un groupe. Il faut que tous ses éléments aient la même valeur pour leurs attributs `name`. L'attribut `value` de chaque bouton radio contient la valeur que l'on veut associer à chacun d'eux. Seule la valeur choisie dans le même groupe est récupérée par le serveur. L'attribut `checked` qui prend la valeur booléenne unique `checked` permet de cocher par défaut un des boutons d'un groupe, par exemple, s'il est plus répandu que les autres. Rien n'empêche cependant le visiteur de modifier le bouton indiquant la valeur proposée. En revanche, l'attribut `readonly` bloque le bouton radio et impose ainsi une valeur. Son utilité ne peut se limiter à rappeler une valeur obligatoire. L'attribut `disabled` dont la valeur unique est `disabled` permet de rendre un bouton radio inactif. Nous pouvons envisager de l'utiliser pour désactiver un choix réalisé antérieurement. En tant qu'élément pouvant recevoir le focus, un bouton radio possède bien sûr les attributs gestionnaires d'événements `onfocus` (quand le bouton reçoit le focus) et `onblur` (quand il le perd).

Pour améliorer l'accessibilité, nous ne négligeons pas, pour notre part, les attributs `title`, `accesskey` et `tabindex`. Le code de création d'un groupe de boutons radio est le suivant :

```
<form action="exemple7-4.php" method="post">
 <fieldset>
  <label>Monsieur</label> <input type="radio" name="sexe"  value="Monsieur"
   checked="checked" />
  <label>Madame</label> <input type="radio" name="sexe"  value="Madame" />
 </fieldset>
</form>
```

Pour les questions qui n'appellent qu'une réponse unique, comme dans l'exemple précédent sur le sexe de la personne, les boutons radio sont les composants idéaux. En revanche, si vous réalisez un questionnaire sur les goûts musicaux de vos visiteurs, rien ne les empêche d'aimer à la fois le rock, le classique et la chanson française ou d'autres encore (si ! c'est possible). Les cases à cocher représentent alors la meilleure solution.

Elles se présentent graphiquement sous la forme de petits carrés dans lesquels apparaît une coche en V si l'utilisateur du formulaire choisit la valeur associée à la case. Le fonctionnement paraît identique aux boutons radio à la différence que les cases à cocher ne font pas partie d'un groupe, et qu'il est possible d'opérer plusieurs choix dans des cases différentes sans que le choix précédent se trouve décoché. Dans ce cas, il faut créer autant de cases à cocher que de choix suggérés. Une case à cocher est encore créée à l'aide de l'élément `<input />` dont l'attribut `type` prend cette fois la valeur `checkbox`. Les attributs `name` de chacune des cases se doivent alors de porter des noms différents. C'est encore l'attribut `value`, dont la définition est ici indispensable, qui contient la valeur associée à chaque case, qui sera récupérée côté serveur après l'envoi du formulaire, uniquement si la case est cochée. L'élément utilisé étant vide, il faut l'associer à un élément `<label>` pour préciser le choix demandé. Le nom de chaque élément étant différent, l'attribut `value` associé à chacun d'entre eux peut avoir une valeur de type booléen de la forme « oui » ou « non » car seules les valeurs des cases choisies sont transmises au serveur. Les attributs `checked`, `disabled`, `title`, `accesskey` et `tabindex` qui ont le même rôle que les boutons radio peuvent être utilisés dans les mêmes conditions. Les gestionnaires d'événements `onfocus`, `onblur`, `onchange` et `onselect` sont aussi utilisables de la même façon. Le code de création d'une case à cocher peut donc être le suivant :

```
<label>Classique : </label>
<input type="checkbox" name="classique" tabindex="1" accesskey="G" value="oui" />
```

L'exemple 7-7 permet de mettre en œuvre ces composants pour obtenir des renseignements divers de la part d'un visiteur. Il contient un premier groupe de boutons radio permettant de déterminer la civilité de la personne (repères ³, · et »). Il est complété par une zone de saisie de texte pour le nom de la personne (repère ¿). Le second groupe permet d'illustrer par une image chaque bouton radio dans un élément `<label>` (repères ´, ² et ¶). Un paragraphe contient ensuite un ensemble de cinq cases à cocher pour offrir un choix de goûts musicaux (repères º, ¾, µ, ¸ et ¹). Chaque case a un nom différent et la valeur `oui` qui lui est associée ne sera récupérée côté serveur que si le visiteur la

coche. Le formulaire est complété par les boutons d'envoi et de réinitialisation (repères et).

Exemple 7-7. Boutons radio et cases à cocher

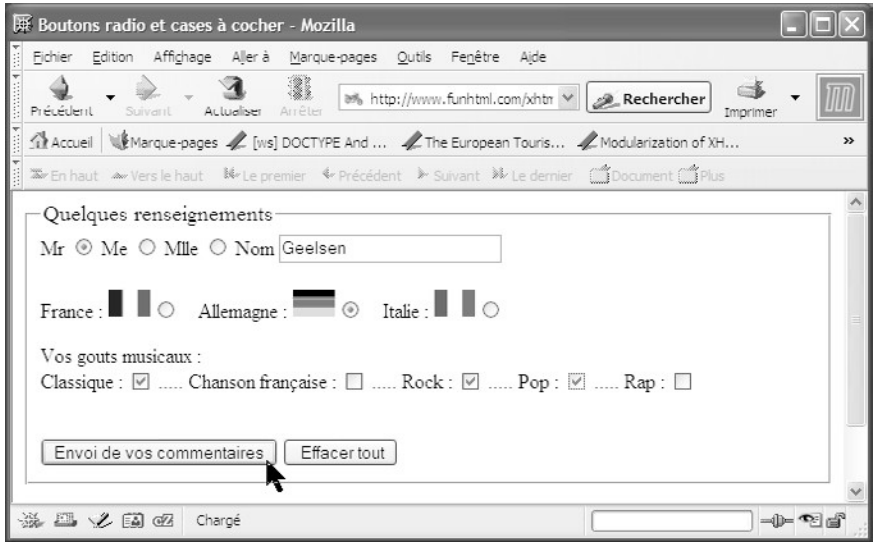
```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Saisie de texte sur plusieurs lignes</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <form action="exemple7-4.php" method="post">
   <fieldset>
    <legend><big>Quelques renseignements </big></legend>
    <!--Civilité-->
    <label>Mr</label> <input type="radio" name="sexe"  value="Monsieur"
     ➥ checked="checked" tabindex="1" /> 
    <label>Me</label> <input type="radio" name="sexe"  value="Madame"
     ➥ tabindex="2" /> ·
    <label>Mlle</label> <input type="radio" name="sexe"  value="Mademoiselle"
     ➥ tabindex="3" /> »
    <!--Nom  -->
    <label>Nom </label><input type="text" name="nom" size="25" tabindex="4" />
     ➥ <br /><br /> ¿
    <!-- Pays  -->
    <label>France : <img src="../images/drapeaufr.gif" alt="France" /></label><input
     ➥ type="radio" name="pays"  tabindex="5"  accesskey="A" />   
    <label>Allemagne : <img src="../images/drapeaud.gif" alt="Allemagne" />
     ➥ </label><input type="radio" name="pays"  tabindex="6"  accesskey="B" />
     ➥   
    <label>Italie : <img src="../images/drapeauit.gif" alt="Italie" /></label><input
     ➥ type="radio" name="pays"  tabindex="7"  accesskey="C" />
    <!-- Goûts -->
    <p>Vos goûts musicaux : <br />
    <label>Classique : </label><input type="checkbox" name="classique"  tabindex="8"
     ➥ accesskey="D" /> 
    <label> ..... Chanson française : </label><input type="checkbox" name="chanson"
     ➥  tabindex="9"  accesskey="E" /> 
    <label> ..... Rock : </label><input type="checkbox" name="rock"  tabindex="10"
     ➥ accesskey="G" /> 
    <label> ..... Pop : </label><input type="checkbox" name="pop"  tabindex="11"
     ➥ accesskey="H" /> 
    <label> ..... Rap : </label><input type="checkbox" name="rap"  tabindex="12"
     ➥ accesskey="K" /> 
    </p>
    <br /><input type="submit" value="Envoi de vos commentaires" />
```

```
    <input type="reset" value=" Effacer tout" /><br />
  </fieldset>
 </form>
</body>
</html>
```

La figure 7-6 illustre le résultat obtenu pour ce formulaire.

Figure 7-6

Les boutons radio et les cases à cocher



Les listes de sélection

Un autre moyen de faciliter la saisie de données par un visiteur de votre site consiste à lui proposer d'effectuer un ou plusieurs choix parmi une liste de sélection déroulante pouvant contenir un grand nombre d'éléments. Ces listes peuvent être utilisées quand l'éventail des réponses est prévisible, comme une liste de pays ou de types de carte bancaire. Elles sont connues de tous car on les rencontre dans tous les logiciels bureautiques pour sélectionner l'extension sous laquelle on veut enregistrer un fichier. La valeur associée à chaque option étant contenue dans le code XHTML, cette méthode a l'avantage d'éviter les erreurs d'orthographe que pourraient faire les différents utilisateurs. Si les informations sont stockées dans une base de données, cela évitera de stocker des valeurs différentes alors qu'elles correspondent en réalité à la même information. Une liste de sélection est créée avec l'élément `<select>` En soi, cet élément n'entraîne aucun rendu visuel, il n'est que le conteneur de la liste. Il doit donc inclure ensuite autant d'éléments `<option>` qu'il y a de choix proposés au visiteur. La structure XHTML d'une liste de sélection prend alors la forme suivante :

```
<select name="pays" size="1">
  <option value="France"> France</option>
```

```
  <option value="Belgique"> Belgique</option>
  <option value="Suisse"> Suisse</option>
  <option value="Canada"> Canada</option>
</select>
```

Le choix de la première option est important car il définit la valeur par défaut qui sera transmise au serveur si elle n'est pas modifiée. L'élément `<select>` possède l'ensemble des attributs communs parmi lesquels nous retiendrons principalement `id` et `class`, d'usage courant. Il possède également les attributs propres suivants :

- `name="texte"` : il permet, comme pour les autres éléments de formulaire, de récupérer la ou les données choisies par le visiteur dans une variable qui est identifiée par ce nom.
- `size="Nombre"` : il définit le nombre de lignes d'options qui sont visibles lors de l'affichage de la liste. Par défaut, seule la première option définie dans le code est visible. Les autres ne le sont que si l'utilisateur déroule la liste en cliquant dessus.
- `multiple="multiple"` : il prend la valeur booléenne unique `multiple` pour indiquer que l'utilisateur peut opérer plusieurs choix simultanément dans la liste. Pour ce faire, ce dernier est obligé de maintenir la touche Ctrl enfoncée pour que le second choix n'annule pas le premier. Cette faculté n'étant pas très connue de la majorité des internautes, je conseille de l'éviter, sans compter qu'elle correspond peu à l'idée que se fait la plupart des usagers d'une liste déroulante. Si son usage s'avère utile, il faut bien signaler la procédure à suivre. Quand cet attribut est défini et que l'on utilise un serveur PHP, les valeurs choisies sont récupérées dans un tableau, et il faut pour toutes les récupérer que le nom de la liste soit suivi de crochets ouvrant et fermant (par exemple : `name="nom[]"`
- `tabindex="Nombre"` : pour attribuer un ordre de tabulation à la liste et la rendre active avec la touche de tabulation.
- `disabled="disabled"` : pour rendre la liste inactive. Aucun choix n'est plus alors possible.
- `onfocus="script"` et `onblur="script"` : pour gérer respectivement les événements produits par l'attribution et la perte du focus à la liste.
- `onchange="script"` : pour gérer l'événement qui survient quand la valeur sélectionnée par défaut est modifiée par l'utilisateur.

Nous avons déjà signalé que le choix de la première option de la liste est important. En effet, si nous n'utilisons pas l'attribut `multiple`, elle est la seule visible tant que l'on ne déroule pas la liste, et surtout elle constitue la valeur par défaut qui est transmise au serveur si le visiteur n'effectue pas un autre choix. Pour cette raison, je conseille d'attribuer à la première option la valeur `null` et un message du type tel que « Faites un choix », pour bien signaler qu'un choix volontaire est nécessaire. La valeur `null` permet en effet une gestion simple de l'absence de réponse côté serveur. Cela est particulièrement utile si l'éventail `<select>` contient une liste de pays ou de départements dans un formulaire de saisie d'adresse. Le texte visible dans la liste de sélection est le contenu de l'élément `<option>` Il peut être plus ou moins explicite, par exemple « Livre en français » dans le formulaire de recherche d'une librairie en ligne. En revanche, la valeur associée à

l'option se doit d'être plus courte ou même codée, par exemple `fr` pour le même type de site. Cette valeur contenue dans l'attribut `value` est invisible pour l'utilisateur et c'est elle qui sera récupérée par le serveur. En plus de l'ensemble des attributs communs, l'élément `<option>` possède l'attribut `selected` qui prend la valeur booléenne unique `selected` et avec lequel on peut définir une valeur par défaut qui peut être autre que celle de la première option de la liste. L'attribut `disabled="disabled"` permet comme pour les autres éléments des formulaires de rendre une option inactive, et donc non sélectionnable. L'utilisation de cet attribut ne présente d'intérêt que pour désactiver une option en réaction à une action du visiteur à l'aide d'un script JavaScript. On pourrait l'utiliser par exemple dans un questionnaire de jeu pour lequel l'internaute n'a droit qu'à un seul essai. S'il clique sur une option, son choix est entériné par la définition de l'attribut `disabled`. Nous aurions dans ce cas le code suivant :

```
onclick="this.disabled='disabled'"
```

Dans l'exemple 7-8, nous créons un formulaire contenant plusieurs listes de sélection d'options. La première liste (repère ³) ne permet qu'un seul choix parmi quatre valeurs (repères » , ¿ , ´ et ²), la première option n'étant créée qu'à titre informatif (repère ·). La liste est affichée initialement sur une seule ligne (car l'attribut `size` vaut 1) (repère ³). La seconde liste (repère ¶) permet plusieurs choix simultanés (l'attribut `multiple` est défini) parmi quatre options (repères º , ¾ , µ et ¸). La première option est choisie par défaut en définissant l'attribut `selected` (repère º). Le formulaire est bien sûr complété par un bouton d'envoi (repère ¹) et un bouton de réinitialisation (repère). Le script PHP contenu dans l'exemple 7-4 permet d'afficher les choix effectués dans une nouvelle page.

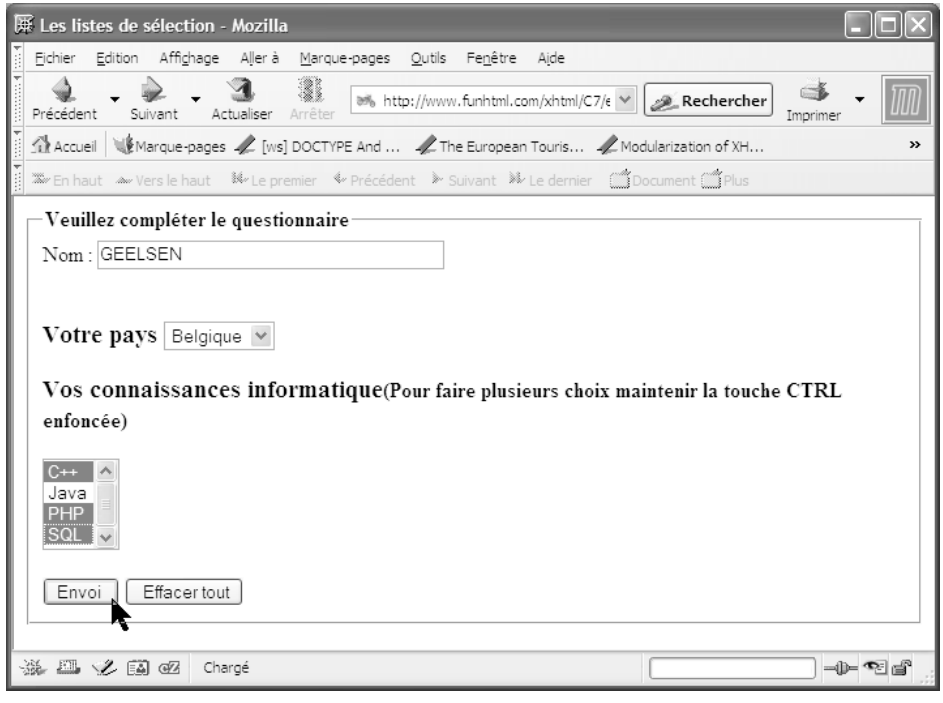
Exemple 7-8. Les listes de sélection

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les listes de sélection</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <form action="exemple7-4.php" method="post"
   enctype="application/x-www-form-urlencoded">
  <fieldset>
   <legend><b>Veuillez compléter le questionnaire</b></legend>
    <label>Nom : </label><input type="text" name="nom" size="40" maxlength="256"
     value="votre nom" onclick="this.value='' tabindex="1"/><br /><br />
    <!-- Liste à choix unique -->
    <h3>Votre pays
    <select name="pays" size="1" tabindex="2">
      <option value="null" disabled="disabled"> Votre pays</option>
      <option value="France" > France</option>
```

```
    <option value="Belgique" > Belgique</option>
    <option value="Suisse" > Suisse</option>
    <option value="Canada" > Canada</option>
  </select></h3>
  <!-- Liste à choix multiples -->
  <h3>Vos connaissances informatique <small>(Pour faire plusieurs choix maintenir
    ➥ la touche CTRL enfoncée)</small></h3>
  <select name="savoir[]" size="4" multiple="multiple" tabindex="3">
    <option value="C++" selected="selected"> C++</option>
    <option value="Java"> Java</option>
    <option value="PHP" onclick="this.disabled='disabled'"> PHP</option>
    <option value="SQL"> SQL</option>
  </select>
  <br /><br />
  <input type="submit" value="Envoi " tabindex="4"/>
  <input type="reset" value=" Effacer tout" tabindex="5"/><br />
 </fieldset>
 </form>
  </body>
</html>
```

La figure 7-7 montre le résultat obtenu.

Figure 7-7
Les listes de sélections



Plutôt que de définir les éléments <option> les uns à la suite des autres, ce qui donne un aspect uniforme à la liste une fois déroulée, il est possible de créer des groupes d'options et de leur attribuer un titre qui fait ressortir les groupements effectués pour structurer la liste, en particulier quand elle contient un grand nombre d'éléments. Pour réaliser ces groupes, il faut faire intervenir l'élément <optgroup> dans l'élément <select>. Les options de chaque groupe sont incluses entre les balises <optgroup> et </optgroup>. À l'intérieur d'un élément <select>, on peut inclure autant de groupes que l'on veut. Le libellé de chaque groupe est donné dans l'attribut label de l'élément <optgroup> qui, outre les attributs communs habituels, possède également l'attribut disabled, déjà rencontré pour les autres composants et qui rend le groupe inactif. Le code de création d'un groupe a donc la structure suivante :

```
<select name="pays" size="1" tabindex="2">
<optgroup label="Europe">
  <option value="France" > France</option>
  <option value="Belgique" > Belgique</option>
</optgroup>
</select>
```

L'exemple 7-9 permet d'illustrer la création de groupes d'options. À titre de démonstration, nous utiliserons ici le protocole mailto : pour l'envoi des données vers l'adresse e-mail indiquée dans l'attribut action (repère ³). Un champ de texte permet la saisie du nom (repère ·). L'élément <select> gère l'événement onfocus dont le code permet de changer la couleur du texte, ainsi que onchange qui affiche un message d'alerte quand un choix a été fait (repère »). Cet élément contient d'abord un élément <option> situé en dehors de tout groupe et dont le rôle est de constituer la valeur par défaut null qui permet de vérifier si l'utilisateur a bien effectué un choix (repère ¿). Le reste de l'élément <select> est constitué de trois groupes d'options (repères ´ , µ et) dont l'attribut label constitue le titre du groupe. Le premier groupe (repère ´) contient quatre options correspondant à des pays d'Europe (repères ² , ¶ , º et ¾). Le second (repère µ) contient la liste de trois pays d'Amérique (repères ¸ , ¹ et). Le troisième groupe (repère) permet aux visiteurs de choisir un continent si leur pays ne figure pas dans les options précédentes. Il contient la liste des continents (repères , , et).

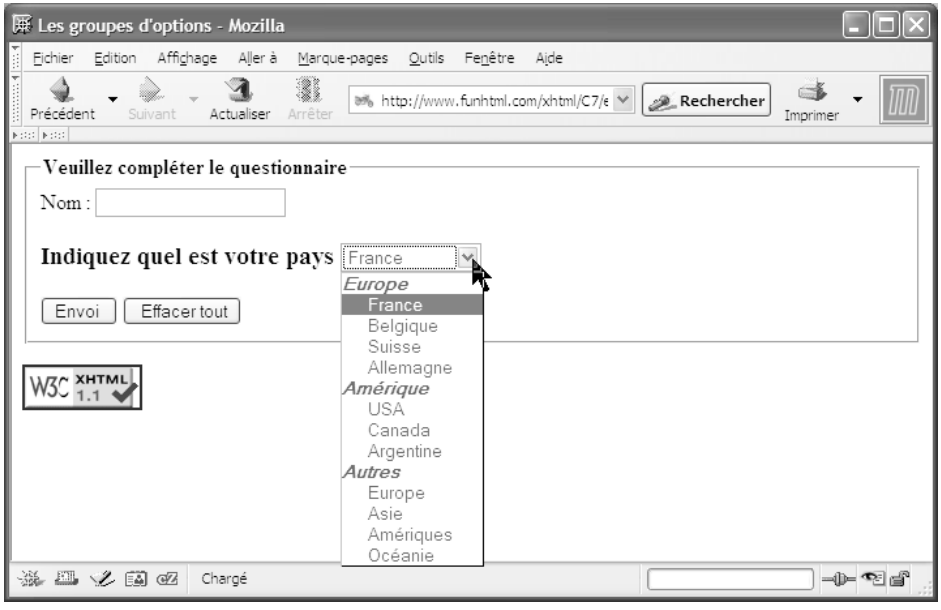
Exemple 7-9. Les groupes d'options

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les groupes d'options</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
```

```
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
<style type="text/css" title="">
</style>
 </head>
 <body>
  <form action="mailto:xhtml@funhtml.com" method="post" >
   <fieldset>
    <legend><b>Veuillez compléter le questionnaire</b></legend>
    <!-- Liste à choix unique -->
    <label>Nom : </label><input type="text" name="nom" tabindex="1"
     accesskey="A"/>
    <h3>Indiquez quel est votre pays
    <select name="pays" size="1" tabindex="2" onfocus="this.style.color='red'"
     onchange="alert('Merci de votre choix')">
     <option value="null" > Votre pays</option>
     <optgroup label="Europe">
      <option value="France" > France</option>
      <option value="Belgique" > Belgique</option>
      <option value="Suisse" > Suisse</option>
      <option value="Allemagne" > Allemagne</option>
     </optgroup>
     <optgroup label="Amérique">
      <option value="USA" > USA </option>
      <option value="Canada" > Canada</option>
      <option value="Argentine" > Argentine</option>
     </optgroup>
     <optgroup label="Autres">
      <option value="Europe" > Europe </option>
      <option value="Asie" > Asie </option>
      <option value="Amériques" > Amériques </option>
      <option value="Océanie" > Océanie</option>
     </optgroup>
    </select></h3>
    <input type="submit" value="Envoi " tabindex="3"/>
    <input type="reset" value=" Effacer tout" tabindex="4"/><br />
   </fieldset>
  </form>
 </body>
</html>
```

La figure 7-8 présente le résultat obtenu après avoir déroulé la liste. Les différents groupes y sont bien visibles.

Figure 7-8
Les groupes d'options



Les champs cachés

Un formulaire peut aussi contenir des champs cachés. Comme leur nom l'indique, ils ne correspondent à aucun rendu visuel et ne font donc l'objet d'aucune saisie de la part du visiteur. Ils permettent de transmettre au serveur des informations particulières. Ils permettent ainsi de recueillir des informations sur le poste client tel que le système d'exploitation, le type de navigateur qu'il utilise et son numéro de version, ceci dans le but de réaliser des statistiques. Ces champs permettent de transmettre tout type d'informations utiles au serveur, par exemple la taille maximale d'un fichier téléchargeable du poste client vers le serveur. Cela évite les abus, ce que nous verrons à la section suivante. Pour créer un champ caché, nous utilisons encore l'élément `<input />` mais avec un attribut `type` auquel il faut attribuer la valeur `hidden`, et bien sûr un attribut `id` et `name` pour identifier le champ et recueillir l'information dans une variable. L'attribut `value` contient la valeur qui sera transmise après l'envoi du formulaire. Pour créer un champ caché, nous pouvons écrire le code suivant :

```
<input type="hidden" id="navigateur" name="navigateur" value="secret" />
```

Quand il s'agit de recueillir des informations sur le poste client, la valeur du champ caché est définie par du code JavaScript. C'est le cas dans l'exemple 7-10 qui contient, outre un champ de texte (repère ») (puisqu'il y faut bien un élément visible) et les habituels

boutons communs (repères ² et ¶), deux champs cachés qui vont permettre de connaître le nom du navigateur (repère ¿) et le système d'exploitation du client (repère ´). La valeur de ces champs est déterminée par du code JavaScript, déclenché lors de l'envoi du formulaire par l'attribut `onsubmit`. Le premier lit le nom du navigateur (repère ³) et le second le nom du système d'exploitation (repère ·). Les données sont ici transmises par le protocole `mailto:` mais le code fonctionnerait de la même façon avec la méthode `post` et un attribut `action` différent.

Exemple 7-10. Les champs cachés

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les champs cachés</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <body>
  <form action="mailto:xhtml@funhtml.com" id="form1" method="post"
   ➥ onsubmit="getElementById('form1').navigateur.value=navigator.appName;
   ➥ getElementById('form1').systeme.value=navigator.platform"   >
   <fieldset>
    <legend><b>Veuillez compléter le questionnaire</b></legend>
    <label>Nom : </label><input type="text" name="nom" tabindex="1"
     ➥ accesskey="A"/»
    <input type="hidden" id="navigateur" name="navigateur" /> ¿
    <input type="hidden" id="systeme" name="systeme" /´>
    <input type="submit" value="Envoi " tabindex="2"/> ²
    <input type="reset" value=" Effacer tout" tabindex="3"/><br />    ¶
   </fieldset>
  </form>
 </body>
</html>
```

Après l'envoi du formulaire, le destinataire de l'e-mail indiqué dans l'attribut `action` reçoit un e-mail dont le contenu a la forme suivante :

```
nom=engels&navigateur=Netscape&systeme=Win32
```

Le transfert de fichiers

Un site interactif peut permettre à ses visiteurs d'effectuer le transfert de divers fichiers du poste client vers le serveur. On trouve cette possibilité par exemple dans un site de petites annonces ou d'enchères en ligne. Le client peut ainsi transférer la photographie de l'objet qu'il met en vente. Cette fonctionnalité est incluse dans un formulaire à l'aide

de l'élément <input /> doté d'un attribut type qui prend cette fois la valeur file . Il crée dans le formulaire un champ composé d'une zone de texte pour taper le chemin d'accès au fichier et d'un bouton de sélection de fichier dont l'intitulé fixe est Sélectionner, qui permet au visiteur de rechercher le fichier à transférer sur son ordinateur. L'aspect type du champ est présenté à la figure 7-9. Quand le visiteur a choisi son fichier, le chemin d'accès complet à ce dernier est automatiquement affiché dans la zone de texte du composant. Pour effectuer le transfert de fichier, l'élément <form> doit utiliser la méthode post et avoir un attribut enctype dont la valeur est explicitement définie de la façon suivante :

```
<form action="exemple7-4.php" method="post" enctype="multipart/form-data">
```

Ce type de composant possède l'ensemble des attributs communs. Il faut y ajouter obligatoirement l'attribut name permettant d'identifier la variable qui contiendra le nom du fichier côté serveur. L'attribut size ="N" permet de limiter la taille visible de la zone de saisie. Pour se prémunir du transfert de fichiers indésirables et assurer un certain degré de sécurité sur le serveur, nous pouvons limiter le transfert à un certain nombre de types de fichiers en définissant l'attribut accept. Il doit contenir la liste des types MIME des fichiers dont le transfert est permis. Pour autoriser plusieurs types, il faut séparer chacun d'entre eux par une virgule. En attribuant à cet attribut une valeur de la forme suivante :

```
<input type="file" name="fichier" accept="image/*" />
```

le serveur accepte le transfert de tous les types d'images.

Pour limiter la taille des fichiers à transférer, nous pouvons saisir celle-ci dans un champ caché dont la valeur est égale au nombre d'octets maximal admis. Le nom de ce champ défini dans son attribut name doit être MAX_FILE_SIZE. La réception des fichiers est gérée en PHP sur le serveur.

Figure 7-9
Le formulaire de transfert de fichier

L'exemple 7-11 contient le code de création d'un formulaire destiné au transfert de fichier d'images dont le format est restreint aux types JPEG ou PNG. Dans l'élément <form> nous définissons les attributs method et enctype spécifiques (repère ³). L'attribut accept de l'élément <input /> permet de limiter les types des fichiers acceptés pour les

images (repère ·). Un champ caché transmet au serveur PHP la taille maximale des fichiers dans la variable nommée MAX_FILE_SIZE (repère »). L'étude de la gestion du transfert côté serveur n'entre pas dans le cadre de cet ouvrage et je vous invite à vous reporter à un ouvrage spécialisé en fonction du langage installé sur votre serveur (PHP, ASP.Net ou autres) pour récupérer les fichiers transmis. À titre indicatif, en PHP, les informations concernant le fichier transféré sont contenues dans la variable de type tableau nommée $_FILES. Notons enfin que le transfert simultané de plusieurs fichiers est parfaitement réalisable en incluant autant d'éléments <input/> munis de l'attribut type="file" qu'il y a de fichiers.

Exemple 7-11. Transfert de fichier

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <title>Transfert de fichier</title>
 </head>
 <body>
  <form action="exemple7-4.php" method="post" enctype="multipart/form-data">
   <fieldset>
    <legend><b>Envoyez-nous votre photo</b></legend>
    <label>Choisissez le fichier JPEG ou PNG : </label><input type="file"
     ➥ name="fichier" accept="image/jpeg,image/png" />
    <input type="hidden" name="MAX_FILE_SIZE" value="50000" /><br /><br />
    <input type="reset" value="Effacer" />   
    <input type="submit" value="Envoyer" />
   </fieldset>
  </form>
 </body>
</html>
```

Un formulaire complet

À titre d'illustration, l'exemple 7-12 récapitule tout ce que nous venons d'étudier. Le document inclut tous les éléments de formulaires possibles. La gestion des données est confiée à un script externe PHP dont le code très simplifié figure dans l'exemple 7-13. Le formulaire contenant un champ de transfert de fichier, son attribut enctype prend la valeur multipart/form-data (repère ³). Il est divisé en trois groupes délimités par des éléments <fieldset> (repères · , et). Le premier groupe permet d'obtenir les coordonnées du visiteur. Il contient trois zones de texte pour la saisie du nom, du prénom et de l'e-mail (repères » , ¿ et ´) et une zone de mot de passe (repères ²).

Ces champs sont suivis d'un groupe de boutons radio pour indiquer le sexe de la personne (repères ¶ et °). Enfin, ce groupe se termine par une liste de sélection (repère¾) qui contient trois groupes d'options (repères µ , ¸ et ¹) pour le choix du pays. Le deuxième groupe (repère) permet de saisir des informations sur les goûts du visiteur. Il comprend trois cases à cocher (repères , et) et une zone de texte multilignes (repère) de 50 caractères de large et de 5 lignes de haut. Le troisième groupe (repère) contient un composant d'envoi de fichier (repère) et un champ caché indiquant la taille maximale des fichiers fixée à 10 Ko (repère). Le formulaire se termine bien évidemment par l'insertion des boutons d'envoi et de réinitialisation (repères © et ©).

Exemple 7-12. Un formulaire complet

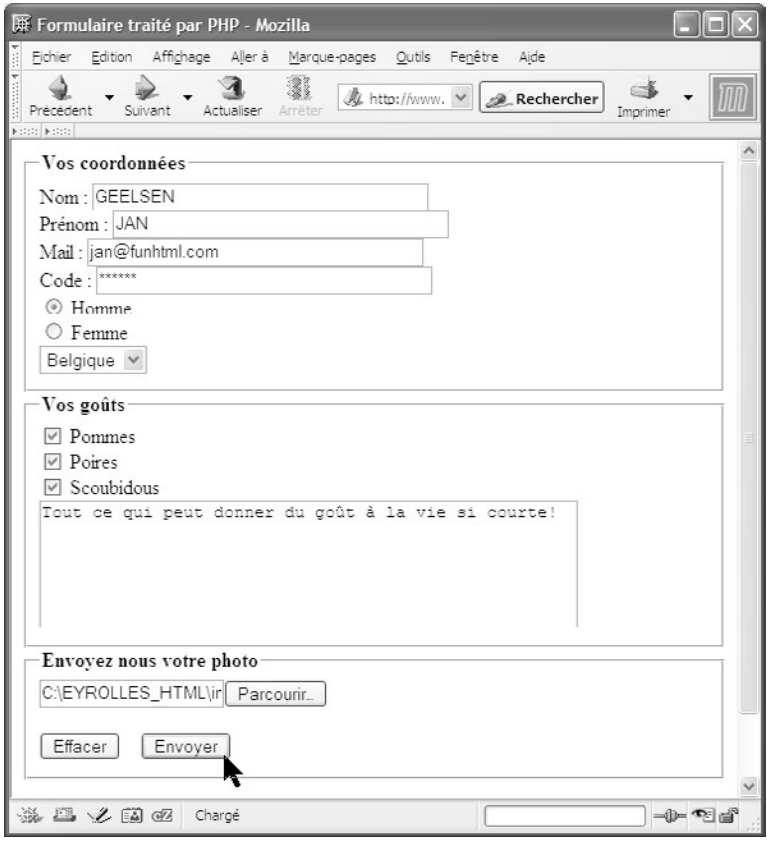
```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <title>Formulaire traité par PHP</title>
 </head>
<body>
 <form action="exemple7-14.php" method="post" enctype="multipart/form-data">
 <!-- Premier groupe de composants-->
  <fieldset>
   <legend><b>Vos coordonnées</b></legend>
   <label>Nom : </label><input type="text" name="nom" size="40" maxlength="256"
    ➥ value="votre nom" /><br />»
   <label>Prénom : </label><input type="text" name="prenom" size="40"
    ➥ maxlength="256" value="votre prénom" /><br />
   <label>Mail : </label><input type="text" name="mail" size="40" maxlength="256"
    ➥ value="votre mail" /><br />
   <label>Code : </label><input type="password" name="code" size="40" maxlength="6"
    ➥ /> <br />
   <input type="radio" name="sexe" value="homme" /><label>Homme</label> <br />
   <input type="radio" name="sexe" value="femme" /><label>Femme</label> <br />
   <select name="pays" size="1" tabindex="2" onfocus="this.style.color='red'"
    ➥ onchange="alert('Merci de votre choix')" onblur="this.style.color='black'">
    <option value="null" > Votre pays</option>
    <optgroup label="Europe">µ
     <option value="France" > France</option>
     <option value="Belgique" > Belgique</option>
     <option value="Italie" > Italie</option>
     <option value="Allemagne" > Allemagne</option>
    </optgroup>
```

```
    <optgroup label="Amérique">
     <option value="USA" label="fr" > USA </option>
     <option value="Canada" > Canada</option>
     <option value="Argentine" > Argentine</option>
    </optgroup>
    <optgroup label="Autres">
     <option value="Europe"  > Europe </option>
     <option value="Asie"  > Asie </option>
     <option value="Amériques"  > Amériques </option>
     <option value="Océanie" > Océanie</option>
    </optgroup>
   </select>
   <br />
  </fieldset>
  <!-- Deuxième groupe de composants-->
  <fieldset>
   <legend><b>Vos goûts</b></legend>
    <input type="checkbox" name="pomme" value="pomme" /> Pommes<br />
    <input type="checkbox" name="poire" value="poire" /> Poires<br />
    <input type="checkbox" name="scoubidou" value="scoubidou" /> Scoubidous<br />
    <textarea name="gouts" cols="50" rows="5" onclick="this.value=‘‘">
      Décrivez vos goûts en détail
    </textarea>
    <br />
  </fieldset>
  <!-- Troisième groupe de composants-->
  <fieldset>
    <legend><b>Envoyez nous votre photo</b></legend>
    <input type="file" name="fichier" accept="image/jpeg" />
    <input type="hidden" name="MAX_FILE_SIZE" value="10000" />
    <br /><br />
    <input type="reset" value="Effacer" />
    &amp;nbsp;&amp;nbsp;&amp;nbsp;<input type="submit" value="Envoyer" />
    <br />
   </fieldset>
  </form>
 </body>
</html>
```

La figure 7-10 montre l'aspect de ce formulaire.

Pour permettre aux lecteurs disposant d'un serveur local ou distant muni de PHP de tester ce formulaire, l'exemple 7-13 désigné par l'attribut `action` du formulaire, fournit le code minimal afin de vérifier le bon envoi des données. Le serveur peut ainsi afficher dans une nouvelle page toutes les données qu'il a reçues.

Figure 7-10
Un formulaire contenant tous les composants



Exemple 7-13. Le code PHP de traitement des données

```
<?php
echo "<h1>Lecture des données </h1>";
foreach($_POST as $cle=>$valeur)
{
  echo "$cle : $valeur <br />";
}
echo "<h1>Informations sur le fichier transféré</h1>";
foreach($_FILES as $tab)
{
 foreach($tab as $cle=>$valeur)
 {
   echo "$cle : $valeur <br />";
 }
}
?>
```

Organisation des formulaires à l'aide de tableaux

L'aspect même d'un formulaire tel qu'il est visible à la figure 7-10 montre des similitudes avec la structure d'un tableau. De plus, à moins d'avoir recours à des artifices comme la multiplication des caractères d'espace en écrivant l'entité ` ` il est difficile d'obtenir un bon alignement des composants, tels que les libellés `<label>` et les champs de saisie de texte par exemple. Pour cette raison, l'utilisation d'un tableau se prête bien à la structuration des éléments d'un formulaire. L'exemple 7-14 en donne une illustration classique. Les composants du formulaire étant les mêmes que ceux de l'exemple 7-13, nous ne détaillerons plus leurs rôles. Notons simplement l'organisation de ces éléments. Dans le cas présent, nous avons choisi de conserver les trois groupes distincts créés avec l'élément `<fieldset>`. Chacun d'entre eux (repères ³ , ´ et ¾) contient donc un tableau spécifique (repères · , ² et µ) pour lequel nous définissons deux groupes contenant chacun une colonne dont la largeur est fixée à 10 % de celle du tableau pour la colonne de gauche et à 70 % pour celle de droite (repères» et ¿ , ¶ et º , ¸ et ¹). Chaque ligne est composée de deux cellules. La première contient le libellé créé avec `<label>`, inclus dans la première colonne de gauche, et le composant du formulaire dans un élément inclus dans celle de droite. Pour les éléments de la colonne de gauche contenant les libellés, nous définissons l'attribut `align` avec la valeur `right` (repères » , ¶ et ¸).

Exemple 7-14. Association d'un tableau et d'un formulaire

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <title>Formulaire traité par PHP</title>
 </head>
 <body>
  <form action="exemple7-14.php" method="post" enctype="multipart/form-data">
  <!-- Premier groupe de composants-->
   <fieldset>
    <legend><b>Vos coordonnées</b></legend>
    <table border="0" width="100%">
     <colgroup width="10%" align="right"></colgroup>
     <colgroup width="70%"></colgroup>
     <tr>
   <td><label>Nom : </label></td>
   <td><input type="text" name="nom" size="40" maxlength="256"
     value="votre nom" /></td>
     </tr>
     <tr>
      <td><label>Prénom : </label></td>
      <td><input type="text" name="prenom" size="40" maxlength="256"
          value="votre prénom" /></td>
```

```
</tr>
<tr>
 <td><label>Mail : </label></td>
 <td><input type="text" name="mail" size="40" maxlength="256"
    ➥ value="votre mail" /></td>
</tr>
<tr>
 <td><label>Code : </label></td>
 <td> <input type="password" name="code" size="40" maxlength="6" /></td>
</tr>
<tr>
 <td><label>Homme : </label></td>
 <td><input type="radio" name="sexe" value="homme" /></td>
</tr>
<tr>
 <td>Femme : </td>
 <td><input type="radio" name="sexe" value="femme" /></td>
</tr>
<tr>
 <td>Votre pays : </td>
 <td>
  <select name="pays" size="1" tabindex="2" onfocus="this.style.color='red'"
     ➥ onchange="alert('Merci de votre choix')"
     ➥ onblur="this.style.color='black'">
   <option value="null" > Votre pays</option>
   <optgroup label="Europe">
    <option value="France" > France</option>
    <option value="Belgique" > Belgique</option>
    <option value="Italie" > Italie</option>
    <option value="Allemagne" > Allemagne</option>
   </optgroup>
   <optgroup label="Amérique">
    <option value="USA" label="fr" > USA </option>
    <option value="Canada" > Canada</option>
    <option value="Argentine" > Argentine</option>
   </optgroup>
   <optgroup label="Autres">
    <option value="Europe"  > Europe </option>
    <option value="Asie"  > Asie </option>
    <option value="Amériques"  > Amériques </option>
    <option value="Océanie" > Océanie</option>
   </optgroup>
  </select>
 </td>
</tr>
</table>
</fieldset>
<!-- Deuxième groupe de composants-->
```

```
 <fieldset>
  <legend><b>Vos goûts</b></legend>
  <table border="0" width="100%" >
   <colgroup width="10%" align="right"></colgroup>
   <colgroup width="70%"></colgroup>
   <tr>
    <td><label>Pommes : </label> </td>
    <td><input type="checkbox" name="pomme" value="pomme" /> </td>
   </tr>
   <tr>
    <td><label>Poires : </label> </td>
    <td><input type="checkbox" name="poire" value="poire" /></td>
   </tr>
   <tr>
    <td><label>Scoubidous : </label></td>
    <td><input type="checkbox" name="scoubidou" value="scoubidou" /></td>
   </tr>
   <tr>
    <td><label>Décrivez nous vos goûts en détail : </label></td>
    <td>
     <textarea name="gouts" cols="50" rows="5" onclick="this.value='‘'">
    Exprimez vous ici...
     </textarea>
    </td>
   </tr>
  </table>
 </fieldset>
 <!-- Troisième groupe de composants-->
 <fieldset>
  <legend><b>Envoyez-nous votre photo</b></legend>
  <table border="0" width="100%" align="right">
   <colgroup width="10%" align="right"></colgroup>
   <colgroup width="70%"></colgroup>
   <tr>
    <td><label>Choisir le fichier : </label></td>
    <td><input type="file" name="fichier" accept="image/jpeg" />
      ➥ <input type="hidden" name="MAX_FILE_SIZE" value="10000" /></td>
   </tr>
   <tr>
    <td> <input type="submit" value=" Envoyer " /></td>
    <td><input type="reset" value="Effacer toutes les données" /></td>
   </tr>
  </table>
 </fieldset>
</form>
</body>
</html>
```

L'aspect obtenu, visible à la figure 7-11, est bien plus présentable que le précédent. Nous reprendrons le même formulaire au chapitre 14 en lui appliquant des styles CSS pour montrer les améliorations qu'il est possible de réaliser au niveau de la présentation. Même si l'utilisation de tableau n'a ici pour but que d'améliorer la présentation, elle ne constitue pas une entorse au principe de séparation du contenu et de l'aspect car le formulaire a toujours une structure logique. L'usage des tableaux n'est dans ce cas nullement abusif et ne nuit pas à la structuration de la page, bien au contraire. Il permet de faire d'une pierre deux coups. Nous verrons dans la deuxième partie de cet ouvrage qu'il est possible de se passer des tableaux dans ce cas, en créant des styles pour les éléments <label> et les différents composants du formulaire <input /> ou <textarea> mais cela nécessitera un codage supplémentaire. Selon certains avis, la solution la plus pragmatique consiste à associer les formulaires inclus dans des tableaux avec des styles CSS pour en améliorer encore l'aspect.

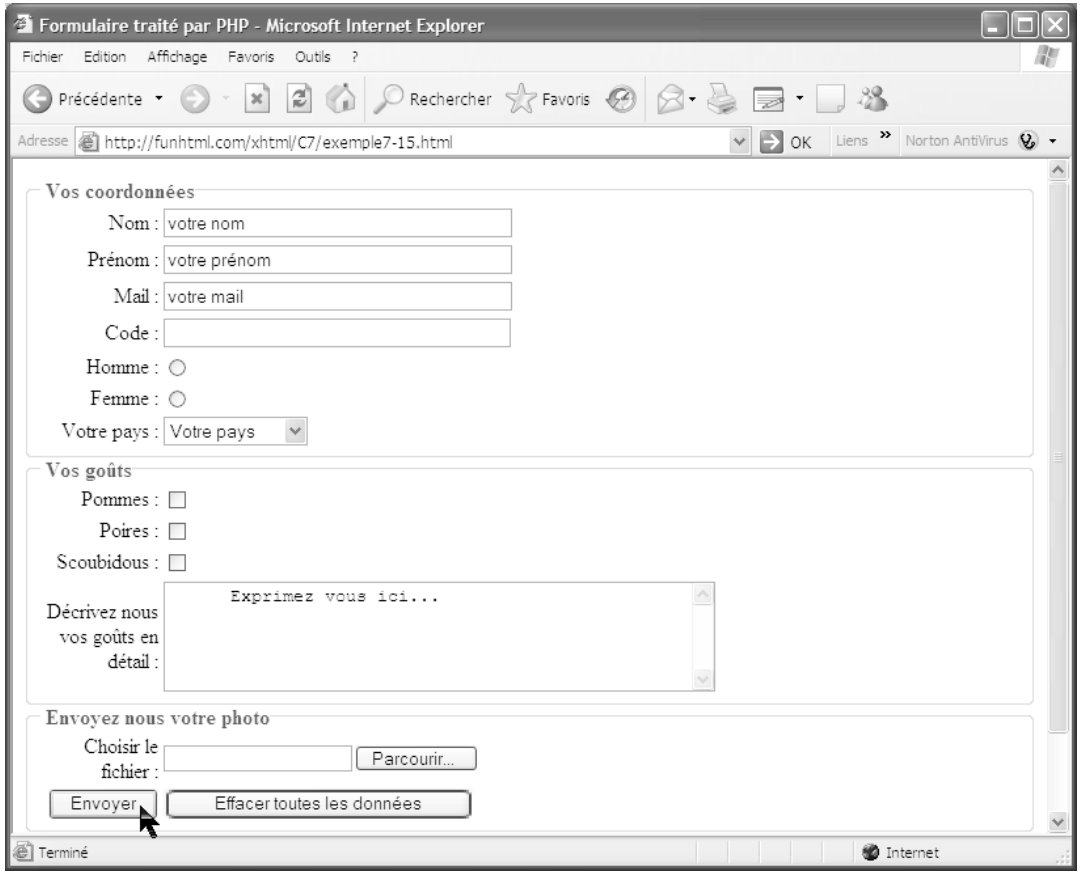


Figure 7-11

Inclusion des composants de formulaire dans un tableau

Exercices

Exercice 1 : L'attribut `action` de l'élément `<form>` est-il obligatoire ? Quel est son rôle ?

Exercice 2 : Quelles sont les différentes applications qui correspondent aux valeurs de l'attribut `enctype` dans l'élément `<form>` ?

Exercice 3 : Créez un bouton d'envoi avec une image.

Exercice 4 : Avec quels éléments peut-on créer des champs de saisie de texte ?

Exercice 5 : Quelles sont les différences entre les méthodes `post` et `get` ?

Exercice 6 : Créez un formulaire permettant la saisie, dans un seul groupe, d'un nom avec un maximum de 25 caractères, d'un mot de passe de huit caractères et d'un e-mail de 40 caractères. Ajoutez des boutons d'envoi et de réinitialisation.

Exercice 7 : Créez un formulaire de saisie de texte de 15 lignes et une largeur de 60 caractères. Ajoutez des boutons d'envoi et de réinitialisation.

Exercice 8 : Créez deux groupes de boutons radio. Le premier pour le choix de la nationalité avec les choix Français, Communauté européenne et Autres, et le second pour saisir le diplôme obtenu le plus élevé avec les choix Licence, Master et Doctorat. Saisir également le nom et l'e-mail. Ajoutez des boutons d'envoi et de réinitialisation.

Exercice 9 : Créez un formulaire comprenant un seul groupe de champs ayant pour titre Adresse client, et qui contient cinq zones de saisie de texte dont les libellés sont nom, prénom, adresse, ville et code postal. Ajoutez un bouton d'envoi.

Exercice 10 : Créez un questionnaire de qualification pour recruter un webmestre, comprenant les cases à cocher avec les libellés suivants : XHTML, CSS, PHP, JavaScript et MySQL. Saisissez le nom et le téléphone.

Exercice 11 : Dans une optique similaire à celle de l'exercice 10, proposez les mêmes choix avec une liste de sélection à choix multiples.

Exercice 12 : Créez une liste de sélection à choix unique permettant de saisir le département du domicile. Les départements sont regroupés par régions (ne créez que trois régions). Les noms des régions doivent apparaître dans la liste.

Exercice 13 : Créez un formulaire de transfert de fichiers destinés à l'envoi de document compressés aux formats ZIP ou TAR vers le serveur.

Exercice 14 : Créez un formulaire complet dont le but est de permettre la saisie de tous les éléments constitutifs d'un CV pour le recrutement d'un informaticien web (nom, prénom, adresse…) en utilisant le maximum de composants de formulaire appropriés en fonction des besoins. Créez obligatoirement plusieurs groupes de composants. Utilisez l'exemple 7-4 pour vérifier la bonne réception des données par le serveur.

8

Créer des cadres

L'utilisation des cadres pour la création de pages web est, disons-le d'emblée, une technique considérée comme obsolète depuis déjà plusieurs années. Le W3C a cependant maintenu cette possibilité dans ses recommandations jusqu'à la version XHTML 1.0 et l'a définitivement supprimée dans la DTD XHTML 1.1. Pour ne pas être plus royaliste que le roi W3C qui admet encore cette possibilité, j'ai cependant tenu à évoquer ce sujet pour ceux qui seraient encore des inconditionnels de cette méthode utilisée en HTML, même après avoir envisagé quels sont les moyens de se passer (en grande partie) de ce mode d'organisation, comme nous le ferons en utilisant des styles CSS associés à un document XHTML 1.1 au chapitre 13. Mais qu'est-ce, au juste, qu'une page avec des cadres ? Les cadres constituent un type d'organisation bien particulier car, comme leur nom l'indique, ils ne font que définir un découpage de la page en zones précises, alors que le document XHTML qui les crée ne possède aucun contenu visible dans le navigateur si ce n'est un éventuel avertissement pour les visiteurs dont le navigateur est configuré pour refuser les cadres. À chaque cadre est associé un document XHMTL habituel dont le contenu s'affiche dans le cadre en question et lui seul, indépendamment du contenu du cadre qui lui est adjacent. Cela peut présenter un avantage pour actualiser une partie donnée de la page sans avoir à réactualiser l'ensemble. Si nous imaginons une page divisée et composée de trois cadres, ce n'est pas un document XHTML que nous allons afficher mais trois documents distincts créés séparément dans trois fichiers différents. La caractéristique des pages avec cadres est l'indépendance, a priori totale, de chaque cadre vis-à-vis des autres. C'est un peu comme si la page était occupée par plusieurs fenêtres du même navigateur. Nous avons cependant la possibilité de créer des interactions entre chacun d'eux à l'aide de scripts JavaScript et c'est sûrement cette faculté qui a séduit plus d'un webmestre. La figure 8-1 est une capture d'écran du site http://www.grdfp.polymtl.ca qui utilise des cadres. Il s'agit d'une application très classique des pages avec cadres car le cadre de gauche contient le menu du site et permet la navigation parmi

toutes les pages du site. Les pages cibles des liens du menu sont affichées dans le cadre de droite sans jamais cacher le cadre de navigation. On peut remarquer d'emblée que chaque cadre possède sa propre barre de défilement vertical indépendante. Quoique séduisants pour l'organisation d'une page, les cadres ne sont pas sans inconvénients. Le fait que le contenu qu'ils affichent provienne d'autres documents XHTML indépendants pose des problèmes d'indexation aux moteurs de recherche qui référencent séparément la page créant les cadres et celles qu'elle contient. Un internaute qui effectue une recherche peut très bien obtenir comme résultat une des pages du site sans obtenir la page initiale qui contient les cadres. Il ne possède plus alors le système de navigation inclus dans un autre document, ce qui peut rendre le site incompréhensible. Le référencement d'un site étant essentiel pour la notoriété d'un site, il s'agit là d'un inconvénient majeur.

Nous allons voir successivement quelle doit être la structure des pages divisées en cadres, puis la manière de créer ces divisions, qu'elles soient horizontales, verticales ou encore plus complexes en mariant ces deux possibilités. Enfin, nous envisagerons la manière de faire communiquer les cadres entre eux.

Figure 8-1
Un exemple de site avec cadres

Structure des pages avec cadres

L'utilisation des cadres n'étant pas permise en XHTML 1.1, il faut recourir à XHTML 1.0 qui admet cette création dans une DTD spéciale nommée « Frameset » (voir le site du

W3C : http://www.w3.org/TR/xhtml1/#a_dtd_XHTML-1.0-Frameset). La déclaration DOCTYPE doit donc être modifiée par rapport à celle des documents créés jusqu'à présent et nous écrirons dans tout ce chapitre la déclaration suivante :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

N'ayant pas de contenu propre, une page avec cadre présente une différence fondamentale avec un document XHTML 1.1, à savoir que l'élément `<body>` n'apparaît pas dans l'élément `<html>`. De plus, son code est généralement très court car il ne contient que la définition des cadres et aucun contenu visible dans la page. Pour créer les cadres de la page, l'élément `<body>` est remplacé par les éléments `<frameset>` et `<frame />` qui permettent la définition des cadres.

Les éléments <frameset> et <frame>

L'élément `<frameset>` permet la définition des divisions de la page en différents cadres, dont les dimensions ou les proportions sont définies par ses attributs `rows` et `cols` sur lesquels nous allons revenir.

Cet élément inclut ensuite autant d'éléments `<frame />` que le concepteur a défini de cadres différents. Comme pour les tableaux, l'élément `<frameset>` peut également contenir d'autres `<frameset>` ce qui permet la création de cadres imbriqués dont l'utilisation risque de rendre la structuration de la page difficile à comprendre. En prévision d'éventuels navigateurs qui ne géreraient pas les cadres, ou d'une désactivation par l'utilisateur de cette gestion, il faut proposer un contenu de remplacement pour ne pas désappointer les visiteurs. Ce contenu est inclus dans un élément `<noframes>` qui contient à son tour un élément `<body>` et ses éléments enfants habituels.

L'élément `<frameset>` possède, comme la plupart des autres, les attributs `id`, `class` et `title` dont l'usage nous est désormais connu. Il possède en plus les attributs suivants qui permettent la division de la page en plusieurs cadres :

- `rows` : il permet de diviser la page en cadres horizontaux et contient une suite de dimensions séparées par des virgules. Ces dimensions peuvent être données en pixel, en pourcentage ou à l'aide de coefficients indiquant des proportions.
- `cols` : il permet la division de la page en cadres verticaux et admet les mêmes types de valeurs que le précédent.

Nous mettrons en œuvre ces attributs dans les sections suivantes.

L'élément `<frameset>` possède également, comme l'élément `<body>` qu'il remplace, les attributs de gestion d'événement `onload` et `onunload` qui correspondent respectivement aux événements survenant quand la page se trouve chargée ou fermée. Ils permettent de déclencher des scripts JavaScript à l'arrivée et au départ d'un visiteur. Il peut s'agir par exemple de vérifier l'existence d'un cookie sur le poste client lors du démarrage ou d'afficher un message en fin de connexion.

L'élément <frame /> est un élément vide et, comme l'élément <img />, il définit principalement un document cible à afficher dans le cadre qu'il crée. Ce sont donc ces attributs qui jouent un rôle prépondérant. En plus des attributs class, id et title habituels, il possède les attributs suivants :

- name : il permet, comme id, d'attribuer un nom au cadre, ce qui est indispensable pour gérer les pages et choisir dans quel cadre va s'afficher la cible d'un lien par exemple. Le nom ne doit pas commencer par le caractère de soulignement mais par une lettre.
- longdesc : cet attribut facilite l'accessibilité des sites en donnant l'URL de la description détaillée du contenu d'un cadre.
- src : il est essentiel car il définit l'URL du document XHTML qui va s'afficher dans le cadre.
- frameborder : il définit si le cadre a une bordure (valeur 1) ou s'il n'en a pas (valeur 0). Quand elle est bien gérée par les navigateurs, l'utilisation de la valeur 0 permet de supprimer le désagréable effet visuel de délimitation des cadres.
- marginwidth : il définit la largeur de la marge entre le bord gauche du cadre et son contenu. Sa valeur est donnée en pixel ou en pourcentage de la largeur totale du cadre.
- marginheight : il définit la hauteur de la marge entre le bord supérieur du cadre et son contenu. Comme le précédent, il est défini en pixel ou en pourcentage de la hauteur totale du cadre. La hauteur du cadre dépendant de son contenu, il peut être hasardeux de définir une hauteur de marge en pourcentage.
- noresize : il prend la valeur booléenne unique noresize pour interdire au visiteur de redimensionner le cadre. Cela permet de préserver les proportions choisies par le créateur du site. Pour autoriser le redimensionnement, il suffit d'omettre cet attribut.
- scrolling : les cadres étant plus petits que la fenêtre du navigateur, les documents qu'ils affichent sont généralement plus grands. Dans ce cas, des barres de défilements horizontal et vertical peuvent apparaître sur les bords droit et bas du cadre. Pour gérer explicitement l'apparition de ces barres de défilement, l'attribut scrolling peut prendre les valeurs yes (barres toujours présentes), no (jamais de barres quel que soit le contenu du cadre) ou auto (apparition des barres si nécessaire, la gestion étant effectuée par le navigateur en fonction du contenu). Pour utiliser la valeur no, il faut être sûr que le contenu soit entièrement intégrable dans le cadre, sinon une partie seulement est visible.

Pour les navigateurs qui ne supporteraient pas les cadres, il est possible d'afficher un contenu de remplacement en l'incluant dans l'élément <noframes>. Il doit être inclus dans l'élément <frameset> et doit contenir à son tour un élément <body> qui inclut les éléments XHTML créant la page de remplacement. Ce contenu peut être du même type que celui d'une page normale mais il ne s'affichera qu'en cas d'impossibilité de gestion des cadres.

Nous allons maintenant envisager la réalisation de différents types de pages avec cadres.

Les cadres horizontaux

Le premier cas de création de cadres consiste à partager la page en plusieurs cadres horizontaux. Pour cela, il suffit de définir l'attribut `rows` de l'élément `<frameset>` de la manière suivante :

```
<frameset rows="100,400,200">
  <frame src="http://www.funhtml.com/pagehaut.html" />
  <frame src=" http://www.funhtml.com/pagecentre.html" />
  <frame src=" http://www.funhtml.com/pagebas.html" />
  <noframes>
   <body>
    <p>Votre navigateur ne supporte pas les cadres!<br /> Voir la <a id="lien"
      ➥ href="http://www.funhtml.com/index.html" tabindex="1" accesskey="A"
      ➥ title="Version sans cadres">version sans cadres</a>
    </p>
   </body>
  </noframes>
</frameset>
```

Dans ce cas, le cadre supérieur a une hauteur de 100 pixels, le suivant de 400 pixels et celui du bas de 200 pixels. Si l'utilisateur a un écran de 800 ¥ 600 pixels, il ne verra le cadre du bas que partiellement ou pas du tout selon la configuration du navigateur. Il est également possible de définir les dimensions d'une partie seulement des cadres et de laisser le navigateur attribuer le reste de la hauteur en fonction des dimensions de l'écran du visiteur.

En écrivant par exemple le code suivant :

```
<frameset rows = "100,*,200">
  <frame src="http://www.funhtml.com/pagehaut.html" />
  <frame src=" http://www.funhtml.com/pagecentre.html" />
  <frame src=" http://www.funhtml.com/pagebas.html" />
  <noframes>
   <body>
    <p>Votre navigateur ne supporte pas les cadres!<br /> Voir la <a id="lien"
      ➥ href="http://www.funhtml.com/index.html" tabindex="1" accesskey="A"
      ➥ title="Version sans cadres">version sans cadres</a>
    </p>
   </body>
  </noframes>
</frameset>
```

nous créons également trois cadres horizontaux, le premier de 100 pixels de haut, le dernier de 200 pixels de haut, et le cadre central a une hauteur variable, selon la définition de l'écran du poste client, laquelle se définit à l'aide du caractère `*` (ce cadre a ici 300 pixels de haut dans un écran de 800¥ 600, et 468 pixels sur un écran de 1024¥ 768).

À cette hauteur du cadre central, il faut enlever l'espace variable occupé éventuellement par les barres de menus et d'icônes des navigateurs. Comme c'est souvent le cas, la définition des hauteurs en pourcentage préserve les proportions dessinées par le créateur de la page. Nous pouvons par exemple définir des dimensions relatives suivantes :

```
<frameset rows="15%,75%,10%">
  <frame src="http://www.funhtml.com/pagehaut.html" />
  <frame src=" http://www.funhtml.com/pagecentre.html" />
  <frame src=" http://www.funhtml.com/pagebas.html" />
  <noframes>
   <body> <a id="lien" href="http://www.funhtml.com/index.html" tabindex="1"
     ➥ accesskey="A" title="Version sans cadres">version sans cadres</a>
    </p>
   </body>
  </noframes>
</frameset>
```

Dans ce cas, le partage de la hauteur disponible de l'écran se fait toujours selon la proportion 15 % en haut, 75 % au centre et 10 % en bas, quel que soit l'écran de visualisation de la page.

Il est aussi possible de mélanger ces deux dernières définitions en utilisant une taille absolue en pixel pour le cadre supérieur qui affiche par exemple un bandeau publicitaire de hauteur fixe, et une ou plusieurs tailles relatives en pourcentage pour les autres cadres. Si nous définissons par exemple le code suivant :

```
<frameset rows = "120, 75%, *">
  <frame src="http://www.funhtml.com/pagehaut.html" />
  <frame src=" http://www.funhtml.com/pagecentre.html" />
  <frame src=" http://www.funhtml.com/pagebas.html" />
  <noframes>
   <body>
    <p>Votre navigateur ne supporte pas les cadres!<br /> Voir la <a id="lien"
     ➥ href="http://www.funhtml.com/index.html" tabindex="1" accesskey="A"
     ➥ title="Version sans cadres">version sans cadres</a>
    </p>
   </body>
  </noframes>
</frameset>
```

Le cadre du haut a toujours 100 pixels de haut, le second 75 % de la hauteur totale de la fenêtre disponible, et le cadre du bas dispose du reste. Une dernière possibilité consiste à définir des dimensions relatives à l'aide de coefficients numériques entiers qui créent un partage proportionnel de la hauteur disponible. En écrivant les définitions suivantes :

```
<frameset rows = "2,5,1">
  <frame src="http://www.funhtml.com/pagehaut.html" />
```

```
<frame src=" http://www.funhtml.com/pagecentre.html" />
<frame src=" http://www.funhtml.com/pagebas.html" />
<noframes>
 <body>
  <p>Votre navigateur ne supporte pas les cadres!<br /> Voir la <a id="lien"
    ➥ href="http://www.funhtml.com/index.html" tabindex="1" accesskey="A"
    ➥ title="Version sans cadres">version sans cadres</a>
  </p>
 </body>
</noframes>
</frameset>
```

le cadre supérieur occupera 2/(2 + 5 + 1) soit 2/8 de la hauteur disponible, le suivant 5/8 et le dernier 1/8. Il suffit de changer un seul des coefficients pour que le partage soit différent.

L'exemple 8-1 crée une page divisée en deux cadres horizontaux dont les hauteurs respectives sont définies à 200 et 600 pixels (repère ³). Le contenu du premier cadre est ensuite déterminé grâce à l'attribut src du premier élément <frame /> (repère ·), et nous définissons de la même façon le contenu du second cadre (repère »). L'élément <noframes> fournit un contenu alternatif en proposant une version sans cadres (repère ¿). La figure 8-2 montre le résultat obtenu, le premier cadre contenant la page d'accueil du site www.w3c.org et le second celle du site www.eyrolles.com. Les valeurs utilisées pour les attributs src sont ici celles de sites existants pour simplifier la démonstration, mais dans la pratique des pages avec cadres, il s'agit des URL de pages XHTML standards qui exposent le contenu du site. Ces pages peuvent donc être écrites à l'aide de la DTD XHTML 1.1.

Exemple 8-1. Création de cadres horizontaux

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
 <title>Page avec cadres horizontaux</title>
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <frameset rows="200,600³">
  <frame src="http://www.w3.org" name="haut" frameborder="0" />
  <frame src="http://www.eyrolles.com" name="bas" frameborder="0" />
  <noframes¿
   <body>
```

```
    <p>Votre navigateur ne supporte pas les cadres!<br /> Voir la <a id="lien"
      ➥ href="http://www.funhtml.com/index.html" tabindex="1" accesskey="A"
      ➥ title="Version sans cadres">version sans cadres</a>
    </p>
   </body>
  </noframes>
 </frameset>
</html>
```

Figure 8-2

Une page avec deux cadres horizontaux

Les cadres verticaux

Notre second exemple nous permet de partager une page en cadres verticaux. Pour y parvenir, il suffit de définir l'attribut `cols` de l'élément `<frameset>` à la place de l'attribut `rows` les valeurs indiquées s'appliquant de gauche à droite. La dimension de chaque cadre peut être définie en pixel selon le modèle suivant :

```
<frameset cols = "250, 600, 100">
  <frame src="http://www.funhtml.com/pagegauche.html" />
  <frame src=" http://www.funhtml.com/pagecentre.html" />
  <frame src=" http://www.funhtml.com/pagedroite.html" />
  <noframes>
```

```
  <body>
   <p>Votre navigateur ne supporte pas les cadres!<br /> Voir la <a id="lien"
     ➥ href="http://www.funhtml.com/index.html" tabindex="1" accesskey="A"
     ➥ title="Version sans cadres">version sans cadres</a>
   </p>
  </body>
 </noframes>
</frameset>
```

Dans ce cas, nous divisons la page en trois cadres verticaux, le plus à gauche ayant 250 pixels de large, le deuxième 600 pixels et le troisième 100 pixels. Les éléments `<frame />`définissent comme dans la section précédente le contenu du cadre au moyen de leur attribut `src`.

L'usage des pourcentages pour définir la part de chaque cadre donnerait le code suivant :

```
<frameset cols = "25%, 60%, 15%">
  <frame src="http://www.funhtml.com/pagegauche.html" />
  <frame src=" http://www.funhtml.com/pagecentre.html" />
  <frame src=" http://www.funhtml.com/pagedroite.html" />
  <noframes>
   <body>
    <p>Votre navigateur ne supporte pas les cadres!<br /> Voir la <a id="lien"
      ➥ href="http://www.funhtml.com/index.html" tabindex="1" accesskey="A"
      ➥ title="Version sans cadres">version sans cadres</a>
    </p>
   </body>
  </noframes>
</frameset>
```

Ce code permet d'obtenir des cadres qui gardent les mêmes proportions dans le navigateur quelle que soit la définition de l'écran du visiteur. Comme pour les cadres horizontaux, il est possible de mélanger les dimensions en pixel et en pourcentage dans le même élément `<frameset>`par exemple :

```
<frameset cols = "200,*, 10%">
```

Dans ce cas, le cadre de gauche mesure 200 pixels de large, celui de droite 10 % de la largeur de la fenêtre et le cadre central occupe la largeur restante.

La dernière possibilité de définition des dimensions des cadres avec des proportions permet d'atteindre le même but. La définition suivante crée un partage proportionnel selon les coefficients 2, 5 et 1, soit 2/8 pour le premier cadre vertical, 5/8 pour le deuxième et 1/8 pour le dernier.

```
<frameset cols = "2,5,8">
  <frame src="http://www.funhtml.com/pagegauche.html" />
  <frame src=" http://www.funhtml.com/pagecentre.html" />
```

```
 <frame src=" http://www.funhtml.com/pagedroite.html" />
 <noframes>
  <body>
   <p>Votre navigateur ne supporte pas les cadres!<br /> Voir la <a id="lien"
    ➥ href="http://www.funhtml.com/index.html" tabindex="1" accesskey="A"
    ➥ title="Version sans cadres">version sans cadres</a>
   </p>
  </body>
 </noframes>
</frameset>
```

L'exemple 8-2 crée une page divisée en trois cadres verticaux, le cadre de gauche ayant une largeur fixée à 30 % de la fenêtre du navigateur, celui de droite une largeur fixe de 250 pixels et le cadre central occupe la largeur restante définie à l'aide du caractère * (repère ³). L'adresse de chacun des documents à afficher dans ces cadres est comme précédemment définie par les attributs src des éléments <frame /> (repères · , » et ¿). Pour chacun d'eux, l'attribut frameborder permet de cacher la bordure de séparation entre les cadres. De plus, le cadre de gauche est défini comme non redimensionnable à l'aide de l'attribut noresize (repère ·).

Exemple 8-2. Création de cadres verticaux

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
 <title>Cadres trois colonnes</title>
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <frameset cols="30%,*,250" ³
  <frame src="http://www.arte-tv.com" name="gauche" noresize="noresize"
   ➥ frameborder="0" />·
  <frame src="http://www.france2.fr" name="milieu" frameborder="0" />»
  <frame src="http://www.lemonde.fr" name="droit" frameborder="0" />¿
  <noframes>´
   <body>
    <p>Votre navigateur ne supporte pas les cadres!<br />
     ➥ Voir la <a id="lien" href="http://www.funhtml.com/index.html" tabindex="1"
     ➥ accesskey="A" title="Version sans cadres">version sans cadres</á>
    </p>
   </body>
  </noframes>
 </frameset>
</html>
```

La figure 8-3 montre le résultat obtenu. Comme dans l'exemple précédent, notons que dans un site réel le contenu de chacun des cadres est constitué du contenu du site et non de sites externes.

Figure 8-3
Partage de la page en cadres verticaux

Les cadres imbriqués

La création d'une page dont les cadres sont tous orientés dans le même sens se révèle donc très simple comme nous venons de le voir. Au niveau de la présentation, elle présente l'inconvénient de créer des divisions qui sont toutes soit dans le sens horizontal, soit dans le sens vertical, comme nous venons de le réaliser. De la même manière que pour les tableaux, il est possible d'améliorer ces divisions en créant des cadres plus complexes combinant ces deux types de divisions pour obtenir par exemple un résultat similaire à celui présenté à la figure 8-4.

Pour obtenir ce type de résultat, il nous faut emboîter des éléments `<frameset>` les uns dans les autres. Ces inclusions demandent une attention particulière car l'ordre d'apparition des différents éléments `<frameset>` et `<frame />` dans le code XHTML a une importance sur le résultat final. L'écriture du code nécessite au préalable un minimum d'analyse. Pour concevoir la division présentée à la figure 8-4, il faut imaginer la manière dont nous découperions des limites entre les cadres dans une feuille avec une règle et

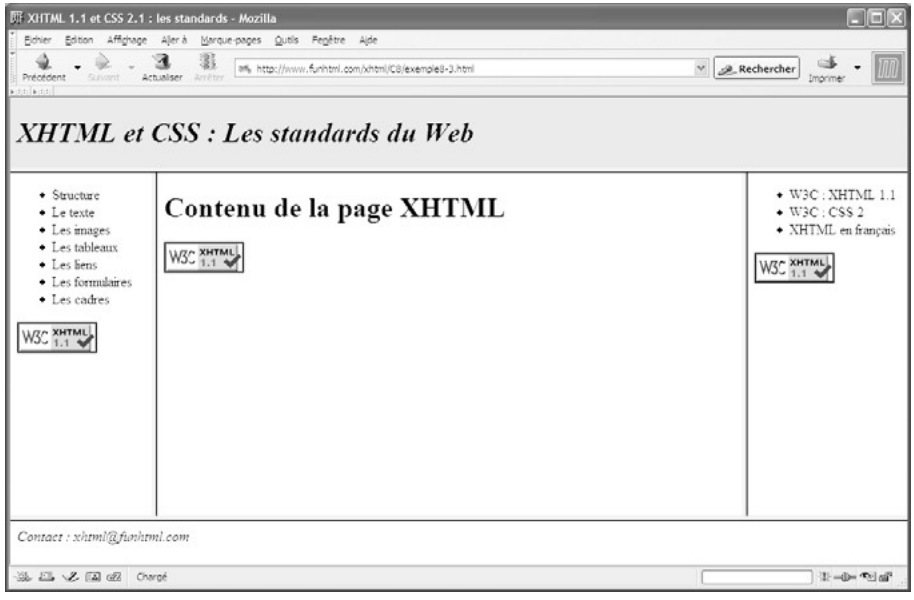


Figure 8-4
Une page avec des cadres imbriqués horizontaux puis verticaux

un cutter, en tenant compte du minimum de coupes, ce qui implique un ordre précis. Le processus est donc le suivant :

- Partager la page en trois cadres horizontaux avec le code suivant (exemple 8-3, repère ³) :

```
<frameset rows="80,*,50">
```

- Définir le contenu du cadre supérieur avec un premier élément `<frame>` et son attribut `src` (exemple 8-3, repère ·).
- Partager le cadre du milieu en trois cadres verticaux à l'aide d'un nouvel élément `<frameset>` selon le code suivant (exemple 8-3, repère ») :

```
<frameset cols="16%,*,18%">
```

- Donner un contenu à ces trois cadres en créant trois éléments `<frame />` et en définissant leur attribut `src` (exemple 8-3, repères ¿ , ´ et ²).
- Refermer le second élément `<frameset>` avec la balise `</frameset>` (exemple 8-3, repère ¶).
- Définir le contenu du cadre du bas avec un dernier élément `<frame>` et son attribut `src` (exemple 8-3, repère º).
- Définir éventuellement un élément `<noframes>` pour créer un contenu de remplacement aux cadres (exemple 8-3, repère ¾).

- Refermer le premier élément <frameset> avec la balise </frameset> (exemple 8-3, repère µ).
- Cette procédure est appliquée dans l'exemple 8-3. Elle permet d'obtenir le résultat présenté à la figure 8-4.

Exemple 8-3. Création de cadres imbriqués

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
 <title>XHTML 1.1 et CSS 2.1 : les standards</title>
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <frameset rows="80,*,50">
  <frame src="entete.html" name="haut" />
  <frameset cols="16%,*,18%">
    <frame src="menu.html" name="gauche" />
    <frame src="contenu.html" name="centre" />
    <frame src="lien.html" name="droit" />
  </frameset>
  <frame src="adresse.html" name="bas" />
  <noframes>
   <body>
    <p>Votre navigateur ne supporte pas les cadres!<br /> Voir la <a id="lien"
      ➥ href="http://www.funhtml.com/index.html" tabindex="1" accesskey="A"
      ➥ title="Version sans cadres">version sans cadres</a>
    </p>
   </body>
  </noframes>
 </frameset>
</html>
```

Comme l'indiquent les images de validation du W3C présentes dans les documents menu.html, contenu.html et lien.html, visibles à la figure 8-4, ceux-ci sont tous validés avec la DTD XHTML 1.1, ce qui confirme l'indépendance entre la page qui crée les cadres en utilisant la DTD XHTML 1.0 Frameset et les documents qui s'affichent dans ces derniers.

Nous pouvons également créer une page avec des cadres imbriqués d'un type différent tels que ceux qui sont représentés à la figure 8-5. En apparence, cette division ressemble beaucoup à celle de la figure 8-4, mais nous allons voir que sa réalisation est beaucoup plus complexe.

La phase préparatoire fait tout d'abord apparaître une division en deux cadres verticaux de 16 % et 84 % de largeur, puis la division du second cadre vertical en deux cadres horizontaux, le premier de 80 pixels de haut et le second occupant la hauteur restante.

Figure 8-5
Cadres imbriqués verticaux puis horizontaux

Ce dernier cadre est ensuite lui-même partagé en deux cadres de 150 pixels de largeur à droite et le reste au centre. Pour obtenir ce résultat, l'ordre d'apparition du code XHTML doit donc être le suivant :

- Partager la page en deux cadres verticaux en utilisant un premier élément `<frameset>` et son attribut `cols` (voir l'exemple 8-4, repère ³).

```
<frameset cols="16%,*">
```

- Utiliser un élément `<frame>` et son attribut `src` pour définir le contenu du cadre de gauche, en l'occurrence le document `menu.html` (voir l'exemple 8-4, repère ·).

```
<frame src="menu.html" name="gauche" />
```

- Partager le cadre vertical droit en deux cadres horizontaux, le plus haut de 80 pixels et le plus bas occupant la hauteur disponible, en écrivant un nouvel élément `<frameset>` et en définissant son attribut `rows` (voir l'exemple 8-4, repère »).

```
<frameset rows="80,*">
```

- Utiliser un élément `<frame>` et son attribut `src` pour définir le contenu du cadre supérieur haut nommé `haut` (voir l'exemple 8-4, repère ¿).

```
<frame src="entete.html" name="haut" />
```

- Partager le second cadre horizontal en deux cadres verticaux à l'aide d'un troisième élément <frameset> (voir l'exemple 8-4, repère ´).

```
<frameset cols="*,150px">
```

- Écrire les deux éléments <frame /> et définir leur contenu respectif (voir l'exemple 8-4, repères ² et ¶). Le cadre nommé centre contient le document contenu.html et celui qui est nommé droit le document lien.html .

```
<frame src="contenu.html" name="centre"  />
<frame src="lien.html" name="droit"  />
```

- Refermer le troisième puis le deuxième élément <frameset> (voir l'exemple 8-4, repères º et ¾).
- Écrire un élément <noframes> pour créer un contenu de substitution (voir l'exemple 8-4, repère µ).
- Refermer le premier élément <frameset> (voir l'exemple 8-4, repère ¸).

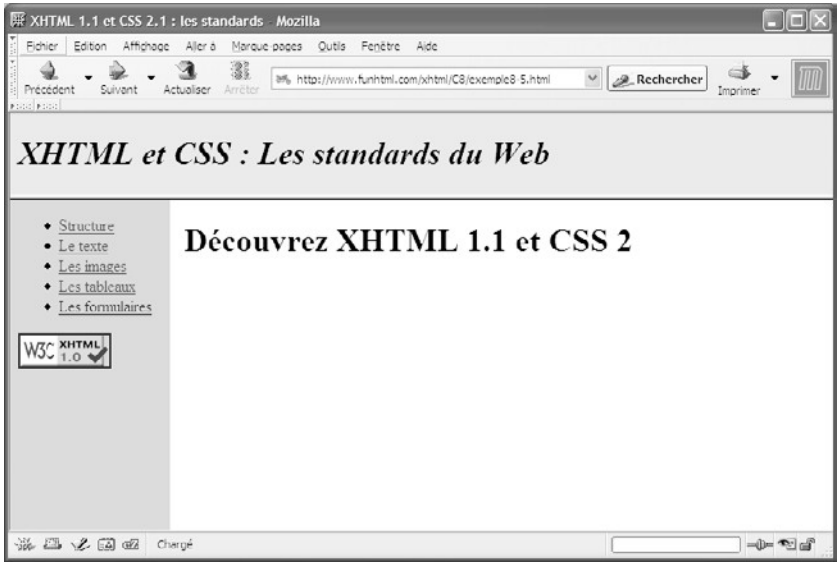
Exemple 8-4. Cadres imbriqués complexes

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
 <title>XHTML 1.1 et CSS 2.1 : les standards</title>
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <frameset cols="16%,84%³">
  <frame src="menu.html" name="gauche" ·/>
  <frameset rows="80,*">»
    <frame src="entete.html" name="haut" /> ¿
  <frameset cols="*,150px">´
    <frame src="contenu.html" name="centre"  />²
    <frame src="lien.html" name="droit"  />  ¶
  </frameset>º
  </frameset>¾
  <noframes>µ
   <body>
    <p>Votre navigateur ne supporte pas les cadres !<br /> Voir la <a id="lien"
      ➥ href="http://www.funhtml.com/index.html" tabindex="1" accesskey="A"
      ➥ title="Version sans cadres">version sans cadres</a>
    </p>
   </body>
  </noframes>
 </frameset>¸
</html>
```

La communication entre les cadres

L'utilisation classique de pages avec cadres est une division en trois cadres comme celle du site présenté à la figure 8-6.

Figure 8-6
Un site composé de trois cadres dont un menu



Le cadre supérieur est habituellement un bandeau publicitaire ou le logo et un texte présentant l'entreprise. Le cadre situé en bas à gauche contient généralement un menu, composé de liens, qui sera toujours visible (rappelons que c'est le principal argument des créateurs de ce type de site). Enfin, le cadre situé en bas à droite est le cadre principal dans lequel s'affichent les pages cibles des liens du menu. Un problème se pose alors pour réaliser cette opération puisque, par définition, chaque cadre a un contenu indépendant des autres et que, par défaut toujours, chaque document cible d'un lien s'affiche dans le même cadre que celui qui contient ce lien. Pour arriver au résultat voulu, nous avons le choix entre deux possibilités : la première oblige à créer tous les documents affichés dans les cadres avec la DTD XHTML 1.0 Transitional, ce qui, il faut le reconnaître, est en parfaite contradiction avec toute la philosophie de cet ouvrage ; la seconde, utilisant JavaScript, permet de créer ces mêmes documents en respectant la DTD XHTML 1.1. C'est pourquoi la seconde méthode a de loin ma préférence, mais nous exposerons cependant ces deux options pour que chacun fasse son choix. La première phase consiste bien sûr à créer la page générant les cadres. Elle est commune aux deux solutions et est réalisée dans l'exemple 8-5 dans lequel nous commençons par partager la fenêtre en deux cadres horizontaux selon la méthode envisagée dans les exemples précédents (repère³). Nous définissons ensuite le contenu du premier cadre nommé haut (repère ·) puis nous partageons le cadre inférieur en deux cadres verticaux à l'aide d'un nouvel élément <frameset> (repère »). Le contenu de ses nouveaux cadres est défini dans deux

éléments <frame /> (repères ¿ et ´). L'élément <noframes> permet de créer un contenu de substitution (repère ²).

Exemple 8-5. La page de création des cadres

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
 <title>XHTML 1.1 et CSS 2.1 : les standards</title>
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 </head>
 <frameset rows="80,*"³
  <frame src="entete.html" name="haut" />
  <frameset cols="16%,*"»
    <frame src="exemple8-6.html" name="gauche" id="gauche" frameborder="0" />
    <frame src="accueil.html" name="centre"  id="centre" frameborder="0" />
  </frameset>
  <noframes>²
   <body>
    <p>Votre navigateur ne supporte pas les cadres!<br /> Voir la <a id="lien"
      ➥ href="http://www.funhtml.com/index.html" tabindex="1" accesskey="A"
      ➥ title="Version sans cadres">version sans cadres</a>
    </p>
   </body>
  </noframes>
 </frameset>
</html>
```

Nous allons maintenant envisager les méthodes qui permettent de fixer l'affichage des documents cibles des liens dans le cadre désiré. La première méthode fait l'objet de l'exemple 8-6, et la seconde de l'exemple 8-7 ; on doit donc pour utiliser cette dernière modifier le code de l'exemple 8-5 en remplaçant tout simplement le code de la ligne suivante :

```
<frame src="exemple8-6.html" name="gauche" id="gauche" frameborder="0" />
```

par celui-ci :

```
<frame src="exemple8-7.html" name="gauche" id="gauche" frameborder="0" />
```

La méthode utilisant la DTD Transitional

Dans la DTD XHTML 1.0 Transitional comme dans les anciennes versions de HTML, l'élément <a> possède un attribut supplémentaire par rapport à celle de XHTML 1.1. Il s'agit de l'attribut target que nous avons déjà évoqué au chapitre 5, *Créer des liens*. La valeur de cet attribut est le nom du cadre dans lequel doit s'afficher le document cible du lien. Cette valeur doit correspondre à celle des attributs name et id du cadre concerné, ce

qui confirme la nécessité de les définir systématiquement avec la même valeur dans les éléments <frame /> Dans ce cas, correspondant au site de la figure 8-6, le cadre de gauche contient le menu vertical créé sous la forme d'une liste de liens qui permet de naviguer dans le site. Ce menu est contenu dans le document de l'exemple 8-6. Chaque clic sur un lien doit afficher un document différent dans le cadre nommé centre (voir l'exemple 8-5, repère ´).

Pour cela, il faut d'abord déclarer l'utilisation de la DTD Transitional dans la déclaration DOCTYPE selon le modèle suivant :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
➥ Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Chaque lien créé avec l'élément <a> et inclus dans la liste à puces introduite par l'élément <ul> (repère ³), peut alors contenir, outre les attributs habituels, l'attribut target dont la valeur est ici fixée à centre puisque tel est le nom du cadre cible (repères · à ²). Du point de vue de la conformité par rapport aux spécifications du W3C, le document de l'exemple 8-5 est conforme à XHTML 1.0 Frameset, et le document de l'exemple 8-6 à XHTML 1.0 Transitional.

Nous obtenons donc un site fonctionnel, même s'il est réalisé au prix d'une entorse à nos principes de base. L'entorse est minime dans la mesure où la DTD Transitional n'est appliquée que pour l'utilisation du seul attribut target , et que nous n'en profitons pas pour ressortir des catacombes des éléments comme ou <font> que nous serions en droit d'utiliser dans ce document. De plus, contrairement à la seconde solution, le site reste fonctionnel même si l'internaute désactive JavaScript.

Exemple 8-6. Méthode utilisant l'attribut target

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Menu</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" >
   body{background-color:#CCC;color:000;}
  </style>
 </head>
 <body>
  <ul> ³
   <li><a  id="structure" href="structure.html" tabindex="1" accesskey="A"
     ➥ title="Structure" target="centre" >Structure</a></li>  ·
   <li><a  id="texte" href="texte.html" tabindex="1" accesskey="B" title="Texte"
     ➥ target="centre" >Le texte</a></li>  »
   <li><a  id="image" href="images.html" tabindex="1" accesskey="C" title="Images"
     ➥ target="centre" >Les images</a></li>¿
```

```
 <li><a  id="tableau" href="tableau.html" tabindex="1" accesskey="D"
    ➥ title="Tableaux"  target="centre">Les tableaux</a></li> ´
 <li><a  id="form" href="form.html" tabindex="1" accesskey="D" title="Formulaires"
    ➥ target="centre">Les formulaires</a></li> ²
 </ul>
</body>
</html>
```

La méthode JavaScript

Comme il a déjà été signalé plusieurs fois, l'utilisation de JavaScript suppose que le visiteur n'a pas eu la mauvaise idée de désactiver les scripts dans son navigateur. Cette désactivation est plutôt rare, ne serait-ce que parce qu'elle demande un effort de configuration manuelle à laquelle peu d'internautes se livrent. Notre seconde méthode peut donc être aussi fonctionnelle dans l'immense majorité des cas. Dans le cas contraire, l'utilisation de l'élément `<noscript>` peut permettre d'afficher un message du type : « Ce site nécessite l'activation de JavaScript », afin de prévenir le visiteur.

La méthode JavaScript permet de détourner subtilement l'interdiction d'employer l'attribut `target` faite par XHTML 1.1 en y faisant pourtant référence, mais sans qu'il soit présent dans le code du document affiché dans le cadre nommé `gauche`. Le menu créé dans l'exemple 8-7 est en effet parfaitement conforme et validé par le W3C selon la DTD XHTML 1.1.

Pour afficher la cible de chaque lien dans le cadre nommé `centre`, il faut gérer l'attribut `onclick` de l'élément `<a>` et écrire un code JavaScript de la forme suivante :

```
 <a  id="structure" href="structure.html" tabindex="1" accesskey="A"
tle="Structure" onclick="document.getElementById('structure').target='centre'"
>Structure</a>
```

ou encore plus simplement, car ce code est contenu dans l'élément lui-même :

```
 onclick="this.target='centre'"
```

le mot-clé `this` permettant de faire référence à l'élément lui-même alors que la méthode `getElementById()` est plus générale et « élégante » car elle permet d'accéder à un élément quelconque du document. L'exemple 8-7 mélange volontairement ces deux codes (repères à ´) pour monter leur équivalence dans le cas présent.

Exemple 8-7. Méthode JavaScript

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html  xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Menu</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
```

```
 <style type="text/css" >
  body{background-color:#CCC;color:000;}
 </style>
</head>
<body>
 <ul>
  <li><a  id="structure" href="structure.html" tabindex="1" accesskey="A"
     ➥ title="Structure" onclick="document.getElementById('structure').
     ➥ target='centre'" >Structure</a></li>
  <li><a  id="texte" href="texte.html" tabindex="1" accesskey="B" title="Texte"
     ➥ onclick="document.getElementById('texte').target='centre'" >Le texte</a>
     ➥ </li>
  <li><a  id="image" href="images.html" tabindex="1" accesskey="C" title="Images"
     ➥ onclick="document.getElementById('image').target='centre'" >Les images</a>
     ➥ </li>
  <li><a  id="tableau" href="tableau.html" tabindex="1" accesskey="D"
     ➥ title="Tableaux"  onclick="this.target='centre'">Les tableaux</a></li>
  <li><a  id="form" href="form.html" tabindex="1" accesskey="D" title="Formulaires"
     ➥ onclick="this.target='centre'">Les formulaires</a></li>
 </ul>
</body>
</html>
```

Comme je l'ai déjà indiqué au début de ce chapitre, cette technique des cadres est considérée comme obsolète, et c'est donc dans le but d'être objectif et avec des réserves que je l'ai présentée dans ce chapitre. En conclusion, je ne peux que recommander à tous ceux qui seraient tentés d'utiliser les cadres dans leurs sites, d'examiner auparavant les solutions qui se présentent à eux. Les chapitres de la deuxième partie de cet ouvrage, et en particulier le chapitre 13 sur la mise en page, le dimensionnement et le positionnement, peuvent les éclairer en la matière. Leur choix final n'en sera que plus motivé.

Exercices

Exercice 1 : La DTD XHTML 1.1 est-elle utilisable pour créer des cadres ?
Les documents affichés dans les cadres peuvent-ils être conformes à XHTML 1.1 ?
Pourquoi ?

Exercice 2 : Dans quel cas, sommes-nous obligés d'utiliser la DTD XHTML 1.0 Transitional ?

Exercice 3 : Quels sont les inconvénients des cadres ?

Exercice 4 : Comment empêcher que les cadres soient redimensionnables par le visiteur ?

Exercice 5 : Comment faire pour supprimer les bordures qui délimitent les cadres par défaut ?

Exercice 6 : Créez une page divisée en trois cadres horizontaux de dimensions respectives 100 pixels, 450 pixels et le reste de la hauteur totale pour le cadre inférieur. Affichez un titre dans le cadre supérieur, une page de votre site préféré dans le cadre central et une page contenant une suite de liens dans le cadre inférieur.

Exercice 7 : Créez une page contenant quatre cadres en opérant une division horizontale et une division verticale sur toute la largeur et la hauteur de la page. Le cadre supérieur gauche contient une image, le cadre inférieur gauche un menu ; le cadre supérieur droit renferme un gros titre, et le cadre inférieur droit le contenu rédactionnel du site.

Exercice 8 : Réutilisez la structure de l'exemple 8-4 (visible à la figure 8-5) pour afficher la cible des liens du cadre de gauche dans le cadre central en n'utilisant pas JavaScript.

Exercice 9 : Utilisez la division opérée dans l'exercice 4 pour afficher les cibles des liens dans le cadre inférieur droit au moyen de JavaScript.

Partie II

Les styles CSS

9

Introduction à CSS

La création de styles CSS (Cascading Style Sheets ou feuilles de style en cascade) est le complément indispensable du langage XHTML. Ce procédé correspond parfaitement à la séparation du contenu et de la présentation sur laquelle nous avons plusieurs fois insisté en décrivant les différents éléments XHTML. D'une part, cette séparation permet d'alléger les pages en centralisant les définitions des styles en un point unique, une seule définition pouvant s'appliquer à un grand nombre d'éléments. D'autre part, elle facilite également la maintenance et l'évolution des sites par voie de conséquence. Elle apporte aussi une plus grande rigueur dans la conception des pages et peut permettre un travail collaboratif entre plusieurs programmeurs travaillant en parallèle, d'où une réduction des délais de fabrication. À l'attention de ceux pour qui ces points peuvent paraîtrent marginaux, nous pouvons ajouter que les styles CSS apportent une bien plus grande richesse créative que ne le permettait le langage HTML utilisé sans CSS. Pour égaler les possibilités graphiques de l'association XHTML et CSS, il aurait fallu alourdir de quantité d'éléments spécifiques le langage XHTML, alors que la tendance actuelle est à l'opération inverse, pour rapprocher XHTML de XML. Dans ce chapitre d'introduction aux styles CSS, nous allons donc aborder les bases indispensables à la compréhension de leur mécanisme. Après les règles générales d'écriture d'un style qui s'avèrent simples, nous envisagerons toutes les nombreuses possibilités d'écriture des sélecteurs qui permettent d'appliquer le style voulu à l'élément voulu, quel que soit son contexte. Nous terminerons cette introduction par l'étude des différentes méthodes d'insertion des styles dans une page, puis par les règles de cascade (le « Cascading » de CSS) et d'héritage, dont la connaissance permet de gérer les situations complexes d'attribution des styles à un élément.

Créer des styles

Les règles générales

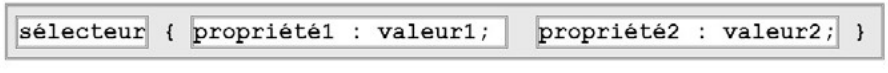
Avant d'aborder les différents méandres de la création de styles, il faut assimiler quelques règles de base et en particulier la syntaxe de la déclaration d'un style (nous parlerons souvent par la suite de « style » au lieu de « déclaration de style »).

Une déclaration de style comporte plusieurs parties, selon l'ordre suivant :

- Un sélecteur qui va déterminer à quel élément et éventuellement dans quelles conditions va s'appliquer le style. Autant que les propriétés, c'est la variété des sélecteurs qui fait la richesse de CSS.
- La déclaration des propriétés que l'on veut voir appliquées à l'élément sélectionné. Elle doit être incluse entre des accolades ouvrante ({) et fermante (}).
- Dans ces accolades doivent apparaître une ou plusieurs propriétés déterminées chacune par un mot-clé propre à CSS suivi du caractère deux-points (:), puis de la valeur attribuée à cette propriété. Si nous définissons plusieurs propriétés dans le même style, il faut séparer chaque déclaration de la précédente par le caractère point-virgule (;). Les propriétés sont en nombre limité et font l'objet d'une recommandation du W3C (voir l'annexe B). La version actuelle de CSS est la version 2.1 (au 13 juin 2005), dans laquelle ont été éliminées un certain nombre de propriétés qui n'étaient mises en application par aucun navigateur. C'est cette version que nous emploierons ici. À chaque propriété correspond un domaine de valeurs particulier constitué par exemple de mots-clés ou de nombres. Vous trouverez dans l'annexe B la liste des propriétés CSS 2.1 et l'éventail de leurs valeurs. Signalons enfin que l'utilisation d'une propriété ou d'une valeur erronée ne provoque pas d'erreur lors de l'exécution comme ce serait le cas dans un langage de programmation. Ces fausses définitions sont simplement ignorées.

La figure 9-1 résume la syntaxe d'écriture d'un style.

Figure 9-1
Syntaxe d'écriture d'un style



Sur ce modèle, nous pouvons par exemple écrire le style suivant :

```
div {color : red ; background-color :yellow ;}
```

dans lequel `div` est le sélecteur, `color` est la première propriété qui détermine la couleur du texte de l'élément, `red` la valeur attribuée à cette couleur, `background-color` qui désigne la couleur de fond est la seconde propriété et `yellow` sa valeur. Tous les éléments `<div>` de la page dans laquelle se trouve cette déclaration ont donc un contenu écrit en rouge sur fond jaune.

Validation du code CSS

Comme pour le code XHTML, il est possible de vérifier la validité des feuilles de style CSS en se connectant sur le site http://jigsaw.w3.org/css-validator/ .

Le validateur signale les erreurs et permet d'apporter les corrections nécessaires. La figure 9-2 montre la page d'accueil du site dans lequel on peut valider le code CSS inclus dans une page XHTML (avec l'extension .htm ou .html) ou dans un fichier externe (avec l'extension .css) situé sur un serveur (repère ³), ou encore inclus dans un fichier local à condition qu'il ne contienne que du code CSS (repère ·)

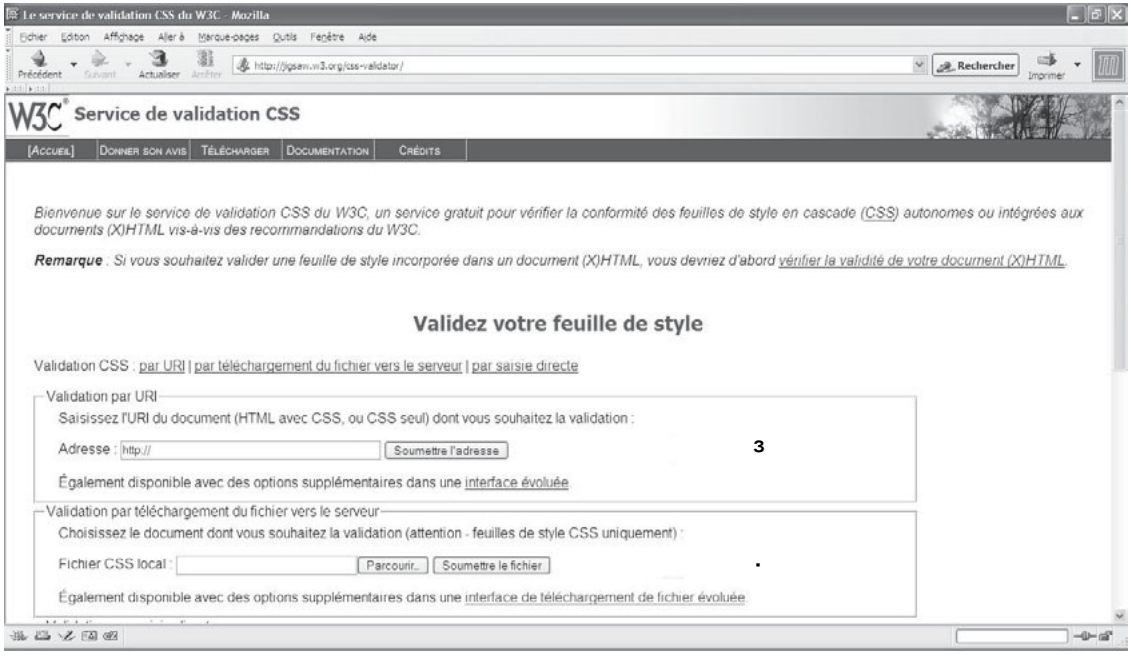


Figure 9-2

La page de validation des feuilles de style

Les sélecteurs

Une des grandes richesses de CSS est la multiplicité des possibilités de sélection des éléments auxquels on veut attribuer un style donné. Cette très grande diversité permet en effet d'appliquer un style aussi facilement à tous les éléments, en une seule ligne de code, qu'à un unique élément isolé dans la page web, sans avoir à écrire la définition localement. De plus, la combinaison de plusieurs sélecteurs dans la même déclaration ouvre la voie à une quasi-infinité de combinaisons, propres à répondre à tous les besoins, même les plus complexes.

Sélectionner un seul élément

Il s'agit de la sélection la plus simple, puisque le sélecteur est constitué du nom de l'élément sans les caractères de début < et de fin de balise />. Nous écrivons par exemple :

```
p {color : yellow ; background-color :blue;}
```

pour que le texte de tous les paragraphes figure en jaune sur fond bleu.

Nous pouvons ainsi définir un style propre à chaque élément comme il en existe un par défaut dans les navigateurs.

Sélectionner plusieurs éléments

Nous pouvons très facilement appliquer le même style à plusieurs éléments différents en les énumérant et en les séparant par une virgule dans le sélecteur. Plutôt que de multiplier les définitions :

```
h1 {color : black ; background-color : red;}
div {color : black ; background-color : red;}
p {color : black ; background-color : red;}
```

nous pouvons écrire le style suivant :

```
h1,div,p {color : black ; background-color : red;}
```

Cette possibilité de regroupement peut être utile pour définir des styles communs à un ensemble d'éléments en écrivant ce type de sélecteur pour cet ensemble, puis en ajoutant d'autres propriétés spécifiques à un des éléments de la liste. On définit ainsi une sorte de tronc commun à un groupe, puis on affine chacun de ses composants.

Si nous écrivons par exemple le code suivant :

```
h1,div,p {color : black ; background-color : red;}
div {margin : 20px;}
```

l'élément va avoir à la fois un texte noir, un fond rouge et une marge de 20 pixels, car la propriété margin définie uniquement pour l'élément s'ajoute à celles déjà définies pour le sélecteur d'éléments h1,div,p .

Le sélecteur universel

Pour appliquer un style à tous les éléments, nous utiliserons le sélecteur universel * avant la définition d'une ou plusieurs propriétés. Par exemple, pour que la couleur du fond de tous les éléments soit le jaune, nous écrirons :

```
*{background-color : yellow;}
```

Cela n'empêche pas de modifier cette couleur de fond pour un élément particulier, en la redéfinissant uniquement pour celui-ci, par exemple :

```
*{background-color : yellow;}
p{background-color : gray;}
```

Dans ce cas, tous les éléments ont un fond jaune, sauf <p> qui a un fond gris redéfini spécialement.

Les classes

Nous avons vu que tous les éléments XHTML possèdent l'attribut class. Ce dernier permet d'appliquer un style défini dans une classe à un élément dont l'attribut class se voit attribuer le nom de cette classe. Pour créer une classe, le sélecteur est constitué du nom choisi pour la classe précédé d'un point (.). Le nom de la classe peut être un mot quelconque, en évitant quand même les noms des propriétés CSS et des éléments XHTML car cela occasionnerait des confusions. Nous pouvons par exemple définir la classe nommée evidence en écrivant le code :

```
.evidence {color : red;}
```

À ce stade, la classe est abstraite et ne s'applique à aucun élément. Pour mettre en évidence un paragraphe précis de la page avec un texte rouge, nous devons alors écrire dans le code XHTML :

```
<p class="evidence">Texte contenu du paragraphe</p>
```

Le texte des autres paragraphes a toujours la couleur qui lui a été attribuée par ailleurs ou la couleur par défaut (noire).

Les classes présentent l'intérêt de pouvoir s'appliquer à n'importe quel élément, n'importe où dans le code de la page. Notre classe evidence peut donc s'appliquer à un titre <h1> une division ou un élément simplement en écrivant pour chacun d'entre eux l'attribut class="evidence".

Nous pouvons également définir une classe en la déclarant applicable seulement à un élément en faisant précéder son nom de celui de l'élément. Nous pouvons écrire par exemple :

```
div.jaune {color : yellow;}
```

Dans ce cas, seules les divisions ayant un attribut class dont la valeur est jaune ont un texte jaune. Les autres éléments, et même s'ils ont le même attribut avec la même valeur, n'ont pas de style défini dans cette classe.

Le sélecteur universel * peut également être employé à la place du nom d'un élément dans la définition d'une classe. Le style s'applique alors à tous les éléments dont l'attribut class a pour valeur le nom de la classe. Nous écrivons alors par exemple :

```
div.jaune {color : yellow;}
```

Il est possible de définir d'abord une classe abstraite, puis de la particulariser en ajoutant une autre propriété pour un élément qui utilisera la même classe. Dans le code CSS ci-après :

```
.rouge {color : red;}
div.rouge {background–color : blue;}
```

suivi du code XHTML dans la page,

```
<div class="rouge">Texte contenu de la division </div>
```

le texte contenu dans l'élément est affiché en rouge sur fond bleu.

Appliquer plusieurs classes au même élément

L'avantage de définir des classes abstraites est, nous l'avons vu, qu'elles peuvent s'appliquer à n'importe quel élément. Leur puissance peut être multipliée car nous pouvons appliquer plusieurs classes indépendantes à un même élément. Celui-ci a alors la combinaison des propriétés de chacune des classes. Pour utiliser plusieurs classes dans le même élément XHTML, il faut donner à son attribut `class` la liste des noms des classes en les séparant par un espace comme ceci :

```
<div class="classe1 classe2"> Ceci est un texte avec la classe 1 et 2 </div>
```

Les combinaisons d'emploi des classes sont alors multiples, chaque classe pouvant définir une caractéristique, et chaque élément pouvant en utiliser plusieurs au choix.

L'exemple 9-1 en donne une illustration. Nous y définissons cinq classes. La première, nommée `jaune`, (repère ³) permet de vérifier le fonctionnement du sélecteur universel dans une classe. Elle s'applique au titre `<h1>` qui utilise la classe `jaune` et est affichée en vert. La deuxième, qui porte le même nom que la précédente, est affectée à l'élément `<div>` et crée un texte en jaune (repère ·). Quand l'attribut `class` d'un élément `<div>` a pour valeur `jaune`, ce n'est pas la première classe qui est utilisée mais la seconde. La troisième crée un texte en rouge (repère »), la suivante un texte en italique (repère ¿) et la dernière crée un fond gris (repère ´). Les différents éléments `<div>` de la page utilisent ces différentes classes. Le premier (repère ¶) utilise celle qui est nommée `classe1` et son texte est donc rouge. Le deuxième (repère º) utilise les deux classes nommées `classe1` et `classe2`, et il combine à la fois un texte rouge et en italique. Le troisième (repère ¾) utilise les classes nommées `classe1 et classe3` et a donc un texte rouge sur fond gris. Enfin, la dernière (repère µ) utilise les classes `jaune`, `classe2 et classe3`. Elle a donc les propriétés de ces trois classes, à savoir un texte jaune, en italique et un fond gris.

Exemple 9-1. Utilisation de plusieurs classes

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <title> Les classes de style </title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <style type="text/css" title="classes">
   *.jaune{color: green;}      ³
   div.jaune {color: yellow;}      ·
   .classe1 {color: red;}      »
   .classe2 {font-style: italic}      ¿
   .classe3 {background-color: #CCC;}´
  </style>
 </head>
 <body>
  <h1 class="jaune">XHTML et CSS</h1>
  <div class="classe1">¶
```

```
  Ceci est un texte avec la classe 1(texte rouge)<br /><br />
 </div>
 <div class="classe1 classe2">°
  Ceci est un texte avec la classe 1 et 2 (texte rouge et en italique)<br /><br />
 </div>
 <div class="classe1 classe3">¾
  Ceci est un texte avec les classes 1 et 3 (texte rouge et fond gris)<br /><br />
 </div>
 <div class="jaune classe2 classe3">µ
  Ceci est un texte avec les classes div.jaune, 2 et 3 (texte jaune, en italique
    ➥ et fond gris)
 </div>
</body>
</html>
```

La figure 9-3 donne un aperçu du résultat obtenu.

Figure 9-3
L'utilisation des classes multiples

Sélecteur d'identifiant id

Pratiquement, chaque élément peut avoir un attribut `id` qui doit être unique dans une page donnée. Nous pouvons écrire un style qui ne sera applicable qu'à l'élément dont l'`id` a une valeur précise en donnant cette valeur au sélecteur (comme pour une classe) et en le faisant précéder du caractère dièse (#).

En écrivant le style suivant :

```
div {color: black;}
#bleu {color: white; background-color: blue;}
```

puis le code XHTML :

```
<div id="bleu">Texte en blanc sur bleu</div>
<div>Texte en noir </div>
```

seul l'élément contenant l'attribut id="bleu" bénéficie du texte en blanc sur fond bleu, et aucun autre, car l'identifiant doit être unique dans une page, et les autres éléments ont un texte noir. Ce type de définition est très ciblé car le style ne peut s'appliquer qu'à un seul élément du fait de l'unicité de l'identifiant id.

La casse des identifiants

Les valeurs de l'attribut id sont sensibles à la casse en XHTML, et bleu et Bleu sont donc deux identifiants différents (sauf dans Internet Explorer pour qui la casse importe peu).

Les sélecteurs d'attributs

Il est également possible d'appliquer un style à un élément déterminé dès qu'il possède un attribut donné, quelle que soit la valeur de cet attribut. Pour appliquer ce sélecteur, le nom de l'élément doit être suivi du nom de l'attribut placé entre crochets ([ ) et ( ]). En définissant le style suivant :

```
acronym[title] {color: red; background-color: gray;}
```

tous les éléments <acronym> qui possèdent un attribut title, quelle que soit sa valeur, ont un contenu affiché en rouge sur fond gris, ce qui permet d'attirer l'attention du visiteur afin qu'il laisse le curseur sur le contenu pour voir apparaître la bulle d'aide donnant la signification de l'acronyme (voir l'exemple 9-2 repères ³, ¸ et ¹).

De même, en définissant le style pour l'élément <img />

```
img[longdesc] {border–color: red; border–weight: 2px;}
```

toutes les images ayant un attribut longdesc ont une bordure rouge de deux pixels de large (voir l'exemple 9-2 repères ·, et). Nous pouvons également créer un style applicable à tous les éléments qui possèdent un attribut donné en utilisant le sélecteur universel * placé devant les crochets qui contiennent le nom de l'attribut choisi. Si nous définissons le style suivant :

```
*[title] {background–color: yellow;}
```

ce sont tous les éléments ayant l'attribut title qui sont affichés avec un fond jaune.

Comme pour les classes, il est possible de sélectionner plusieurs attributs pour un élément en faisant suivre son nom de plusieurs attributs entre crochets. Si nous écrivons le style suivant :

```
h2[title][id]{background-color: yellow;}
```

seuls les titres <h2> ayant à la fois les attributs title et id ont une couleur de fond jaune (voir l'exemple 9-2 repères ¿ et ¾).

Les sélecteurs de valeur d'attribut

Le sélecteur précédent applique un style à un élément par la seule présence d'un attribut précis. Pour affiner ce système, nous pouvons également appliquer un style à un élément à condition que tel attribut ait une valeur précise en utilisant la syntaxe suivante :

```
element [attribut="valeur"] {Définition du style;}
```

En écrivant le style suivant :

```
code[title="code JavaScript"] {color: blue;}
```

tout le contenu des éléments `<code>` ayant l'attribut `title` dont la valeur est la chaîne `code JavaScript` sera affiché avec une police de taille 12 pixels (voir l'exemple 9-2 repères ´ et). La comparaison des valeurs est effectuée au caractère près, par conséquent un élément `<code>` dont l'attribut `title` aurait la valeur `code JavaScript` avec simplement un espace supplémentaire, ne répondrait pas à la condition posée.

Il est ici possible de particulariser davantage l'application du style en sélectionnant plusieurs attributs et leurs valeurs en utilisant la syntaxe :

```
element[attribut1="valeur1"][attribut2="valeur2"] {Définition du style;}
```

Par exemple, avec la définition suivante, appliquée à une cellule de tableau :

```
td {font–size: 12px;}
td [title="nom"] [align="center"] {font–size: 14px; color: red;}
```

seules les cellules ayant un attribut `title` avec la valeur `nom` et un attribut `align` ayant la valeur `center` sont affichés avec un texte rouge et avec une taille de fonte de 14 pixels, alors que le texte des autres cellules a une taille de 12 pixels.

Les exemples précédents impliquent que l'attribut ait exactement la valeur fixée. Il est possible d'étendre encore ce sélecteur en attribuant un style à tous les éléments, dont un attribut donné à une valeur qui ne correspond que partiellement à une chaîne donnée. Pour se voir attribuer le style, les éléments pourront contenir autre chose en plus de la valeur fixée. Pour obtenir cette sélection, il faut utiliser la syntaxe suivante, dans laquelle le signe = est remplacé par ~= :

```
element [attribut ~="valeur"] {Définition des styles;}
```

En écrivant par exemple le style suivant :

```
td[id ~="nom"]{background-color: #222;color: white;}
```

toutes les cellules du tableau dont l'attribut `id` contient la valeur `nom` seront affichées avec un fond gris foncé et en caractères blancs (voir l'exemple 9-2 repères ², ¶, et). Cette possibilité pourrait par exemple être exploitée quand un tableau est construit dynamiquement par un script, l'attribut `id` étant créé comme la concaténation de la chaîne `nom` d'un espace et d'un nombre entier (les chaînes `nom1` ou `prenom` ne conviennent pas).

Si nous appliquons ce style au code XHTML suivant :

```
<tr>
 <td id = "nom 1"> Engels</td>
 <td id = "prenom"> Jean</td>
```

```
</tr>
<tr>
 <td id = "nom 2"> Geelsen</td>
 <td id = "prenom"> Jan</td>
</tr>
<tr> <td id="editeur"> Eyrolles </td> <td>Paris</td></tr>
```

seules les premières cellules des deux premières lignes du tableau ont un contenu affiché en blanc sur fond gris.

Une dernière possibilité consiste à attribuer un style à un élément dont la valeur d'un attribut donné commence par une chaîne fixée. Pour cela, le signe égal doit être précédé du signe | selon la syntaxe suivante :

```
élément [attribut |="valeur"] {Définition des styles;}
```

Dans l'exemple suivant :

```
td[id |="nom"] {font-style: italic;}
```

toutes les cellules de tableau créées par les éléments `<td>` dont l'attribut `id` commence par la chaîne `nom` devraient avoir un contenu affiché en italique. Je précise qu'à ce jour aucun navigateur ne gère ce sélecteur.

L'exemple 9-2 résume toutes les possibilités d'utilisation des sélecteurs de valeur d'attribut que nous venons d'aborder.

Exemple 9-2. Les sélecteurs d'attributs

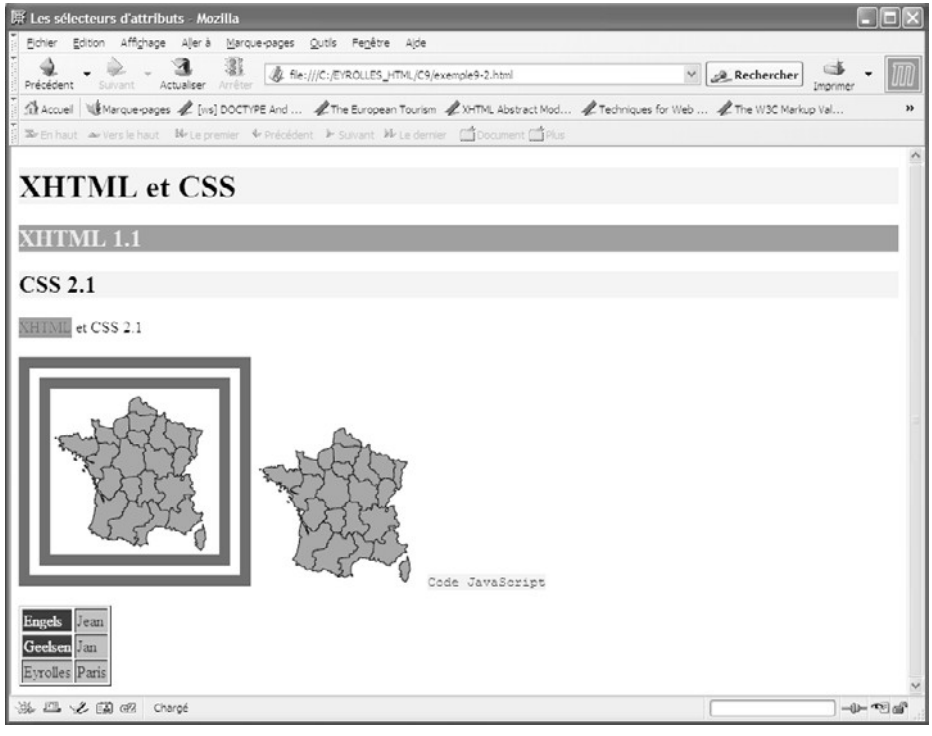
```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les sélecteurs d'attributs</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css">
   acronym[title] {color: red; background-color: gray;}
   img[longdesc]{border: 2em red double}
   *[title]{background-color: #EEE;}
   h2[title][id]{background-color: gray; color: yellow;}
   code[title="code JavaScript"]{color: blue;}
   td{background-color: #AAA; color: blue;}
   td[id ~="nom"]{background-color: #222; color: white;}
  </style>
 </head>
 <body>
    <h1 title="Les outils du Web" >XHTML et CSS</h1>
    <h2 title="Les outils du Web" id="xhtml">XHTML 1.1</h2>
    <h2 title="Les outils du Web" >CSS 2.1</h2>
  <p>  <acronym title="eXtensible HyperText Markup Language">XHTML</acronym> et
       <acronym>CSS 2.1</acronym><br /><br />
```

```
    <img src="../images/france.gif" alt="France"
    ➥ longdesc="funhtml.com/france.html" />
      <img src="../images/france.gif" alt="France" />
      <code title="code JavaScript">Code JavaScript</code>
  </p>
  <table border="1">
   <tr>
        <td id = " nom 1 "> Engels </td>
    <td id = " prénom "> Jean </td>
   </tr>
   <tr>
        <td id = " nom 2 "> Geelsen </td>
    <td id = " prénom "> Jan </td>
   </tr>
   <tr>
        <td id="éditeur"> Eyrolles</td>
    <td> Paris </td>
   </tr>
  </table>
 </body>
</html>
```

La figure 9-4 montre les résultats obtenus pour ces définitions de styles.

Figure 9-4

Les sélecteurs d'attributs



Note
Internet Explorer ignore ces sélecteurs.

Les sélecteurs contextuels parent-descendant

Plutôt que de définir un style pour toutes les occurrences d'un élément, nous pouvons souhaiter ne l'appliquer qu'en fonction de sa position relative par rapport à un autre dans la hiérarchie des éléments de la page. Ce type de sélecteur est dit contextuel. Nous pouvons par exemple définir un style général pour l'élément `<p>` et vouloir lui en appliquer un autre quand il se trouve inclus dans un élément `<div>`. Pour cela, il faut utiliser la syntaxe suivante :

```
element_parent element_enfant {Définition des styles;}
```

En écrivant par exemple le style suivant :

```
p {color: blue;}
div p{color: red;}
```

et en l'appliquant au code XHTML ci-après :

```
<div>Texte de la division
<p>Texte du paragraphe inclus dans div</p></div>
<p>Texte d'un paragraphe non inclus dans div</p>
```

seuls les contenus des éléments `<p>` inclus dans `<div>` sont de couleur rouge, tous les autres étant bleus, et le texte inclus directement dans `<div>` a la couleur par défaut qui est le noir.

Il est aussi possible de préciser la hiérarchie en appliquant plus de deux éléments en les séparant tous par un espace.

Pour appliquer un style différent aux éléments de liste `<li>` inclus directement dans une liste non ordonnée `<ul>` ou dans une liste ordonnée `<ol>` elle-même incluse dans une liste non ordonnée `<ul>`, nous pouvons définir les sélecteurs suivants :

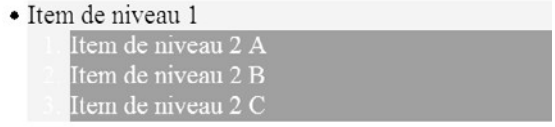
```
ul li {background-color: #EEE; color: black;}
ul li ol li {background-color: gray; color: white;}
```

Une fois qu'ils sont appliqués au code des listes ci-après :

```
<ul>
 <li> Item de niveau 1
  <ol>
   <li> Item de niveau 2 A </li>
   <li> Item de niveau 2 B</li>
   <li> Item de niveau 2 C</li>
  </ol>
 </li>
</ul>
```

nous obtenons pour les items de niveau 1 un texte noir sur fond gris, et pour les trois items de niveau 2, un texte blanc sur fond gris comme le montre la figure 9-5.

Figure 9-5
Les sélecteurs contextuels parent-descendant



Les sélecteurs parent-enfant directs

Nous pourrons préciser le sélecteur précédent en n'appliquant qu'un style à un élément à condition qu'il soit un enfant direct d'un autre élément et non plus un descendant indirect (comme dans la relation petit-enfant/grand-parent).

Pour opérer ce type de sélection, il faut utiliser la syntaxe :

```
element_parent > element_enfant {Définitions des styles;}
```

dans laquelle la présence d'un espace entre chaque élément et le symbole `>` est autorisée mais non signifiante.

En écrivant par exemple le style suivant :

```
span {color: blue;}
div > span{color: red; background-color: #EEE;}
```

et en l'appliquant au code XHTML suivant :

```
<div>Les standards <span>XHTML 1.1</span> et <span>CSS 2.1</span>
➥ s'imposent aujourd'hui
<p>Texte d'un <span>paragraphe</span> inclus dans div</p>
</div>
<p>Texte d'un <span>paragraphe</span> non inclus dans div</p>
```

seul le contenu des paragraphes `<span>` qui sont des éléments enfants directement inclus dans une division `<div>` (repères ³ et ·) est affiché en rouge sur fond gris, les autres éléments `<span>` (repères » et ¿) n'étant pas des enfants directs de `<div>` ont un texte bleu.

> **Note**
> Internet Explorer ignore ces sélecteurs.

Les sélecteurs d'éléments adjacents

Les sélecteurs précédents font appel à la notion de descendance parent enfant. Si on considère dans le code de la page deux éléments consécutifs enfants du même parent, comme deux éléments de bloc `<div>` inclus dans `<body>` par exemple, il n'existe pas de relation de descendance entre eux mais une relation de fratrie. Pour sélectionner ce type

de relation entre éléments, nous disposons du sélecteur + qu'il faut utiliser en adoptant la syntaxe suivante :

```
element1+element2 {Définitions des styles;}
```

Dans ce cas, le style s'appliquera à l'élément de type 2 uniquement s'il suit immédiatement un élément de type 1 dans le code XHTML sans y être inclus. Si nous définissons les styles suivants :

```
p {background–color: yellow; color: blue;}
div+p {background–color: blue; color: yellow;}
```

puis le code XHTML ci-après :

```
<div>Les standards XHTML 1.1 et CSS 2.1 s'imposent aujourd'hui
 <p>Texte d'un paragraphe enfant de div</p>
</div>
<p>Texte d'un paragraphe frère de div</p>
<p>Texte d'un paragraphe frère de div</p>
```

seul le deuxième élément <p> a le style : « texte jaune sur fond bleu », le premier n'étant pas frère mais enfant de l'élément . Le troisième paragraphe aura de plus le même style que le premier car il ne suit pas immédiatement l'élément .

On note que si les deux éléments sont reliés par le sélecteur +, seul le second présente un style défini. En effet, si nous définissons le code suivant au lieu des styles précédents :

```
p {background-color: yellow; color: blue;}
p+p {background-color: blue; color: yellow;}
```

et si nous l'appliquons au même code XHTML que précédemment, seul le troisième paragraphe a le style « texte jaune sur fond bleu ». Les premier et deuxième paragraphes bien qu'étant consécutifs n'ont pas le même parent direct. Notons encore qu'avec ce dernier style, si un élément contenait plus de deux éléments <p>, le style « jaune sur bleu » créé avec le sélecteur p+p s'appliquerait à tous les paragraphes enfants de sauf le premier. De même, s'il existait deux paragraphes consécutifs en dehors de l'élément , le second bénéficierait également du même style.

Note
Internet Explorer ignore tout de ces sélecteurs.

Pseudo-classes et pseudo-éléments

Les sélecteurs précédents permettent d'attribuer un style à un ou plusieurs éléments bien définis dans la hiérarchie d'un document XHTML. Les pseudo-classes et les pseudo-éléments permettent d'attribuer un style à une partie abstraite d'un document non identifiable dans cette hiérarchie, par exemple le premier caractère ou la première ligne d'un paragraphe. D'autres pseudo-classes permettent d'attribuer un style à un document en fonction

des actions prévisibles mais non déterminées de l'utilisateur final, par exemple le fait de placer son curseur sur un lien ou un composant de formulaire.

Les pseudo-classes applicables aux liens

Deux pseudo-classes spécifiques aux éléments possèdent un attribut `href` faisant référence à un document externe (lien vers une autre page) ou interne (ancre vers une partie du même document). Il s'agit des pseudo-classes suivantes :

- `:link`, qui permet d'attribuer un style à un lien qui pointe vers un document non encore vu. C'est l'état normal de tous les liens à l'ouverture de la page.
- `:visited`, pour attribuer un style à un lien qui pointe vers un document déjà vu, après un retour sur la page d'origine.

Avec ces pseudo-classes, on pourra par exemple attribuer une valeur ou une taille de police spécifique au texte des liens visités ou pas. Nous les étudierons en détail au chapitre 12. Pour les employer, il faut faire précéder le nom de la pseudo-classe de celui de l'élément selon le modèle suivant concernant l'élément `<a>` :

```
a:link {color: blue;}
a:visited {color: red;}
```

Les pseudo-classes dynamiques

Elles permettent d'attribuer un style à un élément en fonction des actions effectuées par le visiteur. Ces pseudo-classes sont dynamiques car le style attribué disparaît avec le motif de leur création. Elles sont au nombre de trois :

- `:focus`, pour attribuer un style à l'élément qui a le focus, soit qu'il lui ait été donné par le code XHTML à l'aide des attributs `tabindex` ou `accesskey` soit qu'il l'ait obtenu par un déplacement du pointeur provoqué par l'internaute. Le style disparaît quand l'élément perd le focus. Cette pseudo-classe est mal prise en compte par les navigateurs actuels. Nous pouvons définir par exemple les styles suivants pour affecter les éléments `<a>` et `<input />` quand ils reçoivent le focus :

```
a:focus{color: red;}
input:focus{background–color: blue;}
```

- `:hover`, pour attribuer un style à un élément visible dont la zone est survolée par le pointeur de la souris. Quand le pointeur quitte cette zone, le style est annulé, ce qui peut produire des effets visuels intéressants. Dans le style suivant, les divisions qui sont survolées par le curseur ont un fond rouge et un texte blanc le temps du survol.

```
div:hover{backround-color: red; color: white;}
```

- `:active`, pour attribuer un style à un élément dit actif, c'est-à-dire quand l'utilisateur clique sur son contenu. Là aussi, l'effet est transitoire et ne dure que le temps de l'activation de l'élément.

```
a:active{background-red; color: yellow;}
```

Nous utiliserons ces pseudo-classes de manière plus approfondie au chapitre 12. Les pseudo-classes dynamiques sont mal prises en compte par les navigateurs actuels.

Pseudo-classes diverses

On peut aussi mentionner l'existence de deux autres pseudo-classes d'usage assez général.

- :first-child, qui permet d'appliquer un style à un élément donné uniquement s'il est le premier enfant d'un autre élément. L'élément parent n'est pas nommé et peut être quelconque (à condition de respecter les règles d'inclusion parent-enfant de la DTD). En écrivant par exemple le style :

```
span:firsthild {background-color: red; color: white;}
```

puis le code XHTML ci-après :

```
<div>Les standards tels que <span>XHTML 1.1</span> et <span>CSS 2.1</span>
➥ s'imposent aujourd'hui</div>
```

seul le texte du premier élément contenu dans est de couleur blanche avec un fond rouge. De même, en écrivant le style suivant :

```
a:first-child {backgroor: red; color: white;}
```

et en l'appliquant dans l'exemple de code XHTML :

```
<p> Liens utiles :
<a href= "w3.org" alt="w3c"> W3C </a>  
<a href= "php.net" alt= "PHP"> PHP</a>
</p>
```

seul le texte « W3C » sera en rouge puisque le lien est le premier enfant du paragraphe <p>.

- :lang(code), qui ressemble au sélecteur basé sur la valeur d'un attribut car il permet d'attribuer un style à un élément si son attribut lang a la valeur précisée par son code de langue. En écrivant par exemple :

```
p:lang (fr) {color: black;}
p:lang (en-us) {color: gray;}
```

les paragraphes ayant l'attribut xml:lang="fr" sont affichés en caractères noirs et ceux ayant l'attribut xml:lang="en-us" seront affichés en gris. Cette pseudo-classe est mal prise en compte par les navigateurs actuels.

Les pseudo-éléments

Leur nom vient de ce qu'ils permettent d'agir sur une partie du contenu d'un élément comme s'il était contenu dans un nouvel élément fictif. On dénombre les quatre pseudo-éléments suivants :

- :first-letter, qui permet d'affecter un style à la première lettre du contenu d'un élément indiqué avant ce sélecteur. On l'utilise classiquement pour créer des effets de lettrines en définissant pour ce sélecteur une taille de caractères très supérieure à la taille de l'élément. En écrivant par exemple le style suivant :

```
p:first-letter {fonize: 300%; color: blue;}
```

la première lettre de chaque paragraphe sera trois fois plus grande que les autres et de couleur bleue. Le pseudo-élément :first-letter n'admet que les propriétés suivantes :

font, font-size, font-family, font-style, font-weight, color, background, margin, padding, border, text-decoration, vertical-align, text-transform, line-height, float, letter-spacing, word-spacing, clear.

- first-line, qui permet d'affecter un style à la première ligne du contenu de l'élément indiqué. Cet affichage permet d'attirer l'attention sur un texte. En écrivant le style suivant :

```
div:first{font-size: 150%; font-weight: bold;}
```

La première ligne de chaque division sera affichée en gras et dans une taille 1,5 fois plus grande que la police en cours. Le pseudo-élément : first-line n'admet que les propriétés suivantes :

font, font-size, font-family, font-style, font-weight, color, background, word-spacing letter-spacing, text-decoration, vertical-align, text-transform, line-height.

- before, qui permet d'insérer un contenu doté d'un style particulier avant le contenu réel de l'élément précisé, en l'associant avec la propriété content En écrivant le style suivant :

```
cite:beforeontent:"<<"; font-weight: bold;}
```

Chaque contenu d'une citation <cite> sera précédé des caractères << en gras.

- after, qui joue un rôle similaire au précédent mais définit un contenu doté d'un style à la fin du contenu de l'élément utilisé. En écrivant :

```
cite:after {cnt: ">>";font-weight: bold;}
```

Chaque citation contenu dans l'élément <cite> sera suivie des caractères >> en gras.

La déclaration !important

Chaque déclaration de style peut revêtir un caractère de plus grande importance par rapport à une autre déclaration concernant le même élément et la même propriété qui comporte une valeur différente. Ces deux déclarations peuvent entrer en conflit au moment de la création de la présentation par le navigateur. Pour donner cette importance à un style, il faut insérer la déclaration d'importance à l'aide du mot-clé !important en le plaçant entre la valeur attribuée à la propriété et le point-virgule qui termine la déclaration. Dans l'exemple suivant :

```
*{color: black !important; background-color: yellow;}
div{color: blue; background-color: white;}
```

les couleurs de texte et de fond des sélecteurs * et div sont en conflit, mais comme la propriété color définie dans le sélecteur universel * est marquée !important, le texte de la division figure en noir. En revanche, le fond de la division est de couleur blanche car la valeur yellow n'est pas marquée !important et que la déclaration faite dans div est spécifique.

Nous reviendrons en détail sur les règles de priorités dans la section consacrée à la définition des effets de cascade en fin de chapitre.

Il est évidemment possible d'utiliser la déclaration `!important` pour plusieurs propriétés dans la même déclaration, par exemple :

```
*{color: black !important; background-color: yellow !important;}
div{color: blue; background-color: white;}
```

Dans ce cas, tous les éléments `<div>` ont un contenu affiché en noir sur fond jaune, et tous les styles définis en propre pour cet élément sont ignorés.

Écrire des feuilles de style

Nous allons envisager maintenant les différentes méthodes d'écriture des style CSS et la façon dont on peut les lier à un document XHTML.

Dans l'élément <style>

Défini dans la première partie de ce livre, l'élément `<style>` a pour vocation de renfermer les définitions des styles CSS utilisables dans la page qui le contient. Rappelons qu'il doit toujours être inclus dans l'élément `<head>` et qu'il ne peut contenir que des définitions de styles CSS et des commentaires XHTML délimités par `<!--` et `-->` ou des commentaires CSS délimités par /* et */.

Dans l'éventualité où toutes les pages d'un site ont en commun un certain nombre de styles et que chaque page possède quelques styles propres, les styles communs peuvent être écrits dans un fichier externe (voir la section suivante) et inclus dans l'élément `<style>` au moyen de la directive `@import` selon la syntaxe suivante :

```
@import url(fichier.css);
```

L'URL du fichier peut être relative ou absolue et elle peut être suivie de la désignation du média auquel les styles importés doivent s'appliquer spécifiquement. La page web a alors le comportement correspondant au cas où tous les styles contenus dans le fichier `fichier.css` sont écrits explicitement dans l'élément `<style>`. Cette directive doit figurer avant les autres définitions de style. Un élément `<style>` peut donc avoir la structure suivante, et comporter plusieurs directives `@import` :

```
<style type= "text/css">
 @import url(commun.css)all;
 @import url(ecran.css)screen;
 @import url(imprimante.css)print;
 div,p {font-style: italic;}
 h1,h2 {color: red;}
</style>
```

Si les styles définis après la directive `@import` sont en contradiction avec ceux qui sont contenus dans le fichier importé, les conflits éventuels sont tranchés selon les règles définies dans la section concernant les effets de cascade dans ce chapitre.

Dans un fichier externe

La tendance actuelle étant à la recommandation de la séparation du contenu et de la présentation des pages web, l'écriture des styles dans les fichiers externes est fortement conseillée, même si dans nos exemples nous ne l'utiliserons que très peu par commodité de présentation. Il s'agit de fichiers écrits en texte brut réalisables avec un éditeur simple comme le Bloc-notes de Windows mais aussi avec EditPlus ou même des éditeurs spécialisés qui fournissent une aide à la saisie. Le fichier ne devra contenir que des sélecteurs et les définitions des styles ainsi que des commentaires CSS (délimités par les caractères `/*` et `*/`) mais aucune balise d'élément XHTML. Le fichier CSS doit toujours être enregistré sous l'extension `.css` et être présent sur le serveur, tout comme les fichiers XHTML qui l'utilisent.

L'exemple de code suivant montre un fichier CSS nommé `commun.css`

```
/* Styles communs à toutes les pages */
/* fichier:« commun.css » */
body {background-color: white; color: marine;}
h1 {color: black; font-size: 20px;}
div,p {font-size: 12px;}
a:link {color: blue;}
a:hover {color: red;}
```

Un fichier externe peut inclure les styles d'un autre fichier externe en faisant appel à la directive `@import` de la même façon que nous avons définie plus haut pour les éléments `<style>`.

Pour affecter un fichier de styles à une page de code XHTML, il faut utiliser l'élément `<link />` dans l'en-tête du document avec par exemple le code suivant :

```
<link rel="stylesheet" type="text/css" href="commun.css" media="screen"
➥ title="Styles de base" />
```

Cet élément a été vu en détail au chapitre 2 mais il n'est pas inutile de rappeler que l'attribut `href` contient l'adresse relative ou absolue du fichier CSS, ce qui n'interdit pas de lier un document XHTML à un fichier CSS présent sur un autre serveur. On peut utiliser autant d'éléments `<link />` que souhaité, chacun étant par exemple adapté à un type de terminal particulier précisé par l'attribut `media`

Dans l'attribut style

Nous signalons cette possibilité pour mémoire car la DTD XHTML 1.1 autorise encore la présence de cet attribut. Cependant, il n'est pas conseillé de l'utiliser.

Il servait jusqu'à présent à créer des styles pour tout élément, en particulier pour créer des styles très ponctuels pour l'élément `<span>`

Nous pouvons écrire par exemple :

```
<p> Le langage <span style="color: red "> XHTML </span> représente la dernière
➥ évolution du <span style="color: gray"> HTML </span> </p>
```

Dans ce cas, les mots « XHTML » et « HTML » ont respectivement les couleurs rouge et grise.

Il va de soi que ce type de code ne correspond en rien à la philosophie de l'association XHTML et CSS, qui commande une séparation du contenu et de la mise en forme. De plus, toute modification de ces styles demande une exploration de tout le code XHTML afin de repérer tous les attributs `style`, ce qui rend la maintenance plus longue à réaliser.

Nous abandonnerons cette possibilité encore offerte, mais appelée à disparaître. De nombreux moyens existent pour s'en passer. En effet, la création des classes de styles suivantes et leur utilisation via l'attribut `class` permettent d'obtenir le même effet que le code précédent.

```
/*Dans l'élément style*/
.red {color: red;}
.gris {color: gray;}
< !—Dans le corps du document XHTML-->
<p> Le langage <span class="red"> XHTML </span> représente la dernière évolution
➥ du <span class="gris"> HTML </span> </p>
```

La définition des styles s'y trouve centralisée dans l'élément `<style>` et ce n'est que leur utilisation qui est incluse dans le code XHTML. Il est donc plus aisé de procéder à des modifications selon les besoins.

Cascade et héritage

Savoir définir des styles est une chose, être sûr du résultat final en est une autre. Même si l'on a défini des styles dans l'élément `<style>` ou dans un fichier externe, il faut encore tenir compte, en dehors des problèmes d'interprétation propres aux différents navigateurs, du fait que l'on n'est pas le seul à avoir créé des styles pour des sélecteurs donnés. Les styles CSS ont en effet des origines diverses. Ils peuvent provenir du concepteur (c'est-à-dire de soi-même ou « l'auteur » selon le vocable du W3C). Il peut s'agir aussi de l'utilisateur final qui a la possibilité de définir sa propre feuille de style dans son navigateur. Même si les utilisateurs se prêtent rarement à cet exercice, cela est possible, en particulier pour ceux qui ont des problèmes de vue et qui vont définir par exemple un affichage en très gros caractères, ou ceux qui ont des problèmes de vision de couleurs et qui vont donc définir des choix leur permettant d'obtenir un contraste fond/texte lisible. La dernière origine des styles est le navigateur lui-même qui possède sa propre feuille de style par défaut, dont un modèle est édicté par le W3C, et dont vous trouverez la copie dans l'annexe B.

C'est la partie logicielle CSS du navigateur qui se charge de résoudre les conflits pouvant exister entre les différentes déclarations de styles attachées à un élément. Les différentes

opérations permettant d'attribuer finalement un style propre à un élément constituent ce que l'on nomme les règles de cascade de CSS (d'où le mot « Cascading » de CSS). Pour déterminer les priorités, les navigateurs font d'abord l'inventaire de toutes les déclarations qui s'appliquent à un élément donné, puis procèdent à la sélection en fonction de différents critères que nous allons envisager maintenant.

Sélection selon le média

Les styles pour lesquels le média spécifié ne correspond pas au moyen de visualisation sont éliminés. Le média peut avoir été indiqué dans les éléments `<link>` ou `<style>`, dans les deux cas au moyen de l'attribut `media` (qui peut prendre les valeurs `screen`, `print`, `projection`, `aural`, `braille`, `handheld`, `tty`, `tv` et `all`, dont nous avons déjà donné la définition), ou encore à la suite de la directive `@import` pour ceux qui sont importés directement dans l'élément `<style>`.

Sélection selon le créateur du style

Pour un média donné, des règles de priorité sont définies en fonction de l'origine des styles et du fait qu'ils soient marqués avec la déclaration `!important` ou pas. Les styles marqués `!important` l'emportent toujours sur ceux qui ne le sont pas. En cas d'égalité, les priorités de CSS 2.1 sont les suivantes, de la plus importante à la moins importante :

- Les styles de l'utilisateur employant la déclaration `!important`, ce qui lui permet d'avoir le dernier mot. Cela semble normal, mais ce n'était pas le cas avec CSS 1.
- Les styles du concepteur (l'auteur) marqué avec la déclaration `!important`.
- Les styles du concepteur non déclarés `!important`.
- Les styles du visiteur non déclarés `!important`.
- Les styles par défaut du navigateur passe en dernier et ne peuvent donc s'imposer à personne.

Les exemples qui suivent vont illustrer ces règles.

En écrivant dans le même fichier ou dans le même élément `<style>`, ou encore dans des fichiers différents, les styles suivants, nous obtenons des résultats divers selon les cas.

```
h1{color: blue !important;}
h1{color: red;}
```

L'élément `<h1>` a un contenu affiché en bleu car le style marqué par la déclaration `!important` l'emporte sur tous ceux n'ayant pas cette déclaration.

```
/*style de l'utilisateur */
h1{color: blue;}
/*style de l'auteur */
```

```
h1{color: red;}
/*style du navigateur */
h1{color: black;}
```

L'élément `<h1>` a un contenu affiché en rouge car les styles du concepteur, non déclarés `!important`, l'emportent sur ceux de l'utilisateur, également non déclarés `!important`, et toujours sur ceux du navigateur.

En revanche, avec les déclarations de styles suivantes :

```
/*style de l'utilisateur
h1{color:blue !important;}
/* style du concepteur
h1{color:red !important;}
```

c'est le style de l'utilisateur qui l'emporte quand il entre en conflit avec un style de l'auteur également déclaré `!important`, et l'élément `<h1>` a donc un contenu bleu.

Sélection par spécificité

Pour les styles non encore départagés par les conditions précédentes, on définit une spécificité pour chacun d'eux, à l'aide d'un nombre N de quatre chiffres, sous la forme `abcd` et chaque chiffre est calculé de la manière suivante :

- `a=1` si le style est défini localement dans un attribut `style` et `a=0` sinon. Le nombre de la spécificité vaut alors 1000 et les autres nombres ne sont pas calculés, le style l'emportant sur tous les autres (sauf un style utilisateur marqué `!important`). Quand `a=0` les chiffres suivants `b, c, d` sont évalués.
- Le chiffre `b` représente le nombre de sélecteur d'attributs `id` présents dans l'ensemble du sélecteur. Pour le sélecteur :

```
div#gfonce {Définition des styles}
```

nous avons `b=2`

Le chiffre `c` représente le nombre de classes, de pseudo-classes et de sélecteurs d'attribut présents dans le sélecteur. Pour le sélecteur :

```
div.for a:hover{}
```

e chiffre `c` vaut 2 (une classe `.force` et une pseudo-classe `:hover`).

- Le chiffre `d` représente le nombre d'éléments XHTML utilisés dans le sélecteur. Pour le sélecteur :

```
div p a :be{}
```

e chiffre `d` vaut 3 (trois éléments : `div`, `p` et `a`).

Le exemples de calcul des spécificités suivantes permettront de se familiariser avec ces règles, un peu complexes à assimiler d'emblée.

```
<h1 style="color: red;"> Texte </h1>
```

N=1000 ce style ne peut être dominé que par un style utilisateur déclaré !important (y compris s'il est lui-même déclaré de cette façon).

```
div.h1#force{décl}
```

N=0102 car le sélecteur contient un identifiant (#force) et deux éléments XHTML (div et h1).

```
h1.gras a.rouge {}
```

N=0022 car le sélecteur contient deux classes (.gras et .rouge) et deux éléments XHTML (h1 et a).

```
div#menu {background-color: #FC9;}
```

N=0101 car le sélecteur contient un sélecteur d'identifiant (#menu) et un élément (div).

Sélection selon l'ordre d'apparition

Dans le cas où, malgré tous les critères précédents, deux styles sont encore en conflit, l'ordre d'apparition les départage, le dernier apparu dans le code étant celui qui l'emporte. Ajoutons que les styles importés au moyen de l'élément <link /> sont réputés apparaître avant ceux qui sont écrits dans l'élément <style>. De plus, si plusieurs fichiers de styles sont importés dans le même en-tête <head> les styles du dernier importé l'emportent en cas de conflit. Par exemple, les cas suivants peuvent se produire :

- Cas 1 :

```
h1, h2 {color: yellow;}
h1 {color: navy;}
```

Dans ce cas, le titre <h1> a un texte de couleur navy car ce style apparaît en dernier et écrase le précédent.

- Cas 2 :
  Style écrit dans un fichier externe :

```
/*Style écrit dans le fichier externe « monstyle.css »
h1 {color: navy;}
```

Styles liés et styles internes :

```
Dans l'en-tête <head>
<link rel="stylesheet" type="text/css" href="monstyle.css" />
<style type="text/css">
h1 {color: red;}
</style>
```

ou encore :

```
<style type="text/css">
@import url(monstyle.css)
h1 {color: red;}
</style>
```

Le titre `<h1>` a un texte de couleur `red` car les styles liés apparaissent avant ceux qui sont écrits dans l'élément `<style>`, même si l'élément `<link />` est écrit après l'élément `<style>` dans l'en-tête.

L'héritage

L'héritage est le fait qu'un élément enfant possède les mêmes styles que l'élément qui le contient (son parent dans la hiérarchie des éléments d'une page). Nous pouvons par exemple définir les styles suivants :

```
div{color: white; background-color: blue;}
```

Si dans le code XHTML de la page figurent les éléments suivants :

```
<div>Texte<p>Premier paragraphe <span class="">XHTML</span> et les styles
➥ <strong>CSS 2.1 </strong>sont indispensables </p> à tous !</div>
```

L'élément `<p>` est enfant de `<div>` et les éléments `<span>` et `<strong>` sont eux-mêmes enfants de `<p>`. Par héritage et bien sûr faute d'avoir définis des styles propres pour les éléments `<p>` `<span>` et `<strong>`, ceux-ci ont un contenu qui possède les styles définis pour leur parent direct ou indirect `<div>`. Ils sont donc tous en blanc sur fond bleu. Si nous créons un style différent pour l'élément `<p>`, ses éléments enfants héritent alors de ces styles et non plus de ceux de l'élément `<div>`.

De même, si nous définissons un style pour les éléments de listes ordonnées ou non `<ol>` et `<ul>`, tous les items de la liste, quel que soit leur niveau d'imbrication, ont les mêmes caractéristiques par défaut sans qu'il faille créer un style propre pour eux.

L'héritage concerne un grand nombre de propriétés CSS que nous allons aborder dans les chapitres suivants, mais toutes les propriétés ne sont pas systématiquement héritées, par exemple les marges, les bordures, ou les dimensions et la position pour des raisons évidentes de mise en page. Depuis CSS 2, la quasi-totalité des propriétés peut prendre la valeur `inherit` permettant de définir explicitement l'héritage de la valeur que possède la même propriété dans l'élément parent. Toute modification opérée pour l'élément parent est donc répercutée à ses enfants. Comme trop de précision ne nuit pas, et qu'il est préférable de ne pas laisser l'initiative d'interprétation aux navigateurs, nous conseillons en cas de doute de définir clairement la propriété voulue pour un élément plutôt que de compter sur la réalisation ou non de l'héritage.

Les unités

Toutes les propriétés CSS peuvent prendre une valeur dans un domaine particulier propre à chacune d'elle. En dehors des nombreux mots-clés existants, nous allons faire ici l'inventaire des différents types de valeurs parmi les plus générales que l'on retrouve pour un grand nombre de propriétés.

Les unités de longueur

Elles s'appliquent aussi bien à la taille d'une police qu'à la largeur d'une bordure ou la hauteur d'un élément.

Elles s'expriment par un nombre entier ou décimal selon les cas, suivi d'une unité. On dénombre les unités suivantes :

- Les unités relatives
  - `em` : qui se réfère à la taille de la police utilisée ou à la valeur de la propriété `font-size` (voir le chapitre 12)
  - `ex` : qui correspond à la taille de la lettre « x » minuscule dans la police utilisée.
  - `px` : qui correspond à la taille de 1 pixel. Contrairement à une idée répandue, la taille de 1 pixel n'est pas une taille absolue car elle dépend du média de visualisation et de la distance entre l'œil et le média.
- Les unités absolues : elles sont recommandées quand les caractéristiques physiques (mesurables) du média sont connues.
  - `in` : soit un pouce anglais (un inch), donc 25,4 mm ;
  - `cm` : le centimètre ;
  - `mm` : le millimètre ;
  - `pt` : le point qui représente conventionnellement 1/72 de pouce ;
  - `pc` : le pica qui représente 12 points, soit 1/6 de pouce.
- Les pourcentages qui, comme chacun le sait, ne sont pas des unités mais une convention décriture, le symbole % représentant la fraction 1/100. Leur utilisation fait toujours référence à une autre dimension, celle de l'élément parent le plus souvent, ce qui permet de calculer la dimension voulue.

Les couleurs

Une valeur de couleur s'exprime en mettant en œuvre l'une des trois manières suivantes :

- Un mot-clé parmi une liste limitative donnée à l'annexe C. Tous les mots-clés sont en anglais, par exemple `black`, `yellow` qui correspondent à des couleurs connues ; d'autres sont plus fantaisistes comme `whitesmoke`
- Un code hexadécimal de couleur basé sur les composantes RGB d'une couleur dans le système additif. Chaque composante prend une valeur qui va de 0 à FF, et l'ensemble doit être précédé du caractère dièse (#), par exemple `#F4C5A8`. Il est possible de ne préciser que trois nombres hexadécimaux de 0 à F, par exemple `#FC5` les navigateurs convertissant ces valeurs par réplications (la couleur notée `#FC5` est interprétée comme `#FFCC55`)

- À l'aide de la fonction `rgb()` qui admet trois paramètres représentant la valeur des composantes RGB d'une couleur selon la syntaxe `rgb(Red, Green, Blue)` chaque composante est exprimée par un nombre entier variant de 0 à 255 ou par un pourcentage de 0 à 100 %. On peut définir par exemple des couleurs de la façon suivante :

```
rgb(45, 78, 2ou encore rgb(25%, 85%, 12%).
```

Exercices

Exercice 1 : Écrivez la syntaxe générale de la déclaration d'un style.

Exercice 2 : Écrivez le sélecteur et le style donnant une couleur rouge et un fond noir aux éléments `<h1>` et `<h3>`

Exercice 3 : Écrivez le sélecteur afin que tous les éléments de la page soient écrits en vert.

Exercice 4 : Écrivez une classe qui définit un fond jaune et appliquez-la aux éléments `<h1>` et `<p>`

Exercice 5 : Écrivez une classe spécifique à un élément `<code>` afin que son texte soit bleu.

Exercice 6 : Écrivez les classes correspondant aux styles « fondgris », « textevert » et « textejaune ». Appliquez la première et la troisième à un paragraphe, puis la première et la deuxième à une division `<div>`

Exercice 7 : Écrivez le sélecteur afin que l'élément dont l'attribut `id` vaut « menu » ait un fond rouge.

Exercice 8 : Écrivez le sélecteur afin que tous les éléments ayant un attribut id aient un texte noir sur fond jaune.

Exercice 9 : Écrivez le sélecteur afin que les éléments `<h1>` ayant un attribut `title` aient un texte bleu, les autres ayant un texte noir.

Exercice 10 : Écrivez les sélecteurs afin que les paragraphes inclus dans `<body>` aient un texte gris et que ceux inclus dans `<div>` aient un texte marron.

Exercice 11 : Écrivez le sélecteur afin que seuls les éléments enfants de <p> aient un texte bleu, tous les autres ayant un texte noir.

Exercice 12 : Pour une liste imbriquée sur deux niveaux, écrivez le sélecteur pour que les éléments `<li>` de premier niveau inclus dans `<ol>` soient rouges, et que ceux de second niveau soient en vert.

Exercice 13 : Écrivez le sélecteur pour appliquer un style différent à un élément `<li>` selon qu'il est inclus dans `<ol>` ou `<ul>`.

Exercice 14 : Écrivez le sélecteur pour que le survol d'un élément `<h1>` provoque le changement de couleur du texte en rouge.

Exercice 15 : Écrivez le sélecteur afin que seule la première ligne d'un paragraphe soit en rouge, le reste s'affichant en gris.

Exercice 16 : Où peut-on écrire des styles CSS ?

Exercice 17 : Écrivez des styles dans un fichier externe et les incorporer dans l'élément `<style>`.

Exercice 18 : Comment un utilisateur peut-il s'assurer que ses styles personnels sont bien appliqués ?

Exercice 19 : En reprenant les exercices précédents, définissez les couleurs en utilisant la fonction `rgb()`.

Exercice 20 : Testez les différents styles des exercices précédents dans le validateur du W3C, en les écrivant dans l'élément `<style>` ou dans un fichier externe.

10

Couleurs et image de fond

Dans les anciennes versions de HTML, les définitions des couleurs de polices faisaient appel à une multiplication d'éléments `<font>`, et à leur attribut `color`, appliqués localement. Il en était de même pour les couleurs de fond des éléments qui ne pouvaient être définis que pour un nombre restreint d'éléments. Ces éléments et attributs ont heureusement disparu des recommandations XHTML, au profit de définitions centralisées dans une feuille de style CSS. Chaque élément peut également se voir attribuer une couleur ou une image de fond.

Nous allons envisager ici la manière de définir la couleur d'avant-plan du contenu d'un élément, c'est-à-dire essentiellement celle du texte qui y est inclus. Nous verrons ensuite comment attribuer les couleurs et images de fond, ainsi que le positionnement de ces derniers. L'ensemble de ces définitions permet la création d'effets attractifs.

La couleur d'avant-plan

Nous ne sommes plus au temps des écrans monochromes ni même de ceux qui affichent seize couleurs. Aujourd'hui, même les téléphones portables ont des capacités supérieures. L'emploi de la couleur dans un site est donc aujourd'hui indispensable. Comme toute nouvelle technologie, l'apparition des écrans couleur a entraîné des abus d'usage. Selon la nature du site que vous allez construire, il faudra agir avec circonspection dans le choix et le nombre de couleurs employées. La multiplication des couleurs ne crée plus nécessairement un effet positif sur le visiteur et une certaine harmonie doit aussi être recherchée. C'est là l'affaire du designer, le créatif qui saura choisir et associer les teintes.

Mais on vous demandera peut-être de remplir à la fois cette fonction en plus de celle de programmeur XHTML. Pour vous aider dans le choix et l'association des couleurs en fonction de l'effet désiré, vous pouvez consulter les sites mentionnés dans l'annexe E. Ils vous aideront par exemple à choisir une couleur de fond selon celle du texte pour obtenir différents effets de contraste.

Comme nous l'avons indiqué, la couleur d'avant-plan est avant tout celle du texte, à laquelle on pense immédiatement quand on parle de couleur, mais quand elle est définie, elle est aussi par défaut celle des bordures de l'élément auquel elle s'applique. Il ne faudra pas oublier cet état de fait lors de la définition des bordures afin de leur affecter explicitement une autre couleur (voir la propriété `border-color`). La couleur d'avant-plan est définie par la propriété `color` selon la syntaxe suivante :

```
sélecteur {color:<couleur> | inherit}
```

Si elle n'est pas définie explicitement, sa valeur peut dépendre du navigateur utilisé, mais il s'agit généralement du noir. Nous pouvons définir explicitement une couleur en utilisant les méthodes vues au chapitre 9 à savoir un nom de couleur (par exemple : `black`), une valeur hexadécimale à trois ou six positions représentant les composantes RGB de la couleur précédée du caractère (#) (par exemple : `#F30` ou `#FA3258`) et enfin à l'aide de la fonction `rgb()` qui doit avoir trois paramètres entiers variants de 0 à 255, représentant également les composantes RGB en notation décimale (par exemple : `rgb(56,250,20)`).

L'exemple 10-1 permet de mettre en œuvre les couleurs d'avant-plan et la manière dont elles sont héritées. Nous y définissons la propriété `color` pour l'élément `<body>` (repère ³), `<div>` (repère ·), `<i>` et `<acronym>` (repère »), et enfin `<span>` (repère ¿). Pour l'élément `<div>`, nous définissons également une bordure pour montrer que la couleur d'avant-plan s'y applique. Nous reviendrons en détail sur la création des bordures au chapitre 11.

Dans le corps du document figure un titre `<h1>` (repère ´) qui n'a pas de style propre. Il hérite donc de la couleur définie pour `<body>`. Vient ensuite une division `<div>` dont la couleur d'avant-plan est noire (repère ²). Le titre `<h2>` qu'elle contient figure donc également en noir, de même que le texte qui suit. L'élément `<p>` (repère º) sans style propre et également inclus dans `<div>` a aussi un texte noir. En revanche, les éléments `<acronym>` (repère ¾) et `<i>` (repère µ), tous deux enfants de ce paragraphe, ont un style propre, et leur contenu est affiché en rouge. De même, l'élément `<span>` enfant de ce même paragraphe, mais ayant également un style propre, affiche son contenu en bleu. Il inclut à son tour deux éléments `<big>` et `<i>`. L'élément `<big>` (repère ¹) hérite de la couleur de `<span>` qui est son parent direct et son texte est donc en bleu. En revanche, l'élément `<i>` (repère) est aussi enfant de `<span>` mais il a un style propre (repère ») et son texte est donc en rouge. Le texte placé dans la fin de la division (repère) est affiché en noir en tant que contenu direct de `<div>`. Remarquons que sa bordure a la même couleur que le texte. Vient ensuite un paragraphe indépendant (repère) sans style propre qui hérite donc de la couleur d'avant-plan de son parent direct qui est

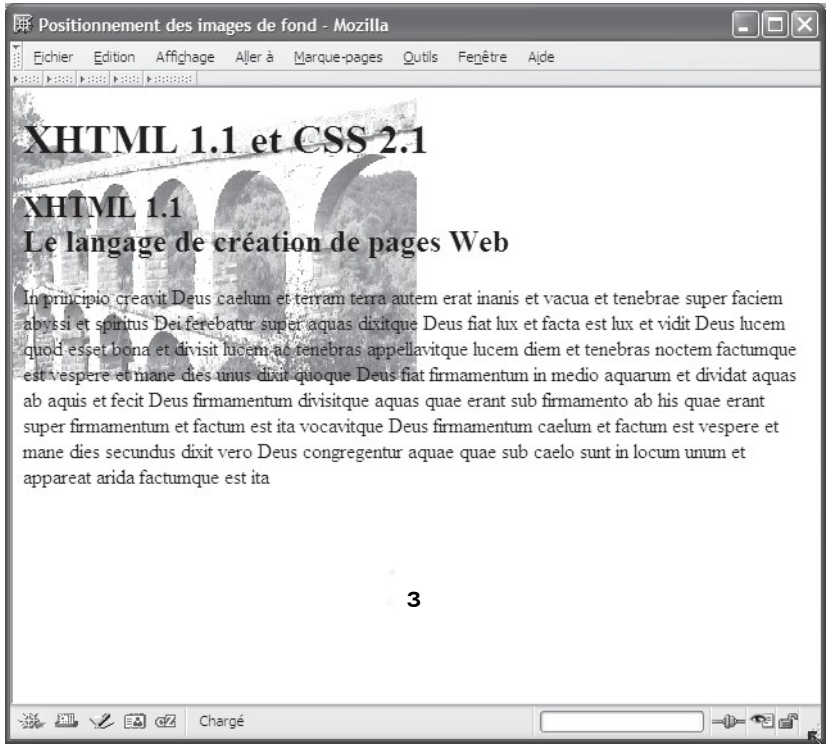
<body>. Comme il inclut des éléments <acronym> et <i> pour lequel la propriété color a la valeur red, le contenu de ces derniers est affiché en rouge comme les éléments inclus dans .

Exemple 10-1. Les couleurs d'avant-plan

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Couleur d'avant plan</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" >
   body{color:#AA5;}
   div{color:#000;  border-width:3px;border-style:solid;}
   i,acronym{color:red;}
   span{color: blue;}
  </style>
 </head>
 <body>
  <h1>In principio creavit Deus caelum et terram terra autem erat inanis</h1>
  <div>
   <h2>Ici commence un &lt;div&gt; </h2>In principio creavit Deus caelum
      ➥ et terram terra autem erat inanis et vacua et tenebrae super faciem abyssi
      ➥ et spiritus Dei ferebatur super aquas.
   <p>  Un paragraphe dans div. Pour <acronym>XHTML </acronym>le respect
      ➥ des recommandations du <i>W3C </i>s'impose à tous les webmestres comme
      ➥ une nécessité...In principio creavit Deus caelum et terram. <br />
    <span>  In principio creavit <big>Deus</big>  caelum et <i>terram</i>
      ➥ terra autem erat inanis </span>et vacua et tenebrae super faciem abyssi et
      ➥ spiritus Dei ferebatur super aquas
   </p>
    Texte de div : In principio creavit Deus caelum et terram terra autem erat
      ➥ inanis et vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur
      ➥ super aquas dixitque
  </div>
  <p>  Un paragraphe indépendant. Pour <acronym>XHTML</acronym>le respect
     ➥ des recommandations du <i>W3C </i>s'impose à tous les webmestres comme une
     ➥ nécessité... </p>
 </body>
</html>
```

La figure 10-1 montre le résultat obtenu pour les différentes couleurs d'avant-plan (ici en niveaux de gris).

Figure 10-1
Les couleurs d'avant-plan et leur héritage



La couleur de fond

La couleur de fond d'un élément est définie par la propriété `background-color` que nous pouvons appliquer à tous les éléments XHTML. Sa syntaxe est la suivante :

```
background-color:<couleur> | transparent | inherit
```

Par défaut, l'arrière-plan est transparent et laisse donc apparaître la couleur de fond de l'élément parent. La transparence peut aussi être définie explicitement avec la valeur `transparent`. Les valeurs du paramètre `<couleur>` sont celles qui ont été définies au chapitre précédent et pour la propriété `color` (mot-clé, code RGB ou fonction `rgb()`). La couleur de fond n'est pas héritée par défaut et il faut utiliser la valeur `inherit` pour obtenir l'héritage de cette couleur. L'exemple 10-2 montre l'application de couleurs de fond à différents éléments. La couleur `yellow` est appliquée au corps du document `<body>` (repère ³). Nous pouvons remarquer dans la figure 10-2 que la couleur de fond de l'élément `<h1>` (repère ¶), n'étant pas définie explicitement, elle est celle de son élément parent `<body>` (repère ³), non pas par héritage, mais parce que son fond est transparent.

Pour la même raison, la définition de la couleur de fond pour l'élément (repère ·) s'applique à l'élément <h2> qui y est inclus (repère ¾). En revanche, le paragraphe <p> (repère µ) a une couleur de fond bien définie (repère »). Il a donc sa couleur propre et non pas celle de l'élément parent

Cette couleur est la même pour l'élément <acronym> (repère ¸) pour la même raison que pour les éléments <h1> et <h2> à la différence près que la couleur de fond est ici définie avec la valeur transparent (repère ´). Enfin, l'élément <i> (repère ¹) ayant un style de fond défini avec le mot-clé orange (repère ¿) son contenu est affiché sur un fond orange. L'association des propriétés color et background-color dans la même classe (repère ²) permet d'obtenir des effets particuliers susceptibles d'attirer l'attention sur un titre (repère).

Exemple 10-2. Application des couleurs de fond

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <title> Les couleurs de fond </title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <style type="text/css" title="">
   body{background-color: yellow;} ³
   div{background-color:#0088FF;}
   p{background-color:rgb(255,0,0);} »
   i{background-color:orange;} ¿
   h2{ background-color:transparent} ´
   .invert{color:white;background-color:#000;}  ²
  </style>
 </head>
 <body>
  <h1>XHTML 1.1 &amp;amp; CSS 2.1</h1>
  <div> º  <h2>XHTML 1.1</h2>
   <p> µ  Pour <acronym>XHTML</acronym> le respect des recommandations du <i>W3C
    ➥ </i> ¹  s'impose à tous les webmestres comme une nécessité...</p><br />
  </div><hr />
  <h2 class="invert">CSS 2.1 : feuilles de style en cascade</h2>
 </body>
</html>
```

La figure 10-2 présente le résultat obtenu.

Figure 10-2
Définition des couleurs de fond

Les images de fond

Bien plus qu'une couleur de fond uniforme, la définition d'une image de fond pour la page entière, ou pour certains de ses composants seulement, est de nature à attirer l'attention du visiteur. Il est cependant conseillé, comme pour toutes les possibilités décoratives, de ne pas en abuser en multipliant différentes images de fond dans la même page. Cette restriction devra être appréciée en fonction de la nature du site et du public visé, portail d'entreprise généralement plus sobre, ou site à caractère ludique plus exubérant.

Définir une image de fond

Pour définir une image de fond, nous utilisons la propriété `background-image` dont la syntaxe est la suivante :

```
background-image:url(<URL>)|none|inherit
```

La valeur `<URL>` passée à la fonction `url()` doit contenir l'adresse relative ou absolue de l'image de fond. Auparavant, l'image choisie doit éventuellement être redimensionnée à l'aide d'un logiciel graphique car il n'est pas possible d'intervenir sur les dimensions de l'image à l'aide des propriétés CSS. Dans tous les cas, l'image n'occupera que l'espace disponible de l'élément quelle que soit sa taille, sans risque de débordement. La valeur `none` définit l'absence d'image de fond et `inherit` permet de forcer l'héritage de l'image de fond de l'élément parent. En effet, les images de fond ne se transmettent pas aux éléments enfants. En pratique, on peut avoir l'impression contraire car un élément enfant, pour lequel aucune des propriétés `background-image` ou `background-color` n'est définie, possède l'image de fond de son parent, car son propre fond est alors transparent.

En revanche, la définition d'une couleur de fond pour un élément, l'emporte sur l'image de fond de son parent. De plus, si nous définissons ces deux propriétés pour le même élément, c'est l'image de fond qui l'emporte sur la couleur. Il est d'ailleurs prudent de définir les deux avec une couleur de fond proche de celle de l'image, car s'il l'on ne trouve pas l'image sur le serveur, c'est la couleur qui sert de fond. Enfin, signalons que, par défaut, l'image de fond est répétée horizontalement et verticalement pour occuper toute la surface de l'élément concerné.

L'exemple 10-3 illustre notre propos en définissant des images de fond pour les éléments <body> (repère ³), <h1> (repère »), (repère ¿), <i> (repère ´) et une classe nommée herit (repère ²) qui est appliquée au second élément <h2> (repère µ). Dans le corps du document, l'image de fond de chacun de ces éléments se superpose à celle de son parent. Pour l'élément <p> (repère °) qui n'a pas d'image de fond, la couleur de fond l'emporte également sur l'image de fond de son parent (repère ¶). L'élément <i> (repère ¾), enfant du paragraphe, possède à la fois une couleur de fond et une image de fond, donc celle-ci l'emporte à la fois sur sa couleur de fond et sur la couleur de fond de son parent.

Exemple 10-3. Les images de fond

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <title> Les images de fond </title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <style type="text/css" >
   body{background-image: url(fondbleu.gif);background-color:blue;} ³
   p{background-color:yellow;} ·
   h1{background-image:url(fond2.gif);} »
   div{background-image:url(fond1.gif);} ¿
   i{background-color:orange;background-image:url(fondjaune.gif);} ´
   .herit{color:red;background-image:inherit;} ²
  </style>
 </head>
 <body>
  <h1>XHTML 1.1 &amp;amp; CSS 2.1</h1>
  <div> ¶ <h2>XHTML 1.1</h2>
   <p> ° Pour <acronym>XHTML</acronym>, le respect des recommandations du <i>W3C
    ➥ </i> ¾ s'impose à tous les webmestres comme une nécessité...</p><br />
  </div><hr />
  <h2 class="herit">CSS 2.1 :<br /> feuilles de style en cascade</h2> µ
 </body>
</html>
```

Sur la figure 10-3 qui montre la visualisation du document dans Mozilla, nous pouvons constater les effets obtenus par les définitions de ces différentes images et couleurs de fond. En étant attentif, on peut remarquer que l'image de fond du second élément `<h2>` contenant le texte « CSS 2.1 », tout en étant la même que celle de la page, est décalée par rapport à celle de `<body>` Ce phénomène ne se produit pas dans Internet Explorer qui effectue un rendu du fond hérité comme si celui-ci avait été déclaré transparent.

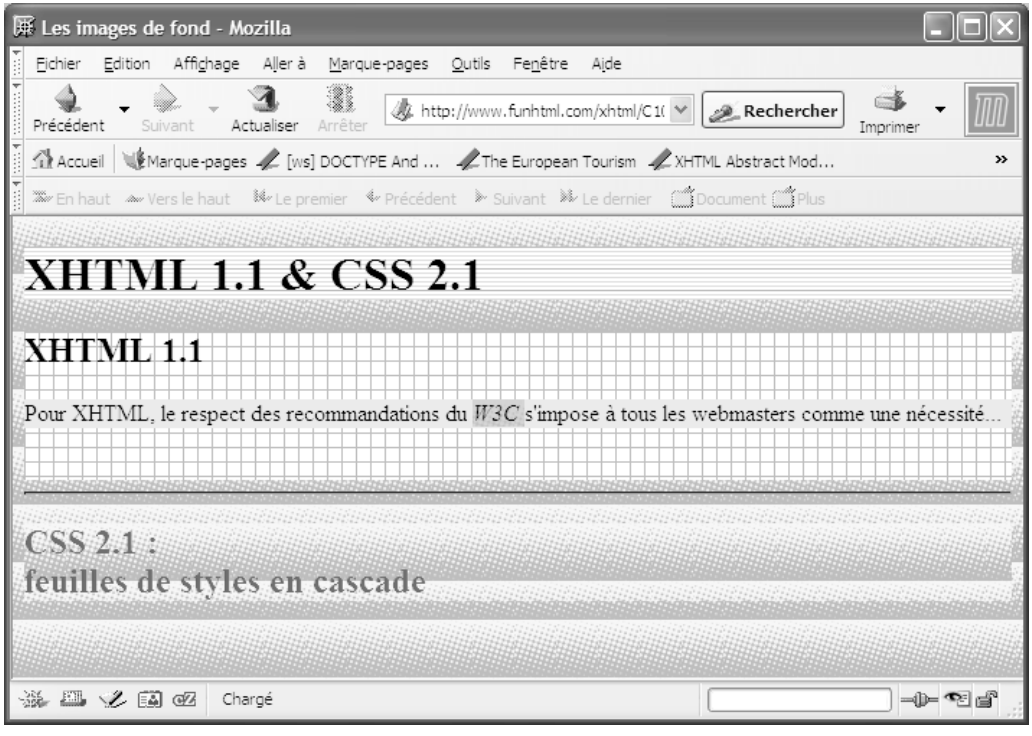


Figure 10-3
Les images de fond

Positionner une image de fond

Par défaut, l'image de fond est répétée horizontalement et verticalement, et occupe tout l'espace disponible de l'élément. Pour modifier cette caractéristique, nous pouvons limiter le type de répétition à l'aide de la propriété `background-repeat` dont la syntaxe est la suivante :

```
background-repeat:repeat|repeat-x|repeat-y|no-repeat|inherit
```

Quand on lui attribue la valeur `repeat`, qui est la valeur par défaut, l'image est répétée selon les axes x et y et occupe toute la boîte de l'élément telle qu'elle a été définie au chapitre 9. Avec la valeur `repeat-x`, l'image n'est répétée que horizontalement en haut de l'élément, avec la valeur `repeat-y`, elle n'est répétée que verticalement sur le côté gauche de l'élément. La valeur `no-repeat` empêche quant à elle toute répétition. Dans ce cas, et si d'autres propriétés ne donnent pas de directives contraires, l'image est placée en haut et à gauche de l'espace occupé par l'élément. La valeur `inherit` permet de forcer l'application de la valeur définie pour l'élément parent car la propriété `background-repeat` n'est pas héritée par défaut.

Dans l'exemple 10-4, nous définissons une couleur, une image de fond et la répétition de l'image selon l'axe des abscisses, pour les éléments `<body>` et `<h2>` (repères ³ et ·). L'image de fond de `<body>` apparaît donc en haut de la page comme s'il s'agissait de celle de `<h1>` (repère ») qui n'en est pas muni. Les deux éléments `<h2>` (repères ¿ et ´) présents dans la page ayant à la fois une couleur et une image de fond avec une répétition horizontale, l'image s'affiche en haut du titre et la couleur de fond occupe le reste de l'espace alloué aux titres.

Exemple 10-4. Répétition de l'image de fond

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Positionnement des images de fond</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" >
   body{color: black;background-color:#FFF;background-image:url(fondbleu2.gif);
    ➥ background-repeat:repeat-x;}
   h2{background-color:yellow;background-image:url(fondjaune.gif);
    ➥ background-repeat:repeat-x;}
  </style>
 </head>
 <body>
  <h1>XHTML 1.1 et CSS 2.1</h1>
  <h2>XHTML 1.1 <br />Le langage de création de pages web</h2>
  <p> In principio creavit Deus caelum et terram terra autem erat inanis et vacua
   ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas </p>
  <h2>CSS 2.1<br />L'outil de création des styles</h2>
  <p> In principio creavit Deus caelum et terram terra autem erat inanis et vacua
   ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas </p>
 </body>
</html>
```

La figure 10-4 montre l'aspect obtenu par ces définitions.

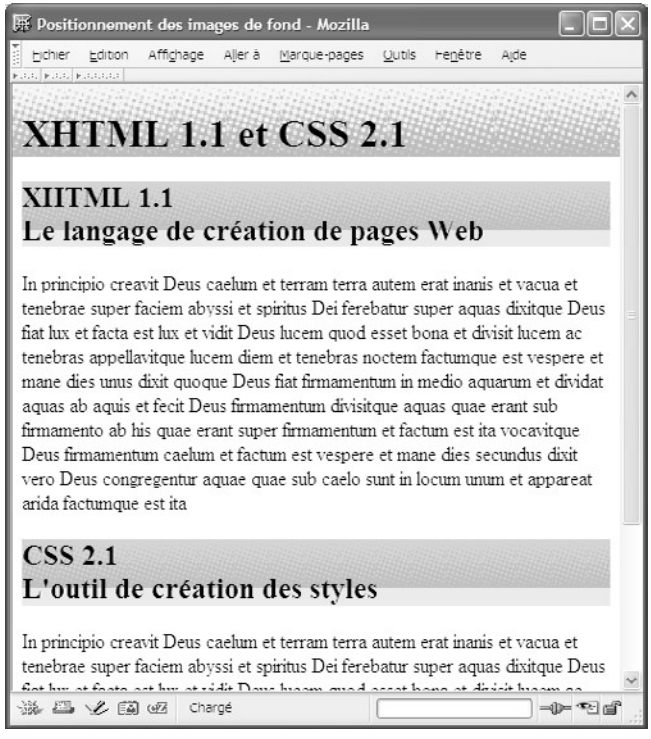


Figure 10-4
Répétition horizontale de l'image de fond

En remplaçant les styles de l'exemple 10-4 par les suivants :

```
body{color: black;background-color:#FFF;background-image:url(fondbleu2.gif);
➥ background-repeat:repeat-y;}
h2{background-color:yellow;background-image:url(fondjaune.gif);
➥ background-repeat:repeat-y;}
```

nous obtenons le résultat présenté à la figure 10-5, qui prouve une fois de plus toute la puissance de CSS car, pour obtenir ces résultats très différents l'un de l'autre, nous n'avons modifié que les deux valeurs de la propriété `background-repeat` (dans le cas présent, un seul caractère a changé dans chaque ligne).

En utilisant la valeur `no-repeat`, l'image de fond sera placée dans l'angle supérieur gauche de la zone de contenu de l'élément. Si cette position par défaut ne convient pas, il est possible d'en définir une autre explicitement à l'aide de la propriété `background-position` dont la syntaxe ci-après offre de nombreuses possibilités :

```
background-position:[[<pourcent> | <long> | left | center | right][<pourcent> |
➥ <long> | top | center | bottom]?] | [left | center | right]||[top | center |
➥ bottom]]| inherit
```

Cette syntaxe, plutôt complexe, mérite quelques explications. La première partie :

```
[<pourcent> | <long> | left | center | right]
```

Figure 10-5
Répétition verticale des images de fond

signifie que la position horizontale peut être définie au minimum par un pourcentage, une unité de longueur ou un des mots-clés `left`, `center` ou `right` dont nous allons donner la signification. La partie suivante :

- [<pourcent> | <long> | top | center | bottom]

indique qu'il est possible, mais facultatif, de définir également la position verticale avec les mêmes unités et les mots-clés `top`, `center` et `bottom`. Si la position est fixée au moyen d'une unité de longueur, elle est calculée par rapport aux bords gauche et haut. Quand elle est donnée en pourcentage, celui-ci est calculé par rapport à la largeur de l'élément pour l'alignement horizontal et par rapport à sa hauteur pour l'alignement vertical.

Si la position donnée en pixel ou en pourcentage, est faite avec une valeur négative, l'image de fond est coupée sur ses côtés gauche ou droit et haut ou bas selon le positionnement choisi. Elle n'apparaît donc que partiellement, ce qui peut entraîner des effets particuliers. Associée à une boucle en JavaScript, cette possibilité peut permettre de faire apparaître l'image progressivement, à partir d'un bord de l'élément auquel elle s'applique.

La dernière partie de la syntaxe :

- [left | center | right]||[top | center | bottom]

indique qu'il est aussi envisageable de n'utiliser que des mots-clés. Dans ce cas, nous avons le choix d'indiquer un seul mot-clé pour la position horizontale, ou deux, donnant

dans l'ordre la position horizontale suivie de la position verticale. Le tableau 10-1 indique la signification des différents mots-clés.

La propriété `background-position` n'étant pas héritée, la valeur `inherit` permet d'appliquer la même valeur que celle de l'élément parent.

Tableau 10-1. Les mots-clés de positionnement des images de fond

Position horizontale	
left	L'image est placée à gauche dans la zone de contenu de l'élément.
center	L'image est centrée dans la zone de contenu de l'élément.
right	L'image est placée à droite dans la zone de contenu de l'élément.
Position verticale	
top	L'image est placée en haut de la zone de contenu de l'élément.
center	L'image est centrée verticalement dans la zone de contenu de l'élément.
bottom	L'image est placée en bas de la zone de contenu de l'élément.

Comme l'indique la syntaxe, il est possible de définir un seul alignement (horizontal ou vertical) ; dans ce cas, l'autre est défini par défaut à la valeur `center`. Il est plus clair de définir les deux à la fois, en donnant deux valeurs successives.

Le tableau 10-2 donne l'ensemble des neuf combinaisons possibles permettant d'utiliser deux mots-clés simultanément, et les équivalences en pourcentage, et avec l'usage d'un seul mot-clé. La dernière colonne de ce tableau donne les repères des résultats obtenus pour chaque valeur, tels qu'ils sont visibles à la figure 10-6. Les valeurs `left center`, `0% 50%` `left` et `0%` sont par exemple toutes équivalentes.

Tableau 10-2. Équivalence des mots-clés

2 mots-clés	2 pourcentages	1 seul mot-clé	1 seul pourcentage	Repères de la figure 10-6
left top	0 % 0 %	non	-	1
center top	50 % 0 %	top	-	2
right top	100 % 0 %	non	-	3
left center	0 % 50 %	left	0 %	4
center center	50 % 50 %	center	50 %	5
right center	100 % 50 %	right	100 %	6
left bottom	0 % 50 %	non	-	7
center bottom	50 % 100 %	bottom	-	8
right bottom	100 % 100 %	non	-	9

Les codes de l'exemple 10-5 indiquent les définitions de styles de toutes les possibilités représentées à la figure 10-6 en utilisant des mots-clés. Les repères ³ à ¾ correspondent à ceux de la figure 10-6.

Exemple 10-5. Le positionnement des images de fond

```
³ body{background-image:url(pont.png);background-repeat:no-repeat;
➥ background-position:left top;}
```

```
· body{background-image:url(pont.png);background-repeat:no-repeat;
➥ background-position:center top;}
» body{background-image:url(pont.png);background-repeat:no-repeat;
➥ background-position:right top;}
¿ body{background-image:url(pont.png);background-repeat:no-repeat;
➥ background-position:left center;}
´ body{background-image:url(pont.png);background-repeat:no-repeat;
➥ background-position:center center;}
² body{background-image:url(pont.png);background-repeat:no-repeat;
➥ background-position:right center;}
¶ body{background-image:url(pont.png);background-repeat:no-repeat;
➥ background-position:left bottom;}
º body{background-image:url(pont.png);background-repeat:no-repeat;
➥ background-position:center bottom;}
¾ body{background-image:url(pont.png);background-repeat:no-repeat;
➥ background-position:right bottom;}
```

Figure 10-6
Différents positionnements de l'image de fond

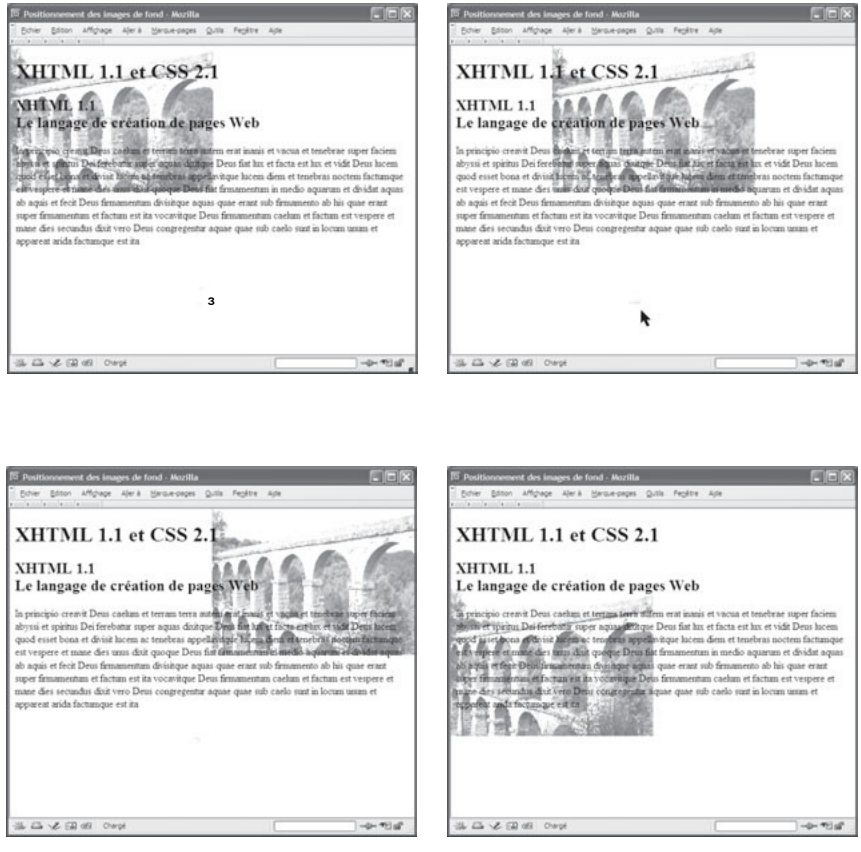
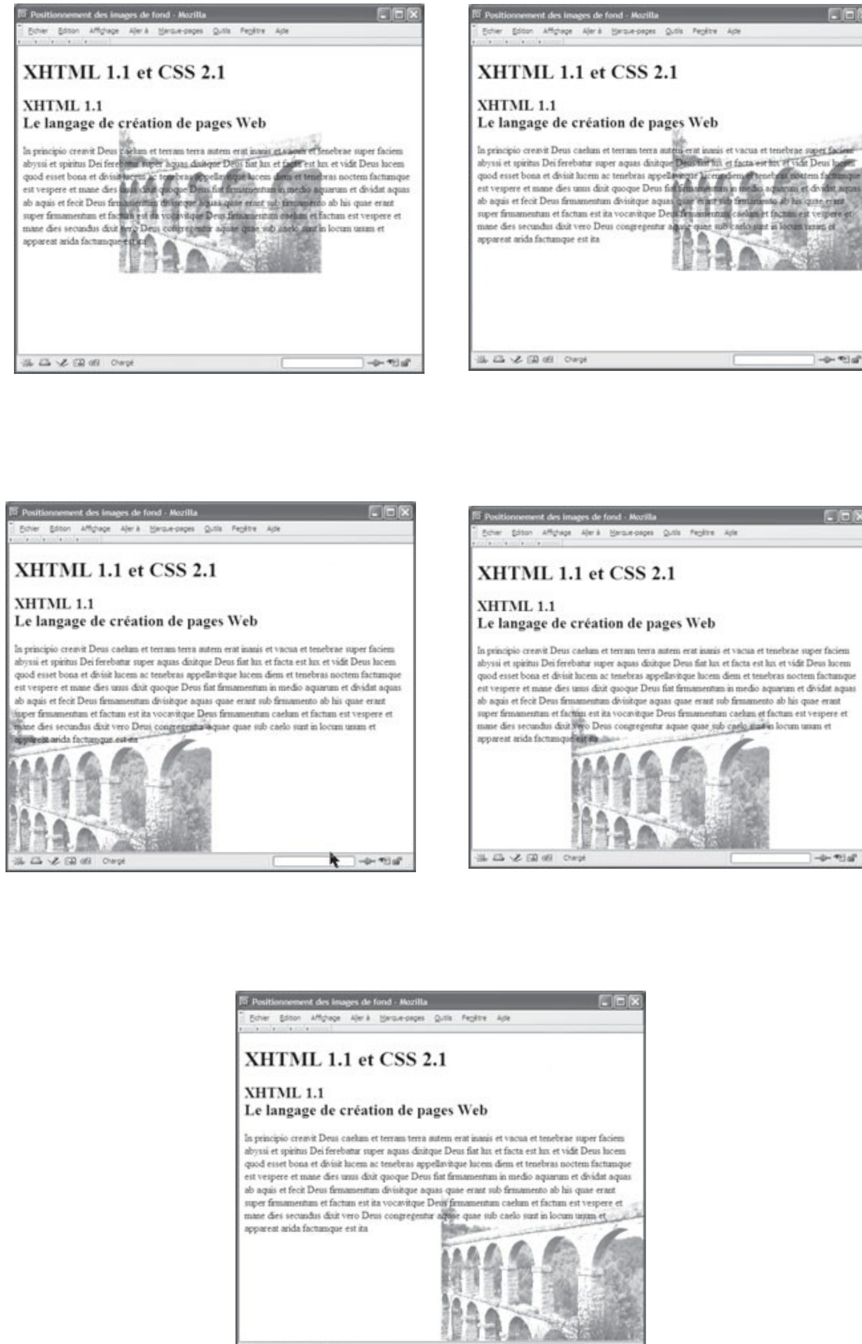


Figure 10-6

Différents positionnements de l'image de fond (suite)



Pour certains éléments, le plus évident étant `<body>` quand le contenu est plus haut que la fenêtre, des barres de défilement apparaissent automatiquement dans le navigateur. Si l'élément possède une image de fond, celle-ci va par défaut défiler avec le reste de la page et elle peut donc disparaître, en particulier si elle est positionnée en haut ou même au centre de la page. Il est possible de choisir si l'on veut permettre ou non ce défilement en utilisant la propriété `background-attachement` dont la syntaxe est :

```
background-attachement:scroll|fixed|inherit
```

Si la valeur `scroll` correspond au comportement par défaut qui implique le défilement de l'image avec le contenu, la valeur `fixed` permet de conserver l'image à sa position initiale, définie par `background-position`. Avec la valeur `fixed` nous pourrons par exemple conserver une image centrée dans la fenêtre quelle que soit la hauteur du contenu d'une page. La propriété n'étant pas héritée par défaut par un élément enfant, la valeur `inherit` permet de définir cet héritage.

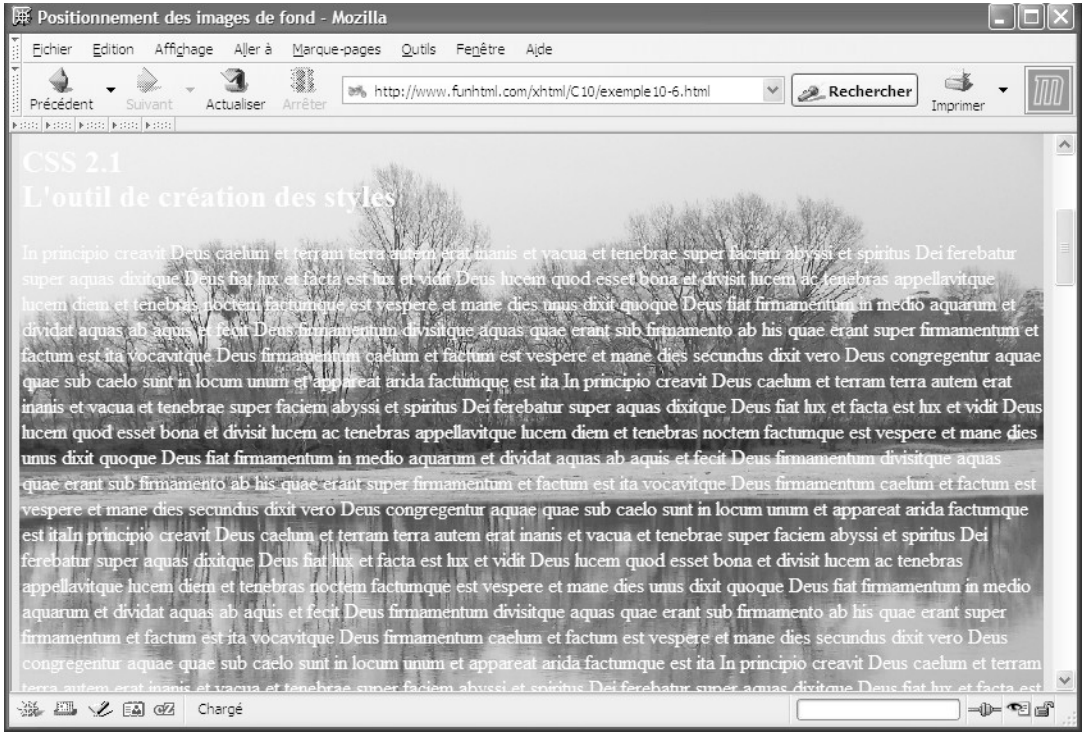


Figure 10-7
Une image de fond fixe

À titre d'exemple, en utilisant le même code XHTML que celui de l'exemple 10-4 pour définir le contenu d'une page, et, en appliquant le style suivant à l'élément `<body>` pour fixer l'image de fond, nous obtenons le résultat présenté à la figure 10-7 après avoir

fait défiler son contenu. Nous pouvons constater que l'image de fond conserve sa position initiale malgré le défilement opéré sur le texte de la page.

```
body{background-image:url(paysage.png); background-repeat:no-repeat;
➥ background-position:top center; background-attachment: fixed; color:#BBB; }
```

En revanche, en définissant les styles suivants qui autorisent le défilement de l'image, nous obtenons le résultat présenté à la figure 10-8. Nous y remarquons que le paysage de fond s'est déplacé avec le texte auquel il est lié.

```
body{background-image:url(paysage.png); background-repeat:no-repeat;
➥ background-position:top center; background-attachment:scroll;
➥ background-color:#BBB; color:white; }
```

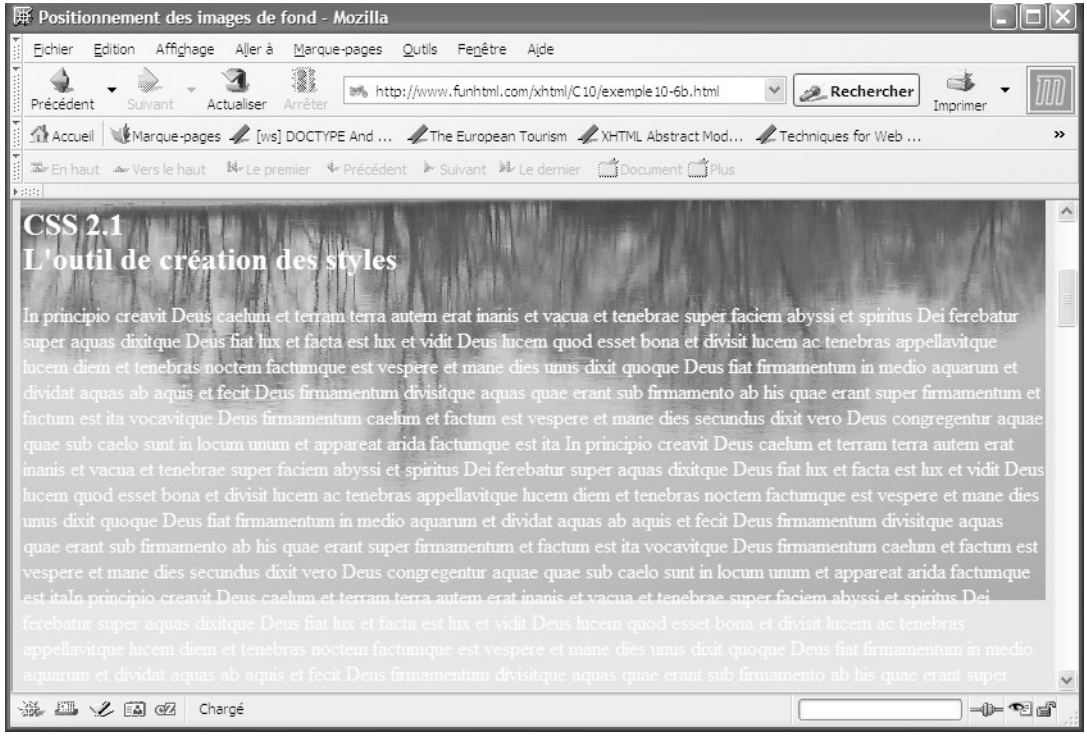


Figure 10-8
Une image de fond défilante

Un raccourci pour les fonds

Plutôt que de définir individuellement chacune des caractéristiques précédentes de couleur, d'image, de position et de défilement pour le fond d'un élément, il est possible de réaliser ces mêmes définitions à l'aide d'une seule propriété nommée `background` qui va toutes les rassembler selon la syntaxe suivante :

```
background:[<background-color>||<background-image>||<background-repeat>||
➥ <background-attachement>||<background-position>]|inherit
```

Dans cette propriété, nous pouvons définir de une à cinq valeurs qui indiquent respectivement pour le fond, sa couleur, l'image de fond, le mode de répétition de celle-ci, son défilement et sa position. L'ordre de définition de chaque valeur n'est pas important, mais chaque indication doit respecter la syntaxe donnée pour les différentes propriétés individuelles que nous avons vues précédemment. À la place de l'ensemble de ces valeurs, il est possible d'écrire le mot-clé `inherit` pour que l'élément ait toutes les caractéristiques de son parent. Il est donc possible de définir par exemple le style suivant :

```
body{background:url(paysage.png) no-repeat top center scroll #BBB;}
```

en lieu et place de l'ensemble des propriétés individuelles :

```
body{background-image:url(paysage.png); background-repeat:no-repeat;
➥ background-position:top center; background-attachment:scroll;
➥ background-color:#BBB;}
```

Exercices

Exercice 1 : Quels sont les différents effets de la définition de la propriété `color` ?

Exercice 2 : Créez les styles pour que les titres `<h1>` soient affichés en rouge sombre, les titres `<h2>` en bleu et les paragraphes en gris moyen. Pour définir les couleurs, utilisez successivement des mots-clés, des codes hexadécimaux et la fonction `rgb()`. Appliquez ces styles à un document.

Exercice 3 : Créez les styles afin que le texte des paragraphes qui suivent un titre `<h1>` soit bleu et que celui des paragraphes qui suivent un titre `<h2>` soit vert. Appliquez ces styles à un document.

Exercice 4 : Créez une page à fond jaune clair et dont le texte soit bleu foncé.

Exercice 5 : Créez une page à fond bleu pale dont le texte par défaut soit noir. Les titres `<h1>` doivent être écrits en rouge sur fond jaune et les titres `<h2>` en bleu foncé sur fond transparent.

Exercice 6 : Créez le style permettant d'afficher le contenu des éléments `<code>` en vert vif sur fond noir, à l'image des antiques terminaux monochromes.

Exercice 7 : Créez les styles de façon à ce que seuls les paragraphes ayant un attribut `id` soient affichés avec un texte gris sur fond rose pale. Les autres paragraphes doivent être affichés en noir sur blanc.

Exercice 8 : Définissez une image de fond de petit format qui soit répétée horizontalement et verticalement dans toute la page. Tous les autres éléments de la page, quels qu'ils soient, doivent avoir un fond blanc.

Exercice 9 : Créez une page dont l'image de fond soit non répétée et située au centre et en haut de la fenêtre.

Exercice 10 : Écrivez un style de façon que n'apparaisse que la moitié inférieure de l'image de fond choisie. Celle-ci doit être placée en haut et au centre de la page.

Exercice 11 : Créez un style pour que l'image de fond (de petite taille de préférence pour créer un motif) de tous les éléments d'une page soit centrée et répétée verticalement.

Exercice 12 : Placez une image de fond dans une page à 15 % du bord gauche et à 30 % du bord supérieur.

Exercice 13 : Fixez l'image de fond de l'exemple précédent pour qu'elle ne défile pas. Qu'observe-t-on quand on redimensionne la fenêtre du navigateur ?

Exercice 14 : Écrivez tous les styles des exercices précédents en utilisant la propriété raccourcie `background` quand cela est possible.

11

Créer des bordures, marges, espacements et contours

CSS définit un espace dans la page pour chaque élément de la hiérarchie du document XHTML. C'est ce que l'on nomme communément le modèle de boîte de CSS. La boîte associée à chaque élément conditionne la surface, et éventuellement la position qu'il a dans la page, et donc, la position des boîtes des autres éléments par rapport à lui. Dans le modèle de boîte, chaque élément visuel a un contenu direct ou indirect que nous pouvons dimensionner, mais il peut également avoir un espacement, une bordure et une marge par rapport aux boîtes des éléments environnants.

Le modèle CSS des boîtes

À chaque élément XHTML visuel correspond dans le navigateur une zone rectangulaire nommée boîte. Que l'élément soit du type bloc comme `<div>` `<p>` ou `<form>` ou de type en ligne comme `<img />`, il se voit donc attribuer sa propre boîte. Il est important de savoir comment les navigateurs gèrent ces boîtes les unes par rapport aux autres et comment est déterminé leur encombrement total à l'écran. La figure 11-1 présente l'organisation du modèle de boîte de CSS 2.

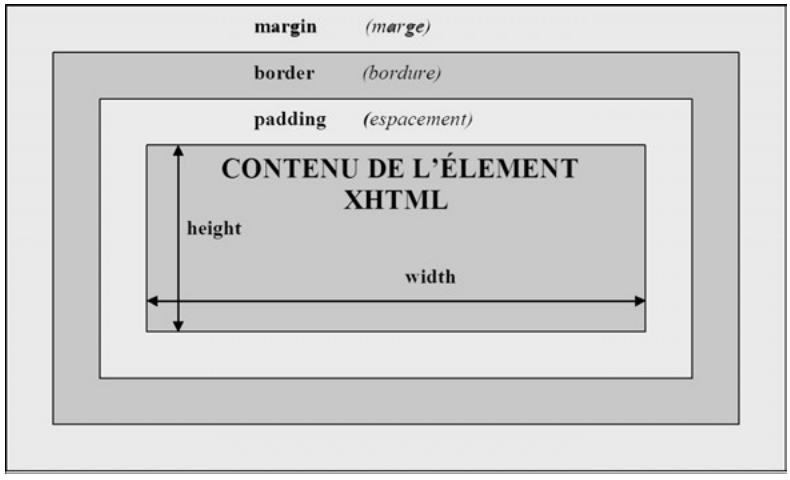


Figure 11-1
Le modèle de boîte CSS 2

En examinant cette figure, nous remarquons les points suivants :

- La boîte la plus interne est celle du contenu de l'élément. Il peut s'agir d'un texte dans un paragraphe, d'une image, ou d'éléments de formulaires par exemple. Nous pouvons agir sur les dimensions du contenu à l'aide des propriétés CSS `width` et `height`. Ces propriétés s'appliquent à tous les éléments, sauf à ceux qui sont à la fois de type en ligne et non remplacés, ainsi qu'aux éléments `<col />` et `<colgroup>`. La propriété `width` définit la largeur de la boîte du contenu et `height` sa hauteur. Ces dimensions peuvent être données à l'aide d'une unité de longueur habituelle (em, ex, px, mm, cm, in, pc, pt) ou en pourcentage. Pour la propriété `width`, ce pourcentage sera calculé par rapport à la largeur de l'élément parent de celui dont nous définissons la largeur. Si l'élément est un bloc directement inclus dans `<body>` il s'agit de la largeur de la fenêtre du navigateur. Pour la propriété `height`, il y a lieu de prendre quelques précautions car les dimensions précisées ne sont pas nécessairement respectées dans le navigateur, en particulier dans les cas où le contenu est plus haut que la hauteur spécifiée. Nous reviendrons en détail sur les règles de dimensionnement des éléments et la manière dont elles sont gérées par CSS au chapitre 13.
- Autour de la boîte du contenu, un espacement (*padding*) peut être défini. Il permet de créer un espace vide entre le contenu et sa bordure. Si l'espacement est nul, ses limites extérieures sont confondues avec celles de la boîte de contenu. L'espacement a toujours une couleur de fond identique à celle du contenu.
- Encadrant la boîte précédente, on trouve la bordure (*border*) de l'élément qui crée une troisième boîte incluant les deux précédentes. Là aussi, si la bordure a une largeur nulle, les côtés de la boîte sont confondus avec ceux de l'espacement. La bordure peut être dotée d'une couleur, d'un style, et bien sûr d'une épaisseur.
- En fin, autour de la bordure une marge (*margin*) peut être définie, laquelle crée un espace vide entre la boîte précédente et les boîtes des éléments voisins dans la page, et

donc dans le code XHTML. Nous allons maintenant faire le tour de ces différents éléments de présentation. Pour déterminer l'encombrement total d'un élément dans le navigateur, il ne faudra pas perdre de vue que les dimensions de tous ces éléments s'additionnent. Si, par exemple, nous définissons une largeur de 500 pixels pour un élément, un espacement de 20 pixels, une bordure de 5 pixels et une marge de 15 pixels, le rendu visuel de l'élément aura une largeur totale de 15 + 5 + 20 + 500 + 20 + 5 + 15 = 580 pixels. Les anciennes versions de certains navigateurs très connus prenaient comme dimension totale celle qui avait été définie avec la propriété `width` et réduisaient donc d'autant le contenu pour placer marge, bordure et espacement. Les navigateurs modernes réalisent bien l'addition de toutes les valeurs et il ne faut donc pas oublier d'en tenir compte pour éviter les débordements par rapport à la fenêtre du navigateur, ce qui obligerait l'utilisateur à faire défiler la page en largeur, ce qui est désagréable et fait perdre du temps.

Les bordures

La création de bordures autour d'un élément permet d'obtenir des effets visuels qui attirent l'attention et permettent également par exemple de mettre davantage en évidence la structure de la page. En CSS, nous pouvons attribuer à une bordure des caractéristiques variées, telles qu'un style, une largeur et une couleur.

Le style de la bordure

CSS offre de nombreuses possibilités de style pour les bordures grâce à la propriété `border-style`. Elle peut prendre un grand nombre de valeurs auxquelles sont attachés des effets visuels divers et variés propres à satisfaire les designers de site. Cette propriété s'applique à tous les éléments visuels et sa syntaxe est la suivante :

```
border-style : [none | hidden | <style>] {1,4} | inherit
```

La valeur `none` indique qu'il n'y a pas de bordure. La valeur `hidden` donne le même résultat pour la plupart des éléments sauf les cellules de tableaux. Avec la valeur `inherit`, la bordure a explicitement le même style que son élément parent. Le paramètre `style` peut prendre une des valeurs suivantes :

- `dotted` : bordure en pointillés courts (figure 11-2, repère ³) ;
- `dashed` bordure en tirets longs (figure 11-2, repère ·) ;
- `solid` : bordure pleine continue (figure 11-2, repère ») ;
- `double`: bordure constituée de deux traits parallèles continus (figure 11-2, repère ¿). Si la largeur de bordure est insuffisante, un seul trait apparaît (pour définir la largeur, voir la propriété `border-width`) ;
- `groove`: bordure en creux. L'effet de creux est créé par l'emploi de couleurs différentes pour les côtés (figure 11-2, repère ´) ;

- ridge : bordure en relief. Même remarque sur la couleur des côtés (figure 11-2, repère ²) ;
- inset : bordure en creux dont chaque côté n'a qu'une seule couleur (figure 11-2, repère ¶) ;
- outset : bordure en relief dont chaque côté n'a qu'une seule couleur (figure 11-2, repère º).

La notation {1,4} précise qu'il est possible d'établir de 1 à 4 valeurs si l'on veut définir séparément les styles des bordures haute, droite, basse et gauche, dans cet ordre (pour mémoriser cet ordre, pensez au sens des aiguilles d'une montre en partant de midi). Cette multiplication des valeurs obéit aux règles suivantes :

- En donnant une seule valeur, elle s'applique aux quatre côtés.
- En donnant deux valeurs, la première s'applique aux côtés haut et bas, et la seconde aux côtés droit et gauche.
- En donnant trois valeurs, la première s'applique au côté haut, la suivante aux côtés droit et gauche, et la dernière au bord bas.

Si vous voulez éviter de mémoriser toutes ces règles, il vaut mieux définir soit une seule, soit quatre valeurs explicitement, le code étant alors immédiatement plus lisible.

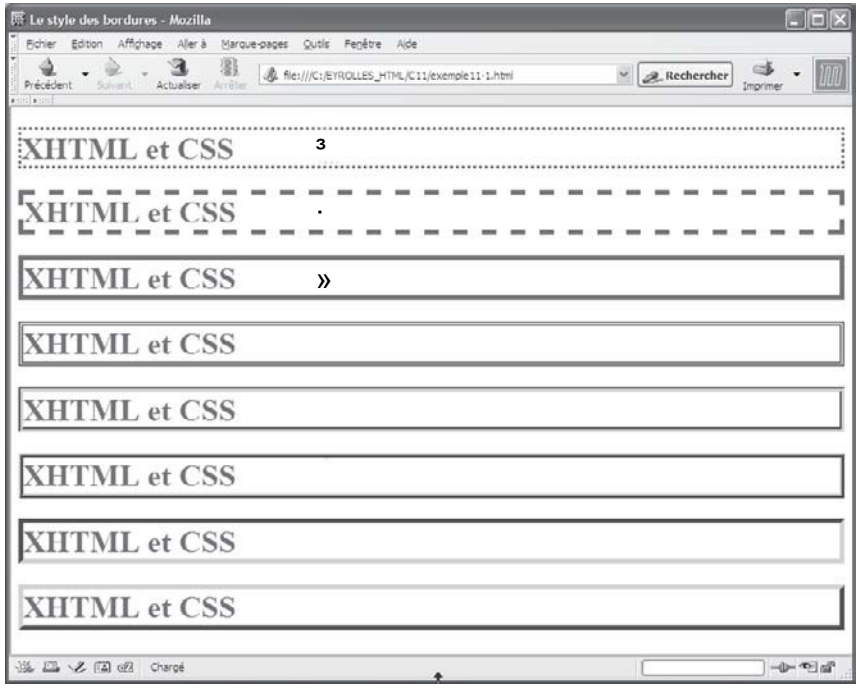
L'exemple 11-1 contient la définition de la couleur d'avant-plan qui est aussi celle des bordures (repère ³). Vient ensuite la définition de huit classes pour les éléments <h1> (repères · à ¾) qui permet de visualiser tous les styles de bordures possibles.

Exemple 11-1. Les styles de bordures

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les styles de bordures</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="bordure">
   body{color:#555;} ³
   h1.dotted{border-style: dotted;}    ·
   h1.dashed{border-style: dashed;} »
   h1.solid{border-style: solid; border-width: thick;}      ¿
   h1.double{border-style: double; border-width: thick;}    ´
   h1.groove{border-style: groove; border-width: thick;}    ²
   h1.ridge{border-style: ridge; border-width: thick;}      ¶
   h1.inset{border-style: inset; border-width: thick;}      º
   h1.outset{border-style: outset; border-width: thick;}    ¾
  </style>
 </head>
```

```
 <body>
  <h1 class="dotted">XHTML et CSS</h1>
  <h1 class="dashed">XHTML et CSS</h1>
  <h1 class="solid">XHTML et CSS</h1>
  <h1 class="double">XHTML et CSS</h1>
  <h1 class="groove">XHTML et CSS</h1>
  <h1 class="ridge">XHTML et CSS</h1>
  <h1 class="inset">XHTML et CSS</h1>
  <h1 class="outset">XHTML et CSS</h1>
 </body>
</html>
```

Figure 11-2
Les différents styles de bordures



Nous pouvons définir individuellement chacune des bordures d'un élément en utilisant les propriétés suivantes :

- border-top-style : définit le style de la bordure haute ;
- border-right-style : définit le style de la bordure droite ;
- border-bottom-style : définit le style de la bordure basse ;
- border-left-style : définit le style de la bordure gauche.

Dans l'exemple 11-2, nous définissons avec les propriétés susmentionnées quatre styles différents pour les bordures des titres <h1> (repère ³). Nous procédons de même en

définissant deux styles différents pour les bordures gauche et droite des paragraphes <p> (repère ·). L'utilisation de la pseudo-classe :hover permet de modifier individuellement chacune de ces bordures quand le curseur survole la boîte du paragraphe (repère »).

Exemple 11-2. Les bordures individuelles

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les bordures individuelles</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="bordures">
   h1{border-top-style: dotted; border-right-style: dashed;
     ➥ border-bottom-style: double; border-left-style: solid;}
   p{border-left-style: inset; border-right-style: dashed;}
   p:hover{border-top-style: dotted ; border-bottom-style: double;}
  </style>
 </head>
 <body>
  <h1>XHTML et CSS</h1>
  <p> In principio creavit Deus caelum et terram terra autem erat inanis
    ➥ et vacua . . .</p>
 </body>
</html>
```

La figure 11-3 illustre le résultat obtenu lors du survol du paragraphe.

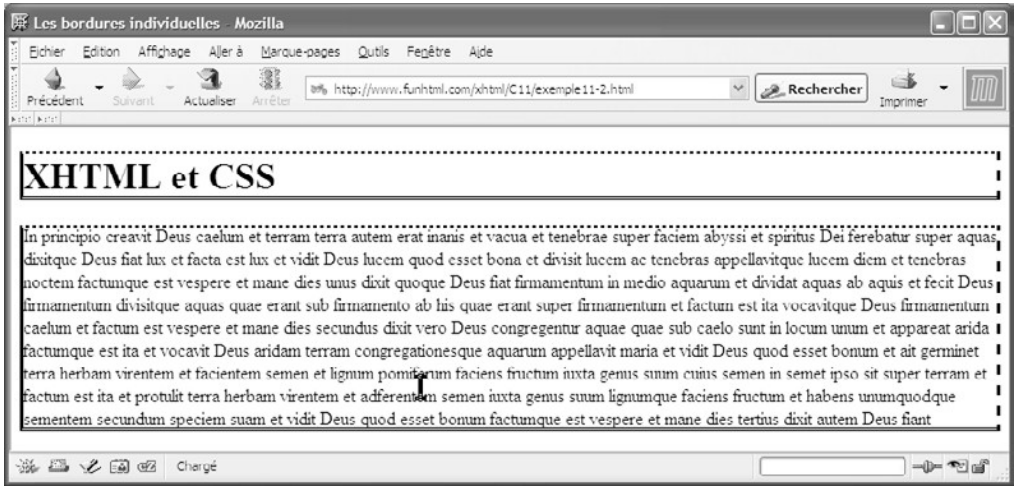


Figure 11-3
Les bordures individuelles

La largeur de la bordure

En donnant un style aux bordures, elles prennent une largeur fixe comme nous l'avons constaté à la figure 11-2. Nous pouvons donner une largeur particulière aux bordures d'un élément grâce à la propriété `border-width`, dont la syntaxe est la suivante :

```
border-width: <bord>{1,4} | inherit
```

Le paramètre `<bord>` peut être défini de plusieurs façons :

- Par un mot-clé, parmi les suivants :
  - `thin` : pour une bordure mince. Avec cette valeur, les effets de style larges comme `double` risquent de ne pas être nettement perceptibles.
  - `medium` pour une bordure moyenne (c'est la valeur par défaut).
  - `thick` : pour une bordure épaisse.

Ces valeurs ne correspondent pas à une épaisseur absolue et celle-ci peut varier selon les navigateurs.

- Par une valeur explicite de longueur, définie habituellement par un nombre (positif) et une unité en px, em, ex, cm, mm, in, pc, pt.
- Par le mot-clé `inherit` si l'on veut la même valeur que celle de l'élément parent.

Si une seule valeur est donnée, elle s'applique aux quatre côtés de la bordure. La définition du paramètre peut aussi contenir de une à quatre valeurs qui peuvent s'appliquer séparément aux bordures haute, droite, basse et gauche dans cet ordre. Cette possibilité obéit aux mêmes règles d'ordre d'affectation, que nous avons indiquées pour la propriété `border-style`, quand on définit deux, trois ou quatre valeurs.

Les meilleurs effets visuels sont souvent obtenus en ne définissant une bordure que sur un ou deux côtés de la boîte de contenu. On procède à ces définitions à l'aide des propriétés suivantes :

- `border-top-width` : pour la bordure haute ;
- `border-right-width` : pour la bordure droite ;
- `border-bottom-width` : pour la bordure basse ;
- `border-left-width` : pour la bordure gauche.

Leur syntaxe est la même que celle de la propriété `border-width` mais ne doit bien sûr comporter qu'une seule valeur.

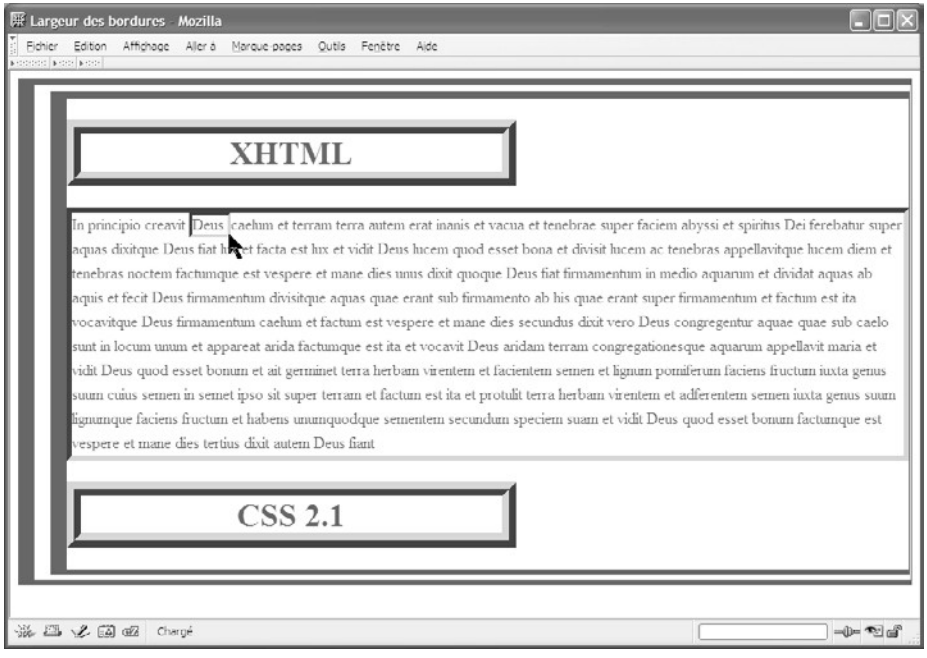
L'exemple 11-3 illustre les différentes possibilités de définition de la largeur des bordures. Le premier style définit la couleur d'avant-plan, et donc également celle des bordures

(repère ³). Pour l'élément , la largeur des bordures est fixée à la valeur double et des largeurs différentes sont créées pour les côtés haut, bas et gauche au moyen d'unités différentes (repère ·). Pour les paragraphes (repère º), la largeur est définie à l'aide du mot-clé thick (repère ») tandis que pour l'élément (repère ¾), qui est inclus dans <p>, la largeur est fixée à la valeur medium (repère ¿). Pour cet élément, le style des bordures est modifié dynamiquement à l'aide d'un code JavaScript et passe de inset, quand le curseur survole le texte (repère µ), à outset, quand il le quitte (repère ¸). Pour les titres <h1> (repères ¶ et ¹), la largeur est définie en unités ex et le style utilise la propriété width pour limiter la largeur de la boîte associée à l'élément à 50 % de celle de son parent (repère ´).

Exemple 11-3. La largeur des bordures

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Largeur des bordures</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="bordures">
   body{color:#444;} ³
   div{border-style: double; border-top-width: 20px ; border-bottom-width: 15px;
    ➥ border-left-width: 3em;} ·
   p{border-style: inset; border-width: thick; line-height: 1.5em;}      »
   span{border-style: outset; border-width: medium;} ¿
   h1{border-style: ridge  ; border-width: 1ex; text-align: center; width:50%;}      ´
  </style>
 </head>
 <body>
  ²  <div>
   ¶  <h1>XHTML</h1>
   º  <p> In principio creavit  ¾ <span µ  onmouseover= "this.style.
    ➥ borderStyle='inset'"  ¸  onmouseout="this.style.borderStyle='outset'">
    ➥ Deus </span> caelum et terram terra autem erat inanis et vacua et tenebrae
    ➥ super faciem abyssi et spiritus Dei ferebatur super aquas
   </p>
   ¹  <h1>CSS 2.1</h1>
  </div>
 </body>
</html>
```

Figure 11-4
La largeur des bordures



La couleur de la bordure

Afin d'améliorer encore l'aspect des bordures, nous pouvons définir une couleur pour chaque bordure d'élément avec la propriété `border-color` dont la syntaxe est la suivante :

```
border-color: [<couleur> | transparent]{1,4} | inherit
```

Le paramètre `<couleur>` est donné par un code de couleur hexadécimal précédé du symbole dièse `(#)`, un mot-clé (voir annexe C) ou les composantes RGB de la couleur données comme paramètres de la fonction `rgb()`. Le mot-clé `transparent` indique que la bordure sera invisible et laissera voir la couleur du fond, ce qui peut constituer un effet particulier ou simplement créer une marge autour du contenu. Comme pour la propriété `border-width`, l'indication `{1,4}` précise qu'il est possible de définir individuellement la couleur des bordures des côtés haut, droit, bas et gauche avec les mêmes conditions d'attribution en tournant dans le sens des aiguilles d'une montre. Il est à noter que si nous définissons une largeur de bordure avec la propriété `border-width`, mais pas sa couleur, le navigateur crée une bordure dont la couleur est celle de la propriété `color` attribuée au texte. Mais il est encore possible de définir individuellement la couleur de chaque côté de la bordure à l'aide des propriétés suivantes :

- `border-top-color` : pour la couleur de la bordure haute ;
- `border-right-color` : pour la couleur de la bordure droite ;
- `border-bottom-color` : pour la couleur de la bordure basse ;
- `border-left-color` : pour la couleur de la bordure gauche.

En modifiant les définitions des styles de l'exemple 11-3 afin d'obtenir pour les bordures des couleurs personnalisées et qui soient différentes de la couleur d'avant-plan définie pour l'élément <body> (repère ³), nous obtenons le code de l'exemple 11-4. Les paragraphes ont alors des bordures jaunes (repère ·). Pour l'élément la propriété border-color est définie avec les valeurs red, blue et yellow , et nous obtenons donc une bordure haute rouge, des bordures droite et gauche bleues et une bordure basse jaune (repère »). L'élément a encore une bordure outset dont la couleur est maintenant bleue (repère ¿). Quant aux titres <h1> leur couleur de bordure est définie avec la fonction rgb() (rgb(25,255,50)) en vert (repère ´).

Exemple 11-4. La couleur des bordures

```
<style type="text/css" title="bordures">
 body{color:#444;} ³
 p{border-style: inset; border-width: thick; line-height: 1.5em; border-color:
  ➥ yellow;} ·
 div{border-style: double; border-top-width: 20px ; border-bottom-width: 15px;
  ➥ border-left-width: 3em; border-color: red blue yellow;} »
 span{border-style: outset; border-width: medium; border-color: #1533FF;} ¿
 h1{border-style: ridge; border-width: 1ex; text-align: center; width: 50%;
  ➥ border-color: rgb(25,255,50);} ´
 </style>
```

Définition globale d'une bordure

Pour définir les caractéristiques d'une bordure, plutôt que d'utiliser comme précédemment trois propriétés CSS, nous pouvons utiliser la propriété border, dont la syntaxe est la suivante :

```
border: [<largeur>||<style>||[<couleur>|transparent]] | inherit
```

Elle s'applique également à tous les éléments XHTML visuels mais elle ne permet pas de définir individuellement chaque côté de la bordure. Les définitions d'épaisseur, de style et de couleur prennent les mêmes valeurs que dans les propriétés border-width, border-style et border-color. L'utilisation de cette propriété raccourcie permet de gagner du temps pour les cas courants dans lesquels la bordure est uniforme sur tous ses côtés. Si nous écrivons par exemple le style suivant :

```
h1{border: 5px double blue;}
```

cela équivaut aux trois définitions suivantes :

```
h1{border-width: 5px border-style: double; border-color: blue;}
```

Il est encore possible d'affiner les définitions individuelles de chaque côté de la bordure d'un élément en utilisant les propriétés suivantes dont la syntaxe est la même que celle de la propriété border. Elles permettent de définir en une seule opération toutes les caractéristiques d'un côté de la bordure :

- border-top : définit la bordure haute ;

- border-right : définit la bordure droite ;
- border-bottom : définit la bordure basse ;
- border-left : définit la bordure gauche.

Ces propriétés raccourcies sont plus simples d'utilisation que les définitions séparées des trois caractéristiques. Nous réserverons ces dernières pour les cas où nous voudrions créer des effets dynamiques en réponse à une action du visiteur afin de ne modifier qu'une seule des caractéristiques d'une bordure.

L'exemple 11-5 illustre ce propos. Nous définissons tout d'abord globalement la bordure gauche des paragraphes (repères ³ et º), puis les bordures haute et basse des divisions (repère ·). Les éléments <h1> (repères ¶ et µ) ont une bordure basse verte de 15 pixels et de style inset (repère »). La pseudo-classe :hover associée à cet élément permet de modifier dynamiquement le style de sa bordure (repère ¿). Tous les navigateurs comme Explorer ne gèrent pas cette pseudo-classe pour tous les éléments. Pour obtenir le même effet, il peut donc être préférable d'utiliser les attributs gestionnaires d'événements onmouseover et onmouseout qui jouent le même rôle, et ainsi effectuer le changement de style au moyen d'un code JavaScript. C'est ce qui est réalisé pour l'élément inclus dans le paragraphe. Nous définissons à l'origine une bordure basse de 3 pixels de style solid . Le gestionnaire onmouseover permet de changer ce style en créant une bordure double quand le curseur survole son contenu, et d'annuler cet effet quand il la quitte (repère ¾).

Exemple 11-5. Définition individuelle des bordures

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Largeur des bordures</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="bordures individuelles">
   p{border-left: 2em dotted blue;}     ³
   div{border-bottom: 10px double red; border-top: 10px groove red;}  ·
   h1{border-bottom: 15px inset rgb(25,255,50); text-align: center;}     »
   h1:hover{border-bottom-style: double;}  ¿
   span{border-bottom: solid red 3px;}  ´
  </style>
 </head>
 <body>
```

```
    <div>
      <h1>XHTML</h1>
      <p> In principio creavit
        <span onmouseover="this.style.borderBottomStyle='double'"
          onmouseout="this.style.borderBottomStyle='solid'">
      caelum et terram terra autem erat inanis et vacua et tenebrae super faciem
      </p>
      <h1>CSS 2.1</h1>
    </div>
  </body>
</html>
```

La figure 11-5 présente le résultat obtenu par la définition de ces bordures et la modification du style du premier titre `<h1>` en cas de survol.

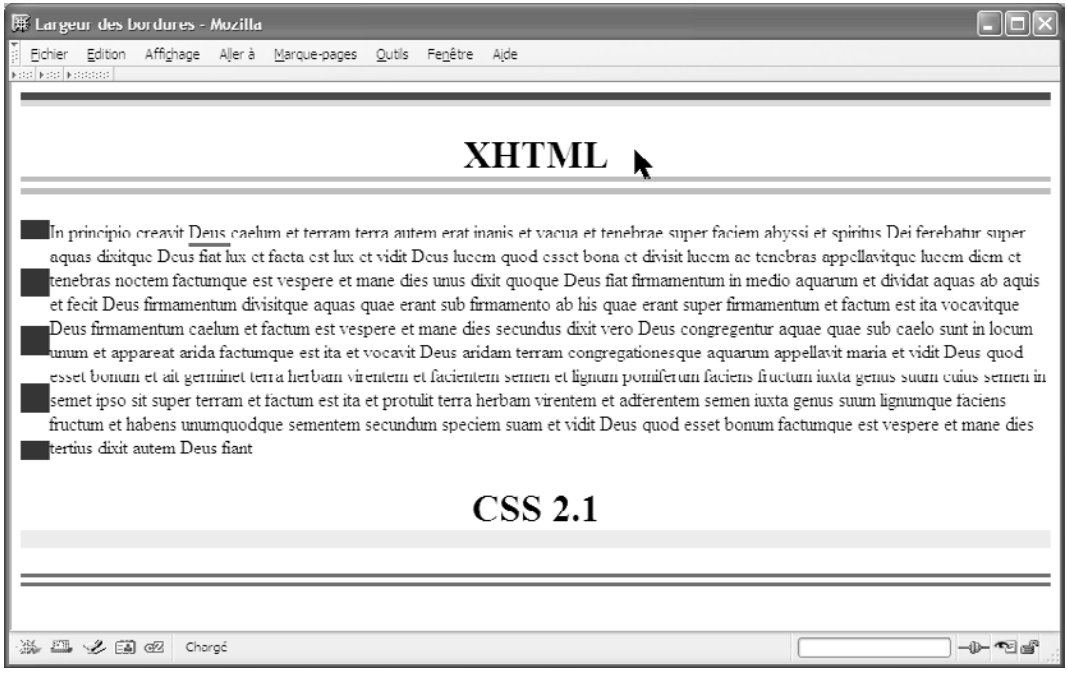


Figure 11-5
Modification dynamique de bordure

Les marges

Afin d'aérer le contenu d'une page et en particulier l'espace entre le rendu d'un élément et ses voisins dans la page, nous pouvons définir une marge autour de chaque élément. Cette dernière est située, d'après le modèle de boîtes CSS illustré à la figure 11-1, à

l'extérieur de la boîte constituée par la bordure de l'élément parent, qu'il s'agisse de <body> ou d'un autre bloc, et il n'est pas possible de la modifier directement sans recourir à quelque astuce.

Pour définir la largeur des marges d'un élément, nous disposons de la propriété margin dont la syntaxe est la suivante :

```
margin: <large>{1,4} | inherit
```

Le paramètre large est un nombre entier suivi d'une unité (px, ex, em, %, mm, cm, in, pc, pt). Les marges peuvent être négatives, et dans ce cas la boîte d'un élément sort de celle de son parent.

Si la largeur de la marge est donnée en pourcentage, elle est calculée par rapport à celle du bloc parent. La notation {1,4} permet ici encore de définir de une à quatre marges dans le sens des aiguilles d'une montre (haut, droit, bas, gauche), et ce avec les mêmes conditions d'affectation si nous ne définissons qu'une, deux ou trois valeurs. La valeur inherit applique la marge de l'élément parent.

Comme dans les cas précédents de bordures, il est possible de définir individuellement chaque marge. Nous utiliserons les propriétés suivantes pour définir par exemple une seule marge (ce qui est aussi possible avec la propriété border en mettant les autres à 0), ou encore pour agir dynamiquement sur une seule des marges.

- margin-top : définit la marge haute ;
- margin-right : définit la marge droite ;
- margin-bottom : définit la marge basse ;
- margin-left : définit la marge haute.

Les valeurs à donner à ces propriétés sont les mêmes que celles de border auxquelles s'ajoute la valeur auto pour laquelle seul le navigateur détermine la valeur de la marge. Nous éviterons ce type de valeur qui laisse l'initiative aux navigateurs et risque ainsi de créer des effets divergents. Il est préférable de définir explicitement une valeur pour créer un effet de présentation donné.

Dans l'exemple 11-6, nous créons des marges pour les différents éléments de la page. Les couleurs de fond qui leur sont attribuées permettent de mieux visualiser ces dernières. Les titres <h1> ont des marges haute et basse de 40 pixels et des marges droite et gauche de 10 % de la largeur de leur élément parent (repère ³). Pour le premier (repère ¿), inclus dans <body> les marges droite et gauche sont donc de 10 % de la largeur de la fenêtre du navigateur. Elles vont donc évoluer si nous redimensionnons cette dernière. Pour l'élément (repère ´), seule la marge gauche est définie à 5 % de la largeur de son parent (ici, il s'agit de <body> (repère ·). En conséquence, le deuxième et le troisième élément <h1> (repères ² et º) qui sont inclus dans ont une marge gauche totale dans la page supérieure à celle du premier car elle représente la somme des 5 % de marge de et des 10 % de marge de <h1> La marge des paragraphes est

donnée par trois valeurs (repère ») ; nous obtenons donc une marge supérieure de 10 pixels, des marges droite et gauche négatives de – 1 em qui font déborder le contenu de l'élément <p> (repère ¶) à droite et à gauche de son parent , et une marge basse de 15 pixels.

Exemple 11-6. Création des marges

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Largeur des bordures</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="bordures individuelles">
   h1{margin: 40px 10%; background-color: yellow;}
   div{margin-left: 5% ; background-color: #CCC;}
   p{margin: 10px -1em 15px; background-color: #EEE;}
  </style>
 </head>
 <body>
    <h1>XHTML</h1>
    <div>
      <h1>XHTML</h1>
      <p> In principio creavit Deus caelum et terram terra autem erat inanis et vacua
        et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas </p>
      <h1>CSS 2.1</h1>
 </div>
 </body>
</html>
```

La figure 11-6 présente le résultat obtenu. Nous pouvons y remarquer que l'espace entre le second titre <h1> et le paragraphe n'est pas la somme des marges de chacun d'eux comme on pouvait s'y attendre mais qu'elle vaut 40 pixels. En effet, les marges basse de <h1> et haute de <p> ont fusionné, et ce, de la manière suivante :

- Les marges hautes et basses de boîtes générées par des éléments de type bloc fusionnent à condition d'être dans un flux normal, c'est-à-dire de ne pas utiliser de propriété de positionnement (comme float ou position que nous aborderons au chapitre 13). Dans ce cas, la marge finale a la plus grande des deux valeurs des éléments contigus. Si une des marges est négative, la marge résultante est la différence des deux marges. Si les deux marges sont négatives, celle qui a la plus grande valeur absolue l'emporte.

- Les marges entre les boîtes flottantes (voir la propriété `float`) ou celles qui sont positionnées absolument ou relativement (voir la propriété `position` au chapitre 13) ne fusionnent jamais.

Figure 11-6
Définition des marges

Les espacements

En nous référant une fois de plus au modèle des boîtes illustré à la figure 11-1, nous pouvons constater qu'il est possible de définir une zone située entre la boîte de contenu d'un élément et sa bordure. Cette zone qui est nommée l'espacement (*padding*) permet comme la propriété `margin` d'aérer la présentation mais cette fois non pas entre deux éléments voisins mais directement autour du contenu, qu'il existe une bordure définie explicitement ou pas. Contrairement à la marge, l'espacement a la même couleur ou image de fond que le contenu de l'élément qu'elle entoure, telles que ces dernières sont définies par les propriétés `background-color`, `background-image` ou `background`. Cet espacement est créé par la propriété `padding` qui s'applique à tous les éléments XHTML, excepté ceux qui sont inclus dans l'élément `<table>` (`<tr>`, `<td>`, `<th>`, `<thead>`, `<tbody>`, `<tfoot>`, `<colgroup>`, `<col />`). La syntaxe de la propriété `padding` est similaire à celle de `margin` :

```
padding: <large>{1,4} | inherit
```

Les valeurs du paramètre `large` sont données par un à quatre nombres positifs qui définissent dans l'ordre les espacements haut, droit, bas et gauche de l'élément. Si nous définissons une seule valeur, elle s'applique ici encore à toutes les marges. Si nous en

définissons deux, la première s'applique aux espacements haut et bas et la deuxième aux espacements droit et gauche. Si nous définissons trois valeurs, la première s'applique en haut, la deuxième à droite et à gauche, et la troisième en bas. Il est souvent plus simple de donner quatre valeurs explicites, dont certaines nulles.

Comme pour les propriétés de bordure ou de marge vues précédemment, il est encore possible de définir individuellement chacun des espacements d'un élément au moyen des propriétés suivantes :

- padding-top : définit l'espacement haut ;
- padding-right : définit l'espacement droit ;
- padding-bottom : définit l'espacement bas ;
- padding-left : définit l'espacement gauche.

Les valeurs possibles et les restrictions sont les mêmes que pour padding.

Dans l'exemple 11-7, nous créons des espacements différents pour quatre paragraphes et des couleurs de fonds qui n'ont ici d'autre rôle que de bien matérialiser les limites de la boîte de chaque élément.

L'espacement du bloc parent de tous les paragraphes est de 20 pixels sur chacun de ses côtés (repère ³). L'espacement de chacun des trois premiers paragraphes (repères ¸ , ¹ et) est défini dans une classe particulière, en pixels, en em et en pourcentage (repères · , » et ¿). L'espacement du dernier paragraphe (repère) a la valeur inherit et va donc hériter de la valeur définie pour son élément parent (repère ´), ici donc de la valeur 20 pixels (l'héritage n'est pas réalisé dans Explorer, ce qui montre une fois de plus l'intérêt de la définition explicite de la valeur souhaitée, sans compter sur l'héritage). Les éléments <h1> inclus directement dans le corps de la page (repère º) n'ont qu'un espacement haut de 0,5 em (repère ²), alors que ceux qui sont inclus dans (repère µ) ont en plus un espacement gauche de 200 pixels (repère ¶).

Exemple 11-7. L'espacement du contenu des éléments

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les marges</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="bordures individuelles">
   div{padding: 20px; background-color: #CCC;}
   p.retrait1{padding: 20px; background-color: #EEE;}
   p.retrait2{padding:2em; background-color: #EEE;}
   p.retrait3{padding-left: 10%; background-color: #EEE;}
   p.herit{padding: inherit; background-color: #EEE;}
   h1{ padding-top: 0.5em; background-color :yellow; border-style: dotted;}
```

```
  div h1{ padding-left: 200px;}    ¶
 </style>
</head>
<body>
 º  <h1>XHTML</h1>
 ¾ <div>
  µ  <h1>XHTML</h1>
  ¸  <p class="retrait1">In principio creavit Deus caelum et terram terra autem erat
  ➥ inanis . . .</p>
  ¹  <p class="retrait2">In principio creavit Deus caelum et terram terra autem erat
  ➥ inanis . . .</p>
     <p class="retrait3">In principio creavit Deus caelum et terram terra autem erat
  ➥ inanis . . .</p>
     <p class="retrait herit">In principio creavit Deus caelum et terram terra autem
  ➥ erat inanis . . .</p>
 </div>
</body>
</html>
```

La figure 11-7 illustre le résultat obtenu pour ces différents espacements.

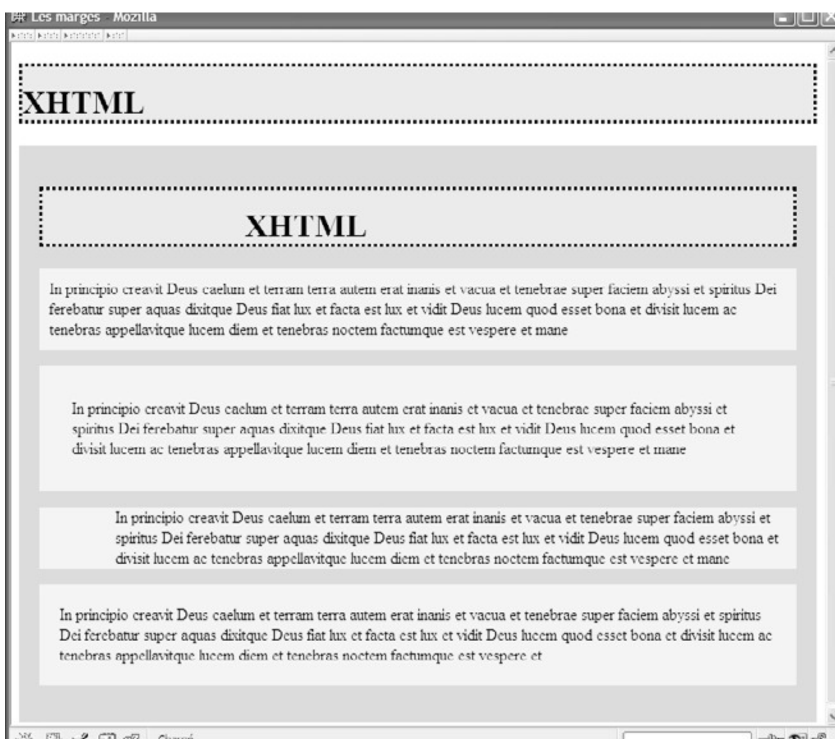


Figure 11-7

Création d'espacements autour des contenus

Les contours

Nous avons vu que les marges situées à l'extérieur des bordures sont transparentes. Dans cet espace extérieur aux bordures, il est possible de créer des contours qui ont une épaisseur, un style et une couleur. Ces contours vont avoir le même aspect qu'une bordure, mais avec deux différences de comportement :

- Les dimensions d'un contour ne sont pas prises en compte dans le modèle de boîte et ne peuvent donc pas modifier les dimensions totales de la boîte la plus externe. Les éléments contigus à la boîte générale pour un élément ne se trouvent pas décalés par l'existence d'un contour comme ils le sont par une marge. L'inconvénient de cet état de fait est qu'un contour très large peut se superposer aux éléments voisins, et en cacher une partie.
- Les contours ne forment pas une boîte rectangulaire et ils peuvent s'étendre sur deux lignes (voir la figure 11-8 pour l'élément `<span>`)

Les propriétés que nous allons utiliser pour créer des contours sont théoriquement applicables à tous les éléments XHTML. Cependant, à ce jour, seul le navigateur Opéra réalise ces contours à peu près correctement. Les autres navigateurs (Mozilla, Firefox, Netscape) ne les affichent que dans certaines conditions voire pas de tout (Explorer 6). En pratique, les propriétés peuvent être appliquées pour faire apparaître dynamiquement un contour autour de certains éléments, comme des images des boutons ou des zones de saisie de formulaire, en réponse aux attributs `onfocus` ou `onmouseover`, gérés par les pseudo-classes `:focus` et `:hover`, ou des scripts JavaScript.

Le style du contour

Le style du contour est la première des propriétés à définir pour obtenir un affichage, les autres ayants des valeurs par défaut. Il est créé grâce à la propriété `outline-style`, dont la syntaxe est similaire à celle de `border-style` :

```
outline-style: none | <style> | inherit
```

La valeur `none` supprime tout contour et le paramètre `<style>` peut prendre une des valeurs suivantes :

- `dotted` : bordure en pointillés courts (figure 11-2, repère ³) ;
- `dashed` bordure en tirets longs (figure 11-2, repère ·) ;
- `solid` : bordure pleine continue (figure 11-2, repère ») ;
- `double`: bordure constituée de deux traits parallèles continus (figure 11-2, repère ¿). Si la largeur de bordure est insuffisante, un seul trait apparaît (pour définir la largeur, voir la propriété `border-width`) ;
- `groove`: bordure en creux. Les quatre côtés sont de couleur différente ce qui permet de créer cet effet de creux (figure 11-2, repère ´) ;

- ridge : bordure en relief. Même remarque sur la couleur des côtés (figure 11-2, repère ²) ;
- inset : bordure en creux dont chaque côté n'a qu'une seule couleur (figure 11-2, repère ¶) ;
- outset : bordure en relief dont chaque côté n'a qu'une seule couleur (figure 11-2, repère º).

Notons que, contrairement aux bordures, il n'est pas possible de définir une valeur de style différente pour chaque côté du contour.

La couleur du contour

La définition explicite de la couleur du contour est obtenue grâce à la propriété outline-color, dont la syntaxe se rapproche de celle de border-color et est la suivante :

```
outline-color: <couleur> | invert | inherit
```

Le paramètre <couleur> est donné par un code de couleur, un mot-clé ou la fonction rgb() comme habituellement. Le mot-clé invert permet d'obtenir un contour dont la couleur est l'inversion vidéo de la couleur du fond sur lequel il est dessiné. Nous sommes ainsi assuré d'obtenir un effet visuel qui attire l'attention. Comme pour le style, la couleur du contour s'applique à ses quatre côtés en même temps. Si nous définissons cette propriété, il faut également obligatoirement définir outline-style, sinon aucun contour ne s'affichera.

Pour créer un contour sur une zone de saisie quand elle reçoit le focus avec le curseur, la touche de tabulation ou un raccourci clavier, nous écrirons par exemple le code suivant :

```
input: focus{outline-style: dotted; outline-color: orange;}
```

La largeur du contour

Nous pouvons enfin définir une largeur pour le contour au moyen de la propriété outline-width, dont la syntaxe est la suivante :

```
outline-width: <long> | thin | medium | thick | inherit
```

Les paramètres sont les mêmes que ceux de la propriété border-width. Les mots-clés thin, medium et thick correspondent respectivement à un contour fin, moyen et épais, leur dimension réelle étant fonction du navigateur. Pour obtenir une dimension fixe, nous préférons définir une longueur fixe dans les unités habituelles (px, em, ex, mm, cm, in, pc, pt).

Comme nous l'avons déjà signalé, les dimensions du contour ne sont pas prises en compte dans le calcul de la dimension totale de la boîte générée pour un élément. Si le contour est plus large que la marge, il y a débordement sur les éléments voisins. Il faut donc veiller à ne pas définir une largeur de contour inesthétique.

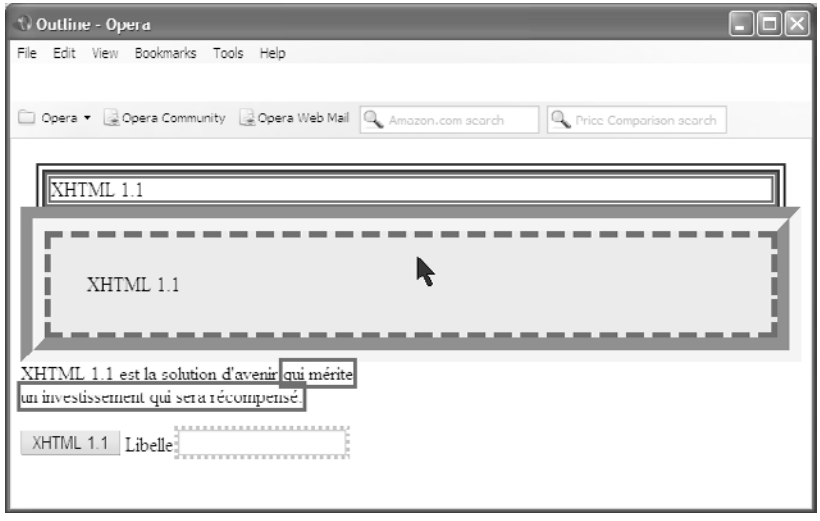
L'exemple 11-8 permet de mettre ces propriétés en œuvre. Les éléments y ont une bordure et une marge, mais également un contour bleu de 7 pixels avec le style double (repère ³). Cela s'applique au premier élément (repère ²). L'association du sélecteur d'attribut id et de la pseudo-classe :hover permet de modifier dynamiquement à la fois le contour mais aussi l'espacement et la bordure de l'élément (repère ·) du second élément (repère ¶). Le survol du bouton inclus dans le paragraphe permet de faire apparaître un contour autour de celui-ci (repère »). Le contour défini dans la classe contour (repère ¿) et appliqué à un élément (repère µ) permet de constater que ce dernier peut apparaître sur plusieurs lignes comme le montre la figure 11-8. Enfin, la pseudo-classe :focus associée à l'élément <input /> (repère ´) permet d'afficher un contour sur une zone de saisie de texte (repère ¸) quand elle reçoit le focus.

Exemple 11-8. Création de contours

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
 <title>Les contours</title>
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="contours">
  div {border: 5px double red; margin: 20px; outline-width: 7px; outline-style:
    ➥double; outline-color: blue;}  ³
  div#out:hover{outline: 20px groove #DDD; padding: 30px; background-color: yellow;
    ➥border-style: dashed;} ·
  button#tour:hover{border-style: dotted; border-width: 4px; outline-style: double;
    ➥outline-color: red; outline-width: 5px}    »
  .contour{outline: 0.2em red solid}   ¿
  input:focus{outline-style: dotted; outline-color: orange;}                    ´
  </style>
 </head>
 <body>
   ²  <div>XHTML 1.1</div>
   ¶  <div id="out"> XHTML 1.1 </div>
   º  <p>XHTML 1.1 est la solution d'avenir
    ¾ <span class="contour">qui mérite <br />un investissement qui sera récompensé.
    ➥ </span>
    µ  <button id="tour">XHTML 1.1</button>
    ➥ Libellé   ¸  <input type="text" name="saisie" size="20" tabindex="1"
    ➥ accesskey="A"/>
  </p>
 </body>
</html>
```

La figure 11-8 montre le résultat obtenu quand la zone de saisie de texte a le focus et quand la seconde division est survolée par le curseur. Nous pouvons remarquer que son contour déborde en haut et en bas sur les éléments adjacents.

Figure 11-8
Les contours obtenus dans Opera 8



Exercices

Exercice 1 : Reproduisez le modèle de boîte de CSS (sans consulter la figure 11-1 bien sûr).

Exercice 2 : Créez une bordure double bleue de 7 pixels pour les divisions, puis une bordure basse rouge de style `inset` et de 1,2 em pour les titres `<h2>` inclus dans ces éléments `<div>`

Exercice 3 : Créez des bordures jaune vif pour les éléments `<input />` de type `text` qui apparaissent uniquement quand ils reçoivent le focus.

Exercice 4 : Créez une bordure inférieure double bleue pour les liens `<a>` ; modifiez dynamiquement la couleur de cette bordure quand ces derniers sont survolés.

Exercice 5 : Créez une bordure d'une couleur différente pour chaque côté de la boîte de l'élément `<p>`

Exercice 6 : Créez une marge gauche de 10 % pour les paragraphes. Réduire dynamiquement cette marge à 2 % quand ils sont survolés par le curseur afin de les mettre en évidence.

Exercice 7 : Créez une bordure simple de 3 pixels et un espacement de 5 pixels pour les éléments `<blockquote>` Augmentez cet espacement à 10 % et rendez la bordure double quand ces éléments sont survolés.

Exercice 8 : Créez un espacement gauche progressif de 1, 2 et 3 em respectivement pour les titres `<h1>` `<h2>` et `<h3>`

Exercice 9 : Créez un contour rouge double de 5 pixels pour les boutons d'envoi des formulaires quand ils reçoivent le focus.

12

Le style du texte et des liens

Ceux qui ont connu l'élément <font>, qui permettait de définir la taille du texte en HTML 3.2, ne risquent pas de regretter sa disparition, tant il leur paraîtrait pauvre par rapport aux multiples possibilités de gestion des polices offertes par les propriétés CSS.

Dans une page web, le texte constitue souvent l'essentiel du contenu. Pour la définition de ses styles, nous allons suivre la même logique que celle mise en œuvre dans un traitement de texte. Nous pouvons par exemple agir sur la police de caractères et leur taille, puis sur leur style physique comme le passage en gras de certaines parties du texte en gras, en italique ou souligné, ou bien encore comme la création de lettrine.

Les anciennes définitions de styles des liens, écrites à l'aide d'attributs de l'élément <body>sont aujourd'hui obsolètes. Elles sont désormais remplacées par l'utilisation de pseudo-classes spécifiques.

Les polices

La plupart des polices de caractères que nous utilisons quotidiennement, telles que Times New Roman ou Arial, sont en réalité des familles de polices car elles se déclinent en plusieurs polices réelles de types différents, par exemple une série de caractères normaux, une série en gras, une série en italique ainsi que les diverses combinaisons possibles. La propriété font-family permet de définir une ou plusieurs familles de polices par leur nom. Si ce nom est composé et contient des espaces, il doit être écrit entre guillemets. En supplément, CSS propose cinq familles de polices génériques, le navigateur pouvant ainsi choisir la plus proche si celle qui est déclinée explicitement n'est pas disponible sur le

poste client. Ces familles génériques sont définies à l'aide des mots-clés suivants qui, pour leur part, ne doivent pas être écrits entre guillemets.

- serif : il s'agit des polices à empattements et proportionnelles, c'est-à-dire pour lesquelles tous les caractères n'ont pas la même largeur (par exemple « w » et « i »). On trouve dans cette famille les polices Times New Roman, Baskerville, Georgia, Modern.
- sans-serif : il s'agit des polices sans empattement et proportionnelles. On trouve dans cette famille les polices Arial, Abadi, Helvetica, Verdana.
- cursive : il s'agit des polices dont l'aspect ressemble à l'écriture manuscrite. Elles sont également proportionnelles. On trouve dans cette famille les polices Script et Vivaldi.
- monospace : il s'agit des polices non proportionnelles dans lesquelles chaque caractère occupe la même largeur, comme dans une machine à écrire ou un éditeur de code comme Edit Plus. On trouve dans cette famille des polices telles que Courier New.
- fantasy : on classe dans cette famille toutes les polices originales ne rentrant pas dans les catégories précédentes. Le rendu final de ce type de police est assez aléatoire selon les navigateurs. On peut par exemple citer la police Comic Sans MS.

Voici la syntaxe de la propriété font-family :

```
font-family :[[<nom> |<generic>],]* | inherit
```

Nous pouvons définir plusieurs noms de familles de polices à la suite en les séparant par une virgule, puis terminer par un nom de famille générique. Dans ce cas, le navigateur tente d'utiliser la première en priorité et, si elle n'est pas disponible, il recherche la deuxième, et ainsi de suite. Si aucune police ne correspond aux noms des familles indiquées, il utilisera la police générique précisée. Il est donc conseillé de toujours indiquer une famille générique en fin de liste pour obtenir un résultat ressemblant à ce qui est attendu.

L'exemple 12-1 permet de mettre en œuvre cette propriété pour différents éléments. La police par défaut de la page est définie comme étant « Times New Roman » avec le sélecteur body (repère ³). Cette police est donc applicable à tous les éléments n'ayant pas de style propre, par exemple <h1> (repère ²) et le troisième paragraphe <p> (repère µ). Le premier paragraphe (repère ¶) utilise une police Arial grâce à un sélecteur d' id (repère ·). Les classes p.cursiv (repère ») et p.fantasy (repère ¿) permettent respectivement d'appliquer une police de type script au deuxième paragraphe (repère ¾) et la police Comic Sans MS au dernier (repère ¸). Pour l'élément <code> (repère º) inclus dans le premier paragraphe, nous définissons une police à espacement fixe, Courier New (repère ´).

Exemple 12-1. Choix de la police

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les polices du texte</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="fontes">
   body{font-family:"Times New Roman",Georgia,serif}
   p#arial{font-family:Arial sans-serif;}
   p.cursiv{font-family: Vivaldi, cursive;color:white;background-color:#333;}
   p.fantasy{font-family: "Comic Sans MS",fantasy;color:white;
    background-color:#333;}
   code{font-family:"Courier New",monospace;}
  </style>
 </head>
 <body>
  <h1>Les polices</h1>
   <p id="arial"> ARIAL : In principio creavit Deus caelum et terram terra autem
   erat inanis et vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur
   super aquas dixitque Deus fiat lux et facta est lux et vidit <br />
   <code>MONOSPACE:Deus fiat firmamentum in medio aquarum et dividat aquas ab aquis
   et fecit Deus firmamentum divisitque aquas quae erant sub firmamento ab his
   quae erant super firmamentum </code>
  <br /> Deus duo magna luminaria luminare maius ut praeesset diei et luminare minus
   ut praeesset nocti et stellas et posuit  </p>
   <p class="cursiv"> CURSIVE: In principio creavit Deus caelum et terram terra
   autem erat inanis et vacua et tenebrae super faciem abyssi et spiritus Dei
   ferebatur super aquas dixitque Deus fiat lux et facta est lux et vidit Deus
   lucem quod esset bona et divisit lucem ac tenebras appellavitque lucem diem et
   tenebras noctem factumque est vespere et mane </p>
   <p> In principio creavit Deus caelum et terram terra autem erat inanis et vacua
   et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas </p>
   <p class="fantasy"> In principio creavit Deus caelum et terram terra autem erat
   inanis et vacua et tenebrae super faciem abyssi et spiritus Dei </p>
 </body>
</html>
```

La figure 12-1 présente le résultat obtenu. Nous pouvons remarquer qu'à taille égale, la police script Vivaldi est beaucoup moins lisible que les autres.

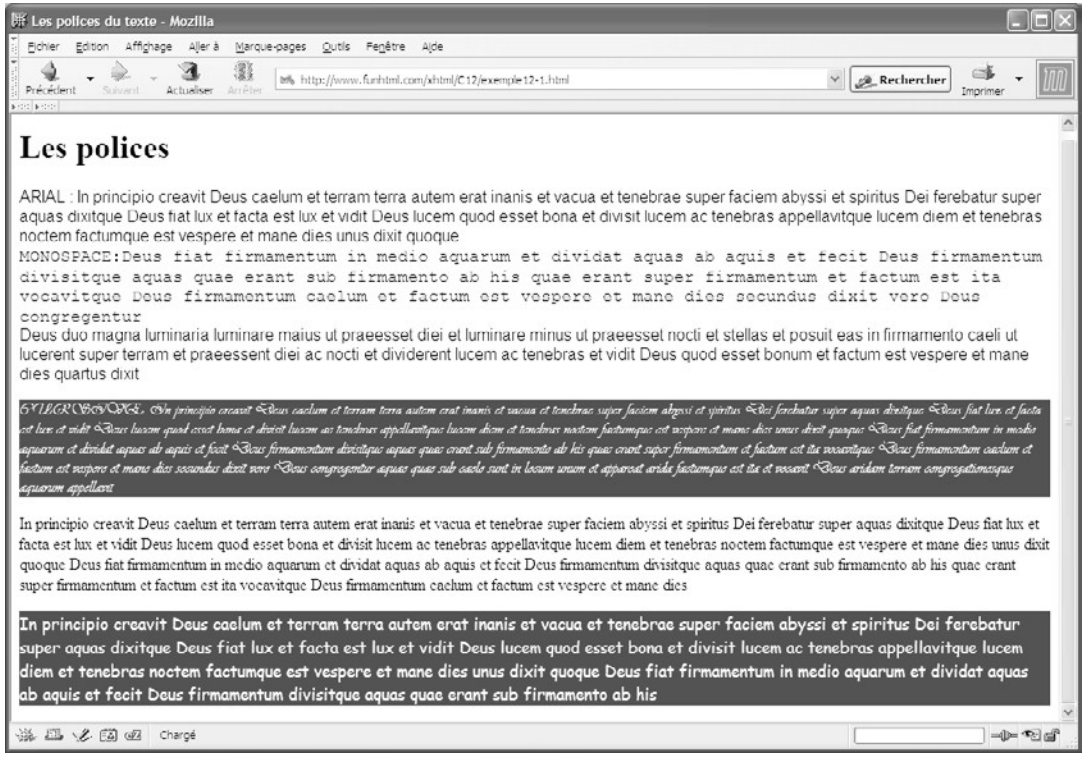


Figure 12-1
Les polices du texte

La taille des polices

Pour mettre en évidence les différentes parties de texte composant une page, il convient de bien déterminer la taille des polices qui y seront utilisées. Celle-ci est définie grâce à la propriété `font-size`, dont la syntaxe est la suivante :

- font-size: <taille-absolue> | <taille-relative> | <long> | <pourcent> | inherit

Elle s'applique à tous les éléments XHTML contenant du texte. La taille peut être définie de manière absolue, relative, à l'aide d'unités de longueur ou en pourcentage. Nous allons examiner ces différentes possibilités et les illustrer en détail.

Les tailles absolues

Les valeurs de taille absolue sont définies par les mots-clés suivants, de la taille la plus petite à la plus grande :

- xx-small, x-small, small, medium, large, x-large, xx-large.

Le mot-clé medium correspond à la taille normale choisie par l'utilisateur dans les préférences de son navigateur ou définie par défaut par ce dernier. Un coefficient multiplicateur de 1,2 est appliqué entre chaque taille et la suivante (il était de 1,5 en CSS 1 et il garde cette valeur dans les anciens navigateurs).

Si, par exemple, la taille correspondant au mot-clé medium est de 15 pixels, la taille calculée définie par small sera de 15 / 1,2 = 12,5 (arrondi à 12), et celle définie par large de 15 ¥ 1,2 = 18 pixels, et ainsi de suite en montant ou en descendant dans l'ordre des mots-clés. L'exemple 12-2 permet de tester ces différentes tailles de caractères. La taille des caractères de la page est définie par défaut avec le mot-clé large pour le sélecteur body (repère ³) ; cette valeur est donc héritée par le paragraphe <p> (repère). Nous définissons ensuite les tailles des caractères des éléments <h1> à <h6> (repères ¸ à) avec les mots-clés de valeurs décroissantes, de xx-large à x-small (repères · à ¶). La taille du contenu de l'élément <address> est fixée à la valeur xx-small. Le pseudo-élément :first-letter est utilisé pour créer une lettrine de taille x-large (repère ¾) pour le paragraphe situé en fin de page (repère) et doté d'une couleur de fond (repère µ). La couleur de fond de la lettrine met en évidence son indépendance par rapport au texte du paragraphe.

Exemple 12-2. La taille des caractères

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Taille des polices</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="fontes">
   body{font-size: large;}
   h1{font-size: xx-large;}
   h2{font-size: x-large;}
   h3{font-size: large;}
   h4{font-size: medium;}
   h5{font-size: small;}
   h6{font-size: x-small;}
   address{font-size: xx-small;}
   p:first-letter{font-size: xx-large; color: #000; background-color: #AAA;}
   p{background-color: #EEE;}
  </style>
 </head>
 <body>
  <h1>Les tailles des caractères : xx-large</h1>
  <h2>Les tailles des caractères : x-large</h2>
  <h3>Les tailles des caractères : large</h3>
  <h4>Les tailles des caractères : medium</h4>
```

```
<h5>Les tailles des caractères : small</h5>
<h6>Les tailles des caractères : x-small</h6>
<address>Les tailles des caractères <i>xx-small</i></address>
<p> La genèse : In principio creavit Deus caelum et terram terra autem erat inanis
➥ et vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
➥ dixitque Deus fiat lux et facta est lux et vidit . . . </p>
</body>
</html>
```

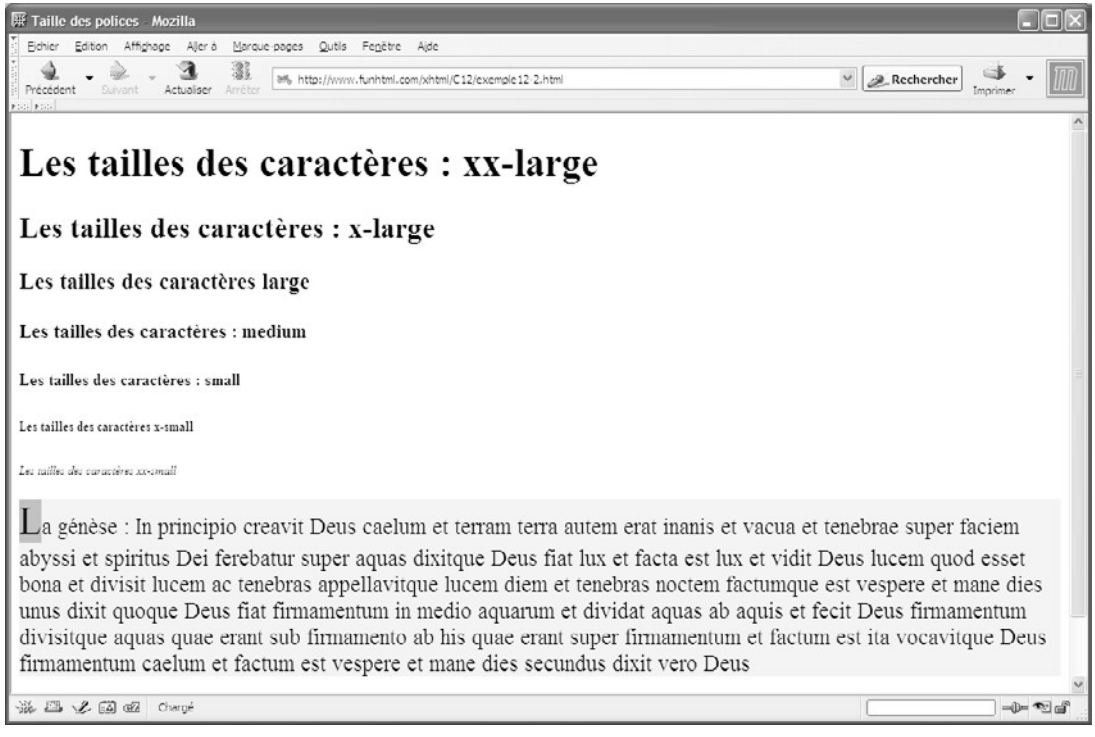


Figure 12-2
Les tailles absolues des polices

Les tailles relatives

Les tailles relatives sont définies à l'aide des mots-clés `smaller` (plus petit) ou `larger` (plus grand). Les tailles réelles obtenues sont calculées d'après la taille de la police de l'élément parent si elle a été fixée, ou par rapport à la taille par défaut utilisée par le navigateur. Le coefficient multiplicateur entre la taille normale et les tailles obtenues est aussi de 1,2 en CSS 2.

Dans l'exemple 12-3, la taille par défaut est définie pour <body> à la valeur medium (repère ³) Le texte de l'élément <h1> (repère ¶) est plus grand car sa taille est définie grâce au mot-clé larger (repère ·). Le texte de la première division (repère º) hérite de la taille medium. Le texte de la seconde division (repère ¾) a une taille explicite définie à la valeur x-large par la classe div.xlarge (repère »). Le texte de l'élément <p> (repère µ) inclus dans cet élément est plus petit que celui de son parent car sa taille est définie par la valeur smaller (repère ¿), et celui de (repère ¸) est encore plus petit, car il est aussi défini à la valeur smaller, l'élément <code> étant enfant de l'élément <p> L'effet de la valeur smaller est donc bien cumulatif à chaque inclusion d'élément. Le texte qui suit (repère ¹) étant directement inclus dans la division, il a la taille x-large. Le dernier paragraphe (repère) hérite explicitement de la taille définie pour la page (repère ²). En supposant que l'utilisateur ait classé par défaut dans son navigateur une taille standard de 16 pixels, correspondant à la valeur medium, nous obtiendrions les tailles de polices suivantes :

- pour <h1> : 16 ¥ 1,2 = 19,2 pixels arrondis à 19 pixels ;
- pour le premier élément : 16 pixels par héritage ;
- pour le contenu direct du second élément : 16 ¥ 1,2 ¥ 1,2 = 23 pixels ;
- pour le paragraphe <p> inclus dans : 23 / 1,2 = 19 pixels ;
- pour l'élément inclus dans <p> : 19 / 1,2 = 16 pixels.

Toutes ces tailles de caractères sont donc liées à la taille par défaut définie dans le navigateur.

Exemple 12.3. Les tailles relatives des polices

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Tailles relatives des polices </title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="fontes">
   body{font-size: medium;} ³
   h1{font-size:larger;}    ·
   div.xlarge{font-size:x-large; background-color: #BBB;}   »
   p.small{font-size: smaller; background-color: #CCC;}  ¿
   .petit{font-size: smaller; background-color: #EEE;}
   .herite{font-size: inherit;}      ²
```

```
 </style>
</head>
<body>
 ¶ <h1>Les tailles relatives des caractères</h1>
 º <div> MEDIUM (héritée de body) : In principio creavit Deus caelum et terram
 ➥ terra autem erat inanis et vacua et tenebrae super faciem abyssi et spiritus
 ➥ Dei . . .</div>
 ¾ <div class="xlarge">x-large : In principio creavit Deus caelum et terram autem
 ➥ erat inanis et vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur
 ➥ super aquas
  µ <p class="small"> SMALLER : In principio creavit Deus
    ¸ <span class="petit">SMALLER : caelum et terram terra autem erat inanis
 </span> et vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur
    ➥ super aquas
 </p>
 ¹ In principio creavit Deus caelum et terram
</div>
    <p class="herite"> Cette taille de la police est héritée de body</p>
</body>
</html>
```

La figure 12-3 présente le résultat obtenu pour ces définitions de tailles relatives.

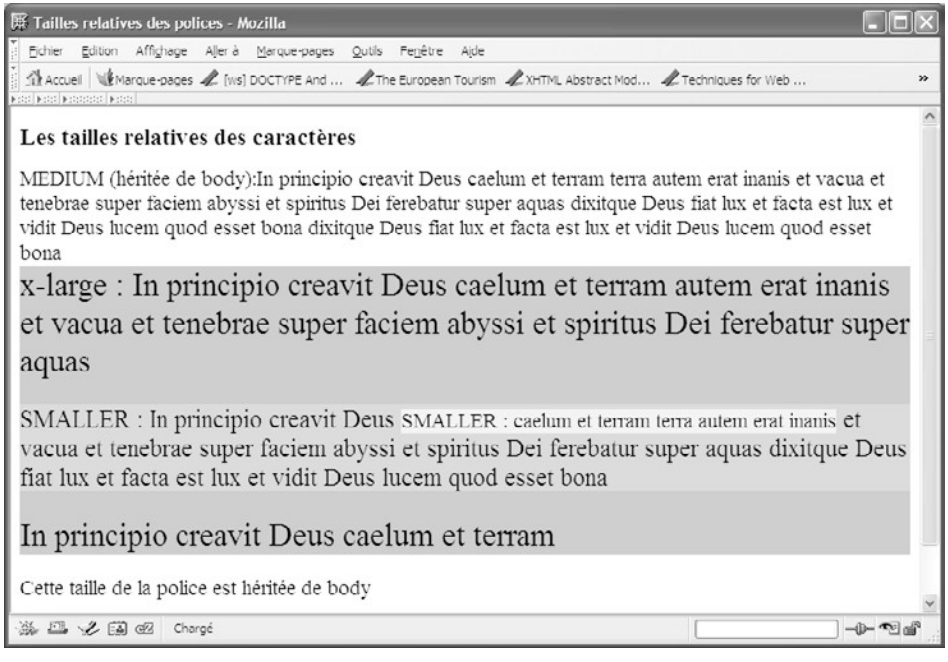


Figure 12-3

Les tailles relatives des polices

Les tailles dimensionnées

Les tailles dimensionnées sont définies à l'aide d'un nombre positif et d'une unité de longueur relative ou absolue (mm, cm, in, px, em, ex, pt, pc). Comme nous l'avons déjà indiqué, le choix des unités doit être fonction du média affichant la page. Les unités comme px, em et ex, se prêtent mieux aux écrans, et pt et pc aux médias imprimés.

L'exemple 12-4 définit des styles de polices à l'aide de tailles dimensionnées. La taille par défaut de la page est fixée à 18 pixels (repère ³). Comme dans les exemples précédents, elle va donc s'appliquer aux éléments n'ayant pas de style propre, à l'instar de la première division (repère º), ou en ayant un explicitement hérité (repère ²), comme le dernier paragraphe (repère ¹). Les éléments <h1> (repère ¶) ont une taille de 35 pixels (repère ·). La classe div.plusem (repère ») permet de définir une taille de 1,5 em, soit 50 % de plus que la hauteur de la boîte correspondant à la police de 18 pixels. Elle s'applique à la première division (repère ¾). La classe p.plusex (repère ¿) permet de définir pour le paragraphe (repère µ) une taille égale à 150 % de celle du caractère « x » de la police en cours. La classe mm7 définit une hauteur explicite de 7 mm (repère ´) qui s'applique à l'élément (repère ¸).

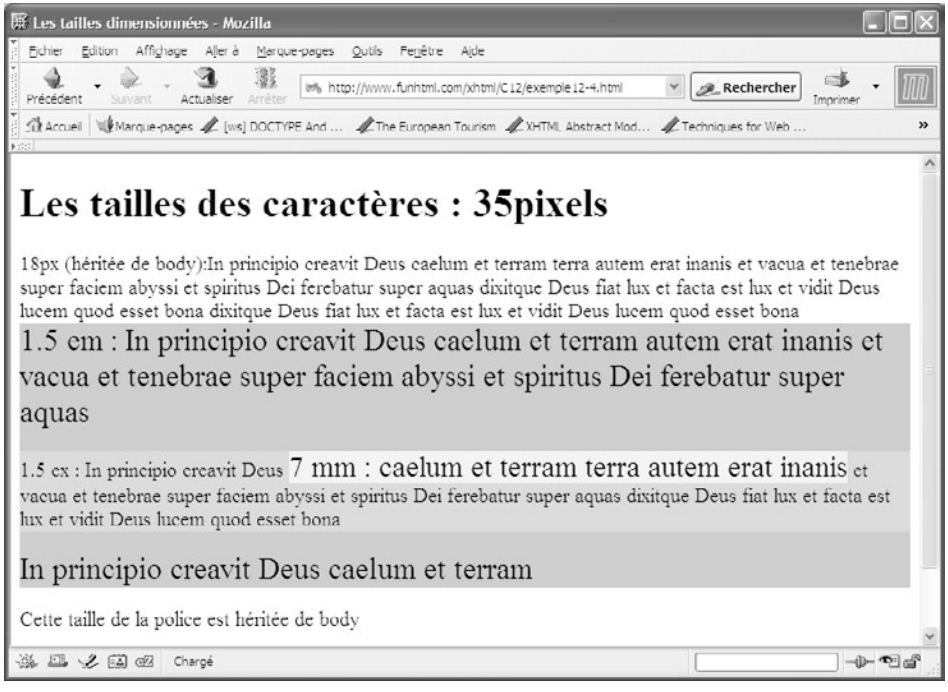
Exemple 12-4. Les tailles dimensionnées

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Dimensionnement des éléments</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="fontes">
   body{font-size: 18px;}  ³
   h1{font-size:35px;}  ·
   div.plusem{font-size:1.5em;background-color:#BBB;}»
   p.plusex{font-size:1.5ex ;background-color:#CCC;}¿
   .mm7{font-size: 7mm;background-color:#EEÉ;}
   .herite{font-size: inherit;}  ²
  </style>
 </head>
 <body>
   ¶  <h1>Les tailles des caractères : 35 pixels</h1>
   º  <div> 18px (héritée de body) : In principio creavit Deus caelum et terram terra
   ➥ autem erat inanis et vacua et tenebrae super faciem abyssi et spiritus Dei
   ➥ . . . </div>
   ¾ <div class="plusem">1.5em : In principio creavit Deus caelum et terram autem
   ➥ erat inanis et vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur
   ➥ super aquas
```

```
µ  <p class="plusex"> 1.5ex : In principio creavit Deus
   ¸  <span class="mm7">7mm : caelum et terram terra autem erat inanis</span>
  et vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
   </p>
 In principio creavit Deus caelum et terram</div>
   ¹  <p class="herite"> Cette taille de la police est héritée de body</p>
 </body>
</html>
```

Le résultat est présenté à la figure 12-4.

Figure 12-4
Les tailles dimensionnées des polices



Les tailles en pourcentage

Pour définir les tailles des polices dans la propriété `font-size`, on peut aussi utiliser des pourcentages sous la forme d'un entier suivi du caractère %. Les dimensions des polices sont alors calculées par rapport à celles de l'élément parent. Dans l'exemple 12-5, nous utilisons le même contenu que dans l'exemple précédent et nous définissons la taille de base pour l'élément `<body>` à 18 pixels (repère ³). Celle des titres `<h1>` (repère ¶) est fixée à 250 %, soit une fois calculée à 250 % de 18 = 45 pixels (repère ·). La taille des caractères de la première division `<div>` (repère °) est héritée implicitement de celle de `<body>`. En revanche, dans la division suivante (repère ¾), les caractères ont une taille

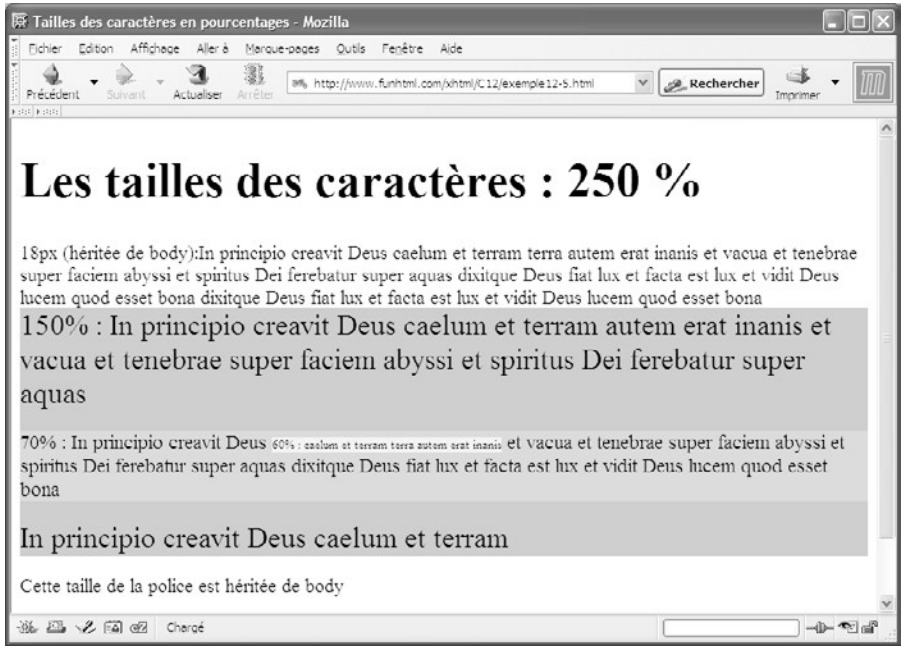
de 150 %, soit 24 pixels (repère »). Cette division contient un paragraphe avec une taille de police fixée à 70 % (repère ¿), donc 70 % de 24 pixels = 17 pixels, puisque la taille dans son parent est de 24 pixels. L'élément <p> (repère µ) contient à son tour un élément (repère ¸) dont le contenu a une taille de 60 % de celle de son parent, soit 60 % de 17 pixels = 10 pixels (repère ´). Enfin, le dernier paragraphe (repère ¹) hérite explicitement de la taille de son parent, soit <body> (repère ²).

Exemple 12-5. Les tailles des polices en pourcentage

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Tailles des caractères en pourcentage</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="tailles">
   body{font-size: 18px;}
   h1{font-size:250%;}
   div.plus150{font-size: 150%; background-color: #BBB;}
   p.moins70{font-size: 70%; background-color: #CCC;}
   .moins60{font-size: 60%; background-color: #EEE;}
   .herite{font-size: inherit;}
  </style>
 </head>
 <body>
    <h1>Les tailles des caractères : 250%</h1>
    <div> 18px (héritée de body) : In principio creavit Deus caelum et terram
    terra autem erat inanis et vacua et tenebrae super faciem abyssi et spiritus
    Dei ferebatur super aquas dixitque Deus fiat lux et facta est lux et vidit
    Deus lucem quod esset bona dixitque Deus fiat lux et facta est lux et
    vidit Deus lucem quod esset bona</div>
    <div class="plus150">150% : In principio creavit Deus caelum et terram autem
    erat inanis et vacua et tenebrae super faciem abyssi et spiritus . . .
      <p class="moins70">70% : In principio creavit Deus
        <span class="moins60">60% : caelum et terram terra autem erat inanis</span>
   et vacua et tenebrae super faciem </p>
  In principio creavit Deus caelum et terram</div>
    <p class="herite"> Cette taille de la police est héritée de body</p>
 </body>
</html>
```

En appliquant ces styles au même code XHTML, nous obtenons l'affichage présenté à la figure 12-5.

Figure 12-5
Les tailles en pourcentage



La graisse du texte

On nomme graisse d'une police de caractère le fait qu'elle soit en caractère gras ou maigre. Cette caractéristique est définie par la propriété `font-weight`, dont les valeurs sont des mots-clés ou des nombres. La syntaxe de la propriété `font-weight` est la suivante :

```
font-weight : normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 |
➥ 700 | 800 | 900 | inherit
```

Les valeurs relatives `bolder` et `lighter` indiquent respectivement que la police doit être plus grasse ou plus maigre que celle de l'élément parent. Notons que si la police de l'élément parent est déjà la plus grasse ou la plus maigre, l'utilisation de `bolder` ou `lighter` sera sans effet visible.

Les valeurs `normal` et `bold` définissent respectivement l'utilisation de la graisse allant de la plus mince (valeur `100`) à la plus forte (valeur `900`). Nous précisons bien que cela est théorique car, pour une police donnée définie avec la propriété `font-family`, nous ne disposons pas de neuf graduations de graisse, mais en général de trois au maximum. L'équivalent du mot-clé `normal` est la valeur `400` et celui du mot-clé `bold` est la valeur `700`.

La valeur `inherit` permet comme d'habitude de choisir explicitement la même graisse que celle de l'élément parent.

Dans l'exemple 12-6, nous définissons la graisse de `<body>` à la valeur `normal` (repère ³) qui s'applique aux éléments pour lesquels aucune graisse n'est définie explicitement. On pourrait donc s'attendre à ce que les éléments `<div>` (repère º) et `<h2>` (repère ¶) aient une graisse normale. Si c'est bien le cas pour `<div>`, il n'en est rien pour `<h2>` qui apparaît en gras, comme on peut le constater à la figure 12-6. En effet, il conserve son style par défaut qui est inclus dans le navigateur et qui apparaît en gras. Il faut donc se méfier de la feuille de style par défaut et ne pas oublier ses caractéristiques. La graisse des éléments `<h1>` et `<em>` est fixée à la valeur `bolder` (repère ·), ils seront donc plus gras que leurs parents. Le sélecteur `div.gras` (repère ») permet d'appliquer la valeur `bold` à une division (repère ¾). Elle contient un élément `<span>` (repère µ) pour lequel la graisse est fixée à la valeur `300` (repère ´). Les paragraphes ont quant à eux une graisse systématiquement moins forte que leur parent (repère ¿).

Exemple 12-6. La graisse du texte

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Graisse du texte</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="graisse">
   body{font-weight:normal;} ³
   h1,em{font-weight: bolder; background-color: #CDD;} ·
   div.gras{font-weight:bold; background-color: #EEE;} »
   p{font-weight: lighter;}  ¿
   span{font-weight: 300;background-color: #CDD;} ´
  </style>
 </head>
 <body>
  <h1>Graisse du texte <span>XHTML et CSS 2</span></h1>
  <h2>XHTML</h2>
   º <div> In principio creavit Deus <em>caelum et terram</em> terra autem erat
   ➥ inanis et vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur super
   ➥ aquas . . .</div>

   ¾ <div class="gras"> In principio creavit Deus
    µ <span>caelum et terram</span> terra autem erat inanis et vacua et tenebrae
    ➥ super faciem abyssi et spiritus Dei ferebatur super aquas dixitque. . .
    ¸ <p>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
    ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super . . .</p>
   Deus fiat firmamentum in medio aquarum et dividat aquas . . .
  </div>
 </body>
</html>
```

Ces styles créent l'affichage présenté à la figure 12-6.

Figure 12-6
La graisse du texte

Le style des polices

Comme dans un traitement de texte, nous pouvons modifier le style des polices de caractères à l'aide de la propriété `font-style`, qui permet principalement de mettre un texte en italique. La syntaxe de cette propriété est la suivante :

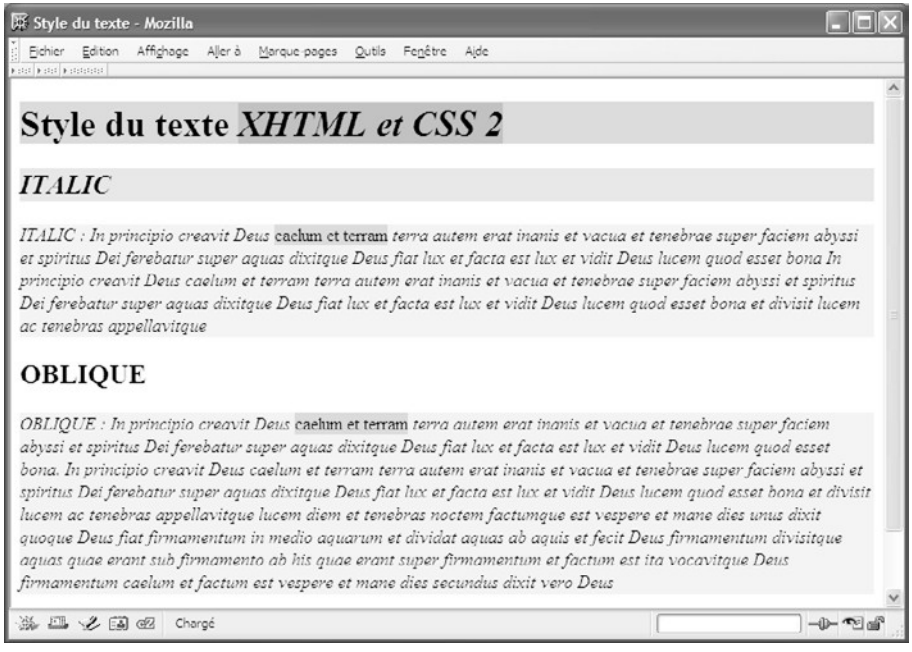
```
font-style:normal | italic |oblique |inherit
```

Dans la pratique, les valeurs `italic` et `oblique` donnent le même résultat pour une police donnée, y compris dans les navigateurs les plus récents. À la valeur `normal` correspond le style de caractère droit, qui est la valeur par défaut.

Dans l'exemple 12-7, le style de la page est défini à la valeur `normal` (repère ³) et les éléments `<h1>` `<em>` et `<span>` également, avec en plus une couleur de fond pour les mettre en évidence (repère ·). Le sélecteur `h1 span` définit le style `italic` pour les éléments `<span>` enfants de `<h1>` (repère »). Cela s'applique à la deuxième partie du titre principal (repère ¶). Le sélecteur `h1+h2` permet d'appliquer le style `italic` uniquement à l'élément `<h2>` qui suit immédiatement `<h1>` (repère ¾) et non pas au second (repère ¹). Nous définissons

ensuite deux classes pour l'élément <p> avec les valeurs respectives italic et oblique (repères ´ et ²). Nous pouvons constater sur la figure 12-7 que leurs contenus ont la même apparence. Les éléments et (repères ¸ et ¹), inclus dans les paragraphes (repères µ et), ayant un style explicite (repère ·) n'héritent pas de leur parent respectif.

Figure 12-7
Le style du texte



Exemple 12-7. Le style des polices

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Style du texte</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="graisse">
   body{font-style:  normal;}
   h1,em,span{font-style: normal; background-color: #CCC;}
   h1 span {font-style: italic; background-color: #AAA;}
   h1 + h2 {font-style: italic; background-color: #DDD;}
   p.italic{font-style: italic; background-color: #EEE;}
   p.oblic{font-style: oblique; background-color: #EEE;}
  </style>
 </head>
```

```
<body>
 ¶ <h1>Style du texte ° <span>XHTML et CSS 2</span></h1>
 ¾ <h2>ITALIC</h2>
 µ <p class="italic">ITALIC : In principio creavit Deus ¸ <em>caelum et terram
 ➥ </em> terra autem erat inanis et vacua et tenebrae super faciem abyssi et
 ➥ spiritus Dei ferebatur super aquas dixitque Deus fiat lux et facta est lux
 ➥ et vidit . . .</p>
 ¹ <h2>OBLIQUE</h2>
   <p class="oblic">OBLIQUE : In principio creavit Deus <span>caelum et terram
 ➥ </span> terra autem erat inanis et vacua et tenebrae super faciem abyssi
 ➥ . . .</p>
</body>
</html>
```

Nous pouvons également enrichir la présentation du texte à l'aide de propriétés annexes. La propriété `font-variant`, dont la syntaxe est :

```
font-variant:small-caps|normal|inherit
```

permet de transformer toutes les minuscules d'un texte en petites majuscules quand on lui attribue la valeur `small-caps` les majuscules conservant leur taille normale. La valeur `normal` permet de conserver pour le texte l'aspect qu'il a dans le code XHTML.

Quand les données textuelles proviennent soit d'une base de données, soit des utilisateurs eux-mêmes, et qu'elles sont utilisées pour créer des pages dynamiques, il peut être utile d'uniformiser la casse du texte. Chaque utilisateur peut en effet avoir saisi son nom par exemple dans une casse différente. Grâce à la propriété `text-transform`, il est possible d'agir sur la casse du texte d'un élément. Sa syntaxe est la suivante :

```
text-transform : uppercase | lowercase | capitalize | none | inherit
```

Elle est héritée par tous les éléments enfants. La valeur `none` implique qu'il n'y a aucune modification de la casse. Les valeurs des mots-clés sont les suivantes :

- `uppercase` : le texte est mis en majuscules ;
- `lowercase` : le texte est mis en minuscules ;
- `capitalize` : seule la lettre initiale de chaque mot est mise en majuscules.

Nous pouvons enrichir la présentation du texte avec certains attributs particuliers, possibles en traitement de texte, comme le soulignement. La propriété `text-decoration` permet la réalisation de ces effets. Voici sa syntaxe :

```
text-decoration :none | [underline || overline || line-through || blink] | inherit
```

La valeur `none` élimine tout effet et peut permettre par exemple de supprimer le soulignement qui apparaît par défaut pour les liens. Les différents mots-clés pouvant être utilisés ont la signification suivante :

- `underline` : le texte est souligné (comme pour les liens).

- overline : le texte est surligné (pas au sens où on l'entend actuellement en utilisant un surligneur, mais au sens propre, un trait horizontal apparaissant au-dessus du texte). Pour surligner au sens courant, il faut utiliser la propriété background-color
- line-through : le texte est barré par un trait horizontal placé à mi-hauteur.
- blink : le texte clignote (il sagit de l'équivalent de l'ancien élément <blink> qui n'était visible que dans Internet Explorer).

L'exemple 12-8 permet de mettre en œuvre ces différentes propriétés pour personnaliser des textes. Les titres <h1> (repère º) sont définis avec une initiale en majuscule à chaque mot et sont soulignés (repère ³). Les titres <h2> (repère µ) conservent leur casse et sont surlignés (repère ·). Les titres <h3> (repère ¸) sont affichés en petites majuscules (repère »). Les éléments sont a priori tous en majuscules (repères ¿ , ¾ et). Cependant, trois classes permettent localement de modifier cet état. La classe min met le contenu en minuscules (repères ´ et). La classe raye permet de « rayer HTML du monde » (repères ² et ¸) et la classe cligno attire l'attention en faisant clignoter « XHTML 1.1 » des feux de la nouveauté (repères ¶ et).

Exemple 12-8. Les modifications du texte

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Transformations et décorations du texte</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="transformations">
   h1{text-transform: capitalize;text-decoration: underline;
    ➥ background-color: #CCC;³}
   h2{text-transform: none;text-decoration: overline;background-color: #EFF;}  ·
   h3{font-variant:small-caps;background-color: #CCF;}»
   span{text-transform: uppercase ;}  ¿                    ´
   .min{text-transform: lowercase;;background-color: black;color:white; }
   .raye{text-decoration: line-through;}                  ²
   .cligno{text-decoration: blink;color: red;font-weight: bold;}        ¶
  </style>
 </head>
 <body>
   º  <h1>style du texte ¾ <span>xhtml et CSS 2</span></h1>
   µ  <h2>XHTML : c'est l'avenir. ¸ <span class="raye">HtMl</span> est mort</h2>
   ¹  <h3>XHTML 1.1 en est la dernière version parue </h3>
      <p>Le langage  <span class="cligno">XHTML 1.1</span> : Les éléments <span
    ➥ class="min">HTml, HeAd et BODY</span> constituent la base de la structure d'une
    ➥ page  <span>xhtml</span>.<br />In principio creavit Deus caelum et terram
    ➥ terra autem erat inanis et vacua et . . . </p>
  </body>
</html>
```

La figure 12-8 permet de visualiser les effets obtenus.

Figure 12-8
Les styles de modifications du texte

Régler l'interligne

Afin d'améliorer la lisibilité du texte d'un élément, tel qu'un paragraphe, en « aérant » son contenu, nous pouvons modifier l'interligne par défaut à l'aide de la propriété `line-height` dont la syntaxe est :

```
line-height:normal | <nombre> | <long> | <pourcent> | inherit
```

La valeur de cette propriété, toujours héritée par les éléments enfants, est généralement supérieure à celle de la propriété `font-size`. Dans ce cas, la différence entre ces deux valeurs est la somme des deux demi-interlignes, situés l'un au-dessus de la ligne de texte et l'autre en dessous. La valeur `normal` définit explicitement l'interligne par défaut correspondant à la police employée. Le paramètre `<nombre>` définit un coefficient multiplicateur qui est appliqué à la valeur de la taille de police définie par la propriété `font-size`. Les valeurs de `<nombre>` peuvent être décimales mais pas négatives. Ce coefficient est hérité par les éléments enfants. La valeur `<long>` est donnée par un nombre positif et une unité de longueur habituelle relative ou absolue (`em`, `ex`, `px`, `mm`, `cm`, `in`, `pc`, `pt`). Il est enfin possible de définir une valeur en pourcentage qui sera appliquée à la taille de la police spécifiée par la propriété `font-size`.

Dans l'exemple 12-9, nous définissons tout d'abord une taille de caractères de 20 pixels (repère ³). L'interligne pour l'élément `<h1>` est fixé à la valeur `2 em` (repère ·) et celui de `<h2>` à `0,6 em` (repère »). La figure 12-9 permet de constater que dans ce dernier cas les caractères montants et descendant tels que « p » ou « f » sont plus hauts que l'interligne et sortent de la boîte de contenu de l'élément matérialisée par une couleur de fond. Les classes `norm` (repère ¿) et `serre` (repère ´) permettent de comparer des paragraphes avec l'interligne standard (repère ¾) et très resserré (repère µ). Avec l'élément `<span>` (repère ¶), qui utilise la classe `norm`, on peut matérialiser au moyen de sa couleur de fond les différences d'interligne, quand il est inclus dans l'élément `<h1>`

Exemple 12-9. Définition de l'interligne

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Réglage de l'interligne</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="interligne">
   body{font-size: 20px;}
   h1{line-height: 2em; background-color: #CCC;}
   h2{line-height: 0.5em; background-color: #AAA;}
   .norm{line-height: 1 em; background-color: yellow;}
   .serre{line-height: 0.7em; background-color: #EEE;}
  </style>
 </head>
 <body>
    <h1>Style du texte XHTML et CSS 2</h1>
    <h2>XHTML : pas de quoi s'en faire un monde</h2>
    <p class="norm">In principio creavit Deus caelum et terram terra autem erat
    inanis et vacua et tenebrae super faciem abyssi et spiritus Dei . . .</p>

    <p class="serre">In principio creavit Deus caelum et terram terra autem erat
    inanis et vacua et tenebrae super faciem abyssi et spiritus. . .</p>
 </body>
</html>
```

En appliquant ce code, nous obtenons le résultat présenté en figure 12-9.

Figure 12-9
Réglages des interlignes

Définir plusieurs propriétés en une fois

Comme nous l'avons déjà vu avec d'autres propriétés, `border` par exemple, il est aussi possible de définir plusieurs propriétés de police à l'aide d'une seule propriété. En effet, la propriété `font` permet en une seule définition de créer des styles en définissant des valeurs pour les propriétés : `font-weight`, `font-variant`, `font-style`, `font-size`, `line-height` et `font-family`. Par rapport à la propriété `font-family`, `font` offre des options supplémentaires pour le choix de la police. Nous pouvons en effet définir des styles de polices ayant les caractéristiques de celles utilisées pour le système d'exploitation lui-même.

Sa syntaxe générale est la suivante :

```
font: [[<font-style> || <font-variant> || <font-weight>] ? <font-size> [/ <line-
➥ height>] ? <font-family> ] | caption | icon | menu | message-box | small-caption
➥ | status-bar | inherit
```

Chacun des paramètres de cette propriété globale peut prendre les valeurs qui correspondent aux propriétés individuelles que nous venons d'aborder dans les sections précédentes.

Nous pouvons par exemple créer le style suivant :

```
p{font:italic small-caps bold 14px/18px Arial, sans-serif
```

Cette déclaration est alors équivalente à l'ensemble des déclarations suivantes :

```
p{font-style:italic; font-variant:small-caps; font-weight:bold; font-size:14px;
➥ line-height:18px; font-family:Arial,sans-serif;}
```

Les valeurs système sont définies de la manière suivante :

- `caption` : font référence à la police système utilisée pour les boutons et les listes déroulantes des formulaires.
- `icon` : police système de légendes des icônes du bureau ;
- `menu` : police des menus des applications ;
- `message-box` : police des boîtes de dialogue (boîte d'alerte par exemple) ;
- `small-caption` : police des contrôles ;
- `status-bar` : police de la barre d'état.

L'utilisation de ces valeurs dans la propriété `font` implique l'acceptation de toutes les caractéristiques de la police résultante. Pour modifier une caractéristique particulière, comme la taille, il faut ensuite redéfinir la propriété concernée. Notez que Explorer ne prend pas en compte cette redéfinition de la taille et affiche la police sans changer ses caractéristiques.

Ainsi, nous pouvons écrire :

```
p{font:menu; font-size: 1em;}
```

pour obtenir la police système des menus, et redéfinir sa taille séparément dans Mozilla par exemple.

L'exemple 12-10 présente ci-après une utilisation possible de la propriété `font`.

L'alignement et l'espacement du texte

Afin d'améliorer ou de rendre plus originale la présentation du texte, nous disposons de plusieurs propriétés d'alignement, d'indentation et d'espacement.

L'alignement horizontal du texte

La propriété `text-align`, dont la syntaxe est la suivante :

```
text-align: left | center | right | justify | inherit
```

n'est applicable qu'aux éléments de blocs. Selon le mot-clé choisi, le texte sera aligné respectivement à gauche, au centre, à droite, ou justifié. Le style défini étant transmis aux éléments enfants, il faut donc prendre des précautions et éventuellement définir un autre style pour ces éléments enfants.

Dans l'exemple 12-10, le contenu des éléments `<h1>` (repère ´) est centré (repère ³), et celui des éléments `<h2>` aligné à gauche, soit sa valeur par défaut (repère ·). Avec la classe `p.droit` (repère ») on peut aligner le contenu du premier paragraphe (repère ²) à droite, et avec la classe `p.justifie` (repère ¿) on peut justifier le texte du second (repère ¶) dans les navigateurs les plus actuels, et même dans Explorer 6.

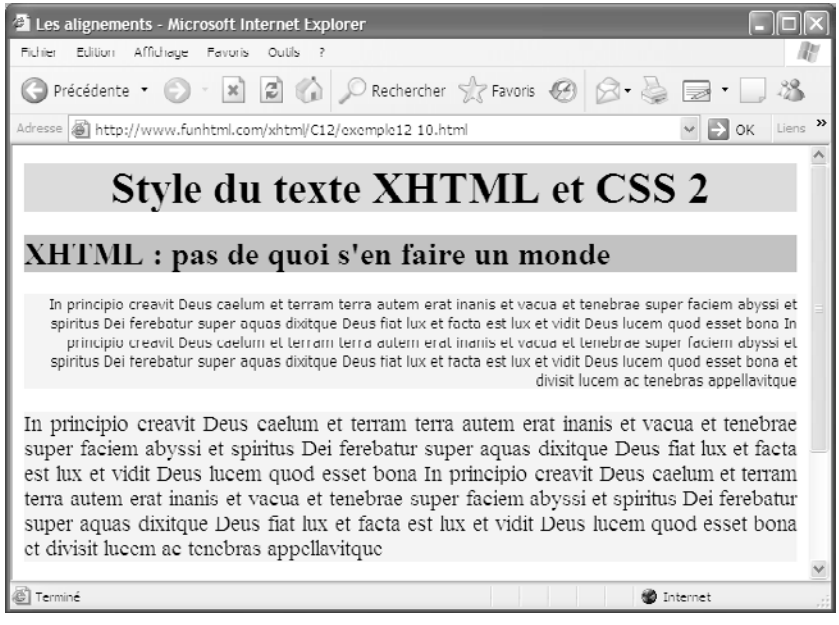
Exemple 12-10. Les styles d'alignement du texte

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les alignements</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="align">
   h1{text-align: center; background-color: #CCC; }  ³
   h2{text-align: left; background-color: #AAA;}     ·
   p.droit{font: menu; font-size: 1.5em; text-align: right ;
     ➥ background-color: #EEE;}»
   p.justifie{text-align: justify; background-color: #EEE;}        ¿
  </style>
 </head>
```

```
<body>
    <h1>Style du texte <span>XHTML et CSS 2</span></h1>
  <h2>XHTML : pas de quoi s'en faire un monde</h2>
    <p class="droit">In principio creavit Deus caelum et terram terra autem erat
    inanis et vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur
    . . .</p>
    <p class="justifie">In principio creavit Deus caelum et terram terra autem
    erat inanis et vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur
    super . . .</p>
</body>
</html>
```

La figure 12-10 montre les différents types d'alignement.

Figure 12-10
Les styles d'alignement du texte



L'indentation du texte

L'indentation du texte est définie par défaut pour certains éléments comme `<blockquote>` ou les éléments de liste `<ul>` et `<ol>` par exemple. Nous pouvons définir une indentation personnalisée mais il faut noter qu'elle ne s'applique qu'à la première ligne de texte, ce qui, pour un titre, ne présente généralement pas de problème, mais ne correspond pas forcément à l'effet désiré pour un paragraphe. Pour décaler tout un paragraphe, il vaudra mieux avoir recours soit à la définition d'une marge, soit à un positionnement de l'élément (voir chapitre 13).

L'indentation du texte peut être définie à l'aide de la propriété `text-indent`, dont la syntaxe est la suivante :

```
text-indent:<long>|<pourcent>|inherit
```

Elle est applicable aux éléments de bloc, aux cellules des tableaux, et elle est héritée.

Les valeurs d'indentation peuvent être définies à l'aide d'un nombre et d'une unité de longueur habituelle (`em`, `px`, `ex`, `mm`, `cm`, `in`, `pt`, `pc`) ou en pourcentage. Dans ce dernier cas, l'indentation est calculée par rapport à la largeur de l'élément qui contient le texte. Les valeurs négatives sont admises pour créer une saillie à gauche (pour le sens de lecture européen avec `dir = "ltr"`) ou le contraire, sinon par rapport à l'alignement normal.

Dans l'exemple 12-11, le titre `<h1>` (repère ²) est indenté de 20 % de la largeur de l'élément, soit, faute de dimensionnement, de 20 % de la largeur de la fenêtre du navigateur (repère ³). Le titre `<h2>` est indenté de 5 em (repère ·). Quand l'indentation s'applique à des éléments imbriqués, nous pouvons obtenir des effets intéressants. L'indentation de l'élément `<div>` est fixée à 15 % (repère »), ce qui s'applique à son contenu direct (repère ¾). Pour l'élément `<p>` qu'elle contient, l'indentation est négative (repère ¿) et elle contraint à définir une marge gauche, sinon le texte « XHTML 1.1 » n'est plus visible, car il disparaît sur la gauche de la page. Les listes (repère ¸) ont une indentation par défaut mais nous pouvons la fixer à la valeur désirée pour personnaliser leur présentation (repère ´).

Exemple 12-11. L'indentation du texte

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Indentation du texte</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="indentattion">
   h1{text-indent: 20%; background-color: #CCC; }³
   h2{text-indent: 5em; background-color: #AAA;}
   div{text-indent: 15%; background-color: #EEE;}
   p.ret{text-indent: -5em; margin-left: 10em; background-color: #DDD;}
   ol{text-indent: 15ex; background-color: #CCF; list-style-position: inside;}
  </style>
 </head>
 <body>
  <h1>Style du texte <span>XHTML et CSS 2</span></h1>
```

```
<h2>XHTML 1.1 : l'avenir est là</h2>
<div>
 Le langage XHTML dans sa version la plus actuelle : In principio creavit Deus
    ➥ XHTML et CSS terra autem erat inanis et vacua et tenebrae super faciem abyssi
    ➥ et spiritus Dei ferebatur super aquas dixitque Deus fiat lux et facta est lux
 <p class="ret">XHTML 1.1 : In principio creavit Deus caelum et terram terra autem
    ➥ erat inanis et vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur
    ➥ super aquas dixitque Deus fiat lux et facta est lux et vidit Deus lucem quod
    ➥ esset bona In principio creavit Deus caelum et terram terra autem erat inanis
    ➥ et vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
    ➥ dixitque Deus fiat lux et facta est lux et vidit Deus lucem quod esset bona et
    ➥ divisit lucem ac tenebras appellavitque</p>
 <ol>
  <li>XHTML 1.0 strict</li>
  <li>XHTML 1.0 frameset</li>
  <li>XHTML 1.1</li>
 </ol>
 </div>
</body>
</html>
```

La figure 12-11 montre l'affichage obtenu. Les couleurs de fond permettent de situer les différentes boîtes de contenu des éléments.

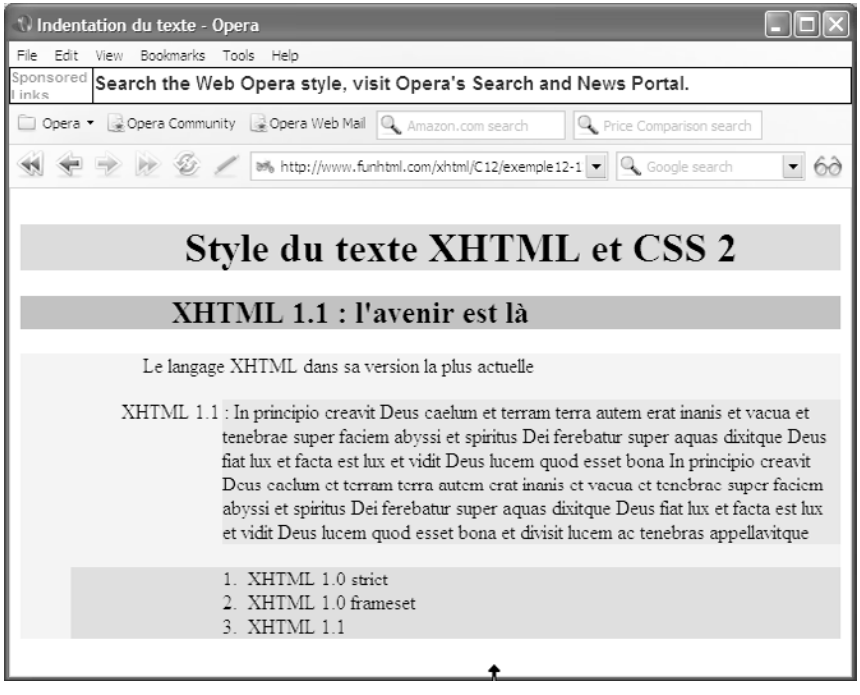


Figure 12-11
L'indentation du texte

L'espacement des mots et des caractères

Chaque police est définie avec un espacement donné entre chaque caractère et une largeur fixe qui crée l'espacement entre chaque mot. Nous pouvons intervenir sur l'espacement des caractères en le définissant au moyen de la propriété `letter-spacing`, dont la syntaxe est la suivante :

```
letter-spacing:normal|<long>|inherit
```

En utilisant le mot-clé `normal` nous définissons explicitement un espacement égal à la valeur initiale fixée par la police. Pour créer un espacement personnalisé, il faut utiliser une valeur de longueur définie par un nombre et une unité relative ou absolue (em, px, ex, mm, cm, in, pt, pc). La valeur indiquée dans le paramètre `<long>` n'est pas celle de l'espacement lui-même mais celle qui s'ajoute ou se soustrait à la valeur normale associée à la police utilisée. Le mot-clé `normal` correspond à cette valeur, et donc à la valeur `0` de `<long>` Les valeurs négatives sont donc utilisées pour diminuer l'espacement. Agir sur l'espacement des lettres risque de donner l'aspect à la police utilisée d'une police à espacement fixe telle que nous pouvons le définir en définissant la propriété `font-family` à la valeur `Courier New` ou le mot-clé `monospace`

La définition de cette propriété implique que l'espacement entre les mots d'une phrase soit aussi conditionné par la valeur choisie pour `letter-spacing`. Afin de définir indépendamment l'espacement entre les mots, il faut utiliser la propriété `word-spacing` dont la syntaxe est la même que précédemment :

```
word-spacing : normal | <long> | inherit
```

Elle est également héritée et les valeurs négatives sont admises pour diminuer l'espacement. L'exemple 12-12 permet de mettre en œuvre ces deux propriétés et d'effectuer des comparaisons d'aspect pour différentes valeurs. Les styles appliqués à l'élément `<p>` (repère ³) créent un texte très resserré pour le premier paragraphe (repère ¿). Les éléments `<span>` ont au contraire un style (repère ·) qui permet d'étirer le texte de leurs contenus (repères ´ , ² , º et ¾), indépendamment du style de leur parent. La classe `space` (repère ») permet d'afficher le même texte avec un espacement beaucoup plus grand dans le second paragraphe (repère ¶).

Exemple 12-12. L'espacement des lettres et des mots

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>L'espacement des lettres et des mots</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="liens">
   p{letter-spacing: -0.1em; word-spacing: 0.7em;} ³
   span{letter-spacing: 10px; background-color: #EEE;} ·
   .space{letter-spacing: 0.2em; word-spacing: 1em;} »
```

```
 </style>
</head>
<body>
  ¿ <p>Le langagé  <span>XHTML 1.1 </span>: Les éléments html, head et body
  ➥ constituent la base de la structure d'une page  ² <span>XHTML</span>.
  ➥ In principio creavit Deus caelum et terram terra autem erat inanis et . . .</p>
  ¶ <p class="space">Le langage <span>XHTML 1.1 </span>: Les éléments html, head
  ➥ et body constituent la base de la structure d'une page  ¾ <span>XHTML</span>.
  ➥ In principio creavit Deus caelum et terram terra autem erat inanis et . . .</p>
</body>
</html>
```

La figure 12-12 montre les grandes différences de présentation obtenues avec ces styles.

Figure 12-12

L'espacement des lettres et des mots

La dernière possibilité de gestion des espaces est offerte par la propriété `white-space` qui permet de choisir le mode d'affichage des différents caractères d'espacement placés dans le code XHTML. Nous retrouvons ici l'équivalent de ce qu'il est possible de faire avec l'élément `<pre>` Sa syntaxe est la suivante :

```
white-space: normal | pre | nowrap | inherit
```

La valeur `normal` crée l'affichage habituel des navigateurs (un seul espace dans la page, quel qu'en soit le nombre dans le code). La valeur `pre` préserve tous les caractères d'espacement présents dans le code. On peut alors directement positionner les différents

contenus d'un élément dans l'éditeur de code, en particulier pour afficher des listings. La valeur `nowrap` gère les espaces de la même façon que la valeur `normal` à la différence quelle désactive en plus les sauts de lignes, le contenu étant donc affiché sur une seule ligne, le visiteur est donc contraint d'effectuer un défilement horizontal, ce qui ne facilite pas la lecture du contenu s'il est long.

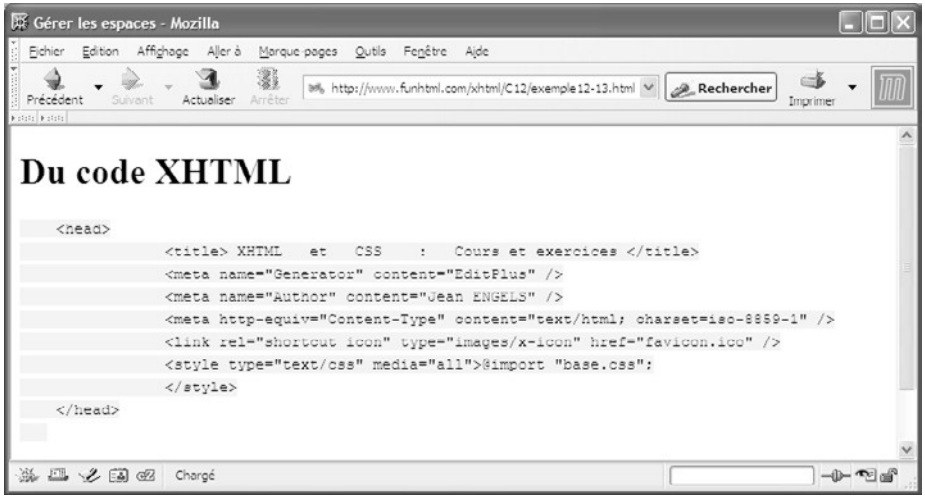
L'exemple 12-13 présente une illustration de cette propriété avec la valeur `pre` (repère ³) pour afficher du code XHTML contenu dans un élément `<code>` (repère ·).

Exemple 12-13. Gestion des espaces

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Gérer les espaces</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="espaces">
   code{white-space: pre; background-color: #EEF;}³
  </style>
 </head>
 <body>
  <h1>Du code XHTML</h1>
  <p>
   <code>·
    &lt;head&gt;
        &lt;title&gt; XHTML et CSS : Cours et exercices &lt;/title&gt;
        &lt;meta name="Generator" content="EditPlus" /&gt;
        &lt;meta name="Author" content="Jean ENGELS" /&gt;
        &lt;meta http-equiv="Content-Type" content="text/html;
            ➥ charset=iso-8859-1" /&gt;
        &lt;link rel="shortcut icon" type="images/x-icon" href="favicon.ico" /&gt;
        &lt;style type="text/css" media="all"&gt;@import "base.css";
       &lt;/style&gt;
    &lt;/head&gt;
   </code>
  </p>
 </body>
</html>
```

La figure 12-13 montre qu'avec la valeur `pre` tous les espaces et sauts de lignes sont bien préservés dans la page affichée.

Figure 12-13
La gestion des espaces



En utilisant la valeur `normal` ou encore en ne définissant pas la propriété, nous obtiendrions le résultat présenté ci-après, qui est bien peu lisible par rapport au précédent.

```
<head> <title> XHTML et CSS : Cours et exercices </title> <meta name=
"Generator" content="EditPlus" /> <meta name="Author" content="Jean ENGELS" />
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<link rel="shortcut icon" type="images/x-icon" href="favicon.ico" />
<style type="text/css" media="all">@import "base.css"; </style> </head>
```

Le style des liens

En faisant l'inventaire des propriétés CSS, il faut bien reconnaître qu'aucune d'entre elles ne s'applique spécifiquement aux liens hypertextes. Cependant, il n'est sans doute pas inutile de consacrer une section aux styles que nous pouvons appliquer aux liens généralement créés avec un élément `<a>` Un lien présente en effet plusieurs états différents lors de la consultation d'une page par un visiteur. Lors du premier chargement de la page, il est vierge et prêt à être activé. Au moment du clic sur le texte ou l'image, il est dit actif. Après ce clic, que le lien soit interne ou externe, l'origine du lien est dans l'état dit « visité ». À chacun de ces états correspondent des styles, et en particulier des couleurs par défaut relativement standard et définies dans la feuille de style par défaut préconisée par le W3C (voir l'annexe B). J'ai déjà indiqué qu'il fallait manier les couleurs des liens avec précaution dans la mesure où l'internaute lambda est habitué à ces styles (couleur généralement bleue et soulignement). Cependant, on peut être contraint de les modifier, par exemple pour s'adapter à une couleur de fond particulière de la page.

Pour créer des styles propres aux liens, nous utilisons les pseudo-classes qui leur sont dédiées et des propriétés déjà rencontrées par ailleurs comme la couleur du texte, celle du fond ou la définition d'une police particulière.

Les pseudo-classes :link , :visited et :active sont prévues à cet effet, les deux premières n'étant applicables qu'aux liens. Nous rappelons ici leurs définitions, déjà vues au chapitre 9 ;

- :link , qui s'applique au lien non encore visité avant toute action du visiteur. Elle permet de définir les styles du lien, généralement textuel.
- :visited , qui s'applique au lien qui a déjà été visité au moins une fois. Elle permet par exemple de changer la couleur du lien.
- :active , qui s'applique au lien au moment précis où l'utilisateur maintient le bouton de la souris enfoncé, quand le curseur est positionné sur le texte du lien. Cette pseudo-classe est mal prise en compte par les navigateurs et ne présente pas réellement un intérêt fondamental aujourd'hui.

Nous pouvons également ajouter les pseudo-classes :focus et :hover qui ne sont pas spécifiques aux liens mais que nous pouvons utiliser :

- :focus, qui permet de modifier le style d'un lien quand il reçoit le focus à l'aide de la touche Tab ou d'un raccourci clavier.
- :hover, qui permet de donner un style particulier quand le curseur est positionné sur le lien. L'effet réalisé, comme un changement de couleur par exemple, est réversible instantanément quand le curseur quitte la zone du lien.

L'exemple 12-14 illustre l'emploi de ces différentes pseudo-classes. La page contient une liste à puce de liens (repères ², ¶ et º). Le sélecteur a:link crée le style de base des liens soulignés et de couleur bleue (repère ³). Les liens ayant le focus ont une taille de police double de la normale et une bordure double jaune (repère ·). L'usage de la pseudo-classe :hover nécessite un sélecteur plus complexe ul li a:hover , car le sélecteur simple a:hover ne fonctionne pas dans la plupart des navigateurs. Il permet de gérer une taille de police de 25 pixels, une bordure, une couleur de fond et de texte modifiée (repère »). Les liens visités sont affichés en vert sur fond gris (repère¿), et les liens activés en rouge sur fond noir (repère ´).

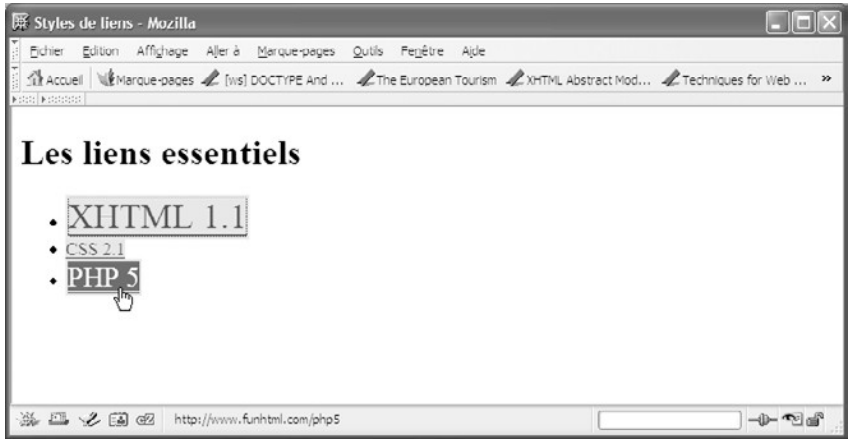
Exemple 12-14. Les styles des liens

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Styles des liens</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="liens">
   a:link{text-decoration: underline;color: navy;}
```

```
  a:focus{font-size:200%;border: yellow 2px double;}
  ul li a:hover{font-size: 25px; background-color: red; color: #FFF;
    border: yellow 2px double;}
  li a:visited{background-color: #DDD; color: #700;}
  li a:active{background-color: #red; color: #567;}
 </style>
</head>
<body>
 <h1>Les liens essentiels</h1>
 <ul>
    <li><a href="http://www.w3.org/TR/xhtml11/" tabindex="1">XHTML 1.1</a></li>
    <li><a href="http://www.w3.org/TR/CSS21/" >CSS 2.1</a></li>
    <li><a href="http://www.funhtml.com/php5">PHP 5</a></li>
 </ul>
</body>
</html>
```

La figure 12-14 présente les effets obtenus. Dans cette page, le lien « XHTML 1.1 » a le focus et le lien « CSS 2.1 » est survolé par le curseur, mais pas cliqué.

Figure 12-14
Les styles des liens



Exercices

Exercice 1 : Définissez la police Georgia pour un paragraphe et ajouter le nom de la police générique correspondante.

Exercice 2 : Écrivez le style de façon à ce que les éléments `<code>` et `<samp>` aient une police à espacement fixe.

Exercice 3 : Créez les styles afin que les titres `<h1>` soit affichés avec une police Arial de 24 pixels et que les titres `<h2>` soient dans une police de la famille `serif` avec une taille de 1,2 em.

Exercice 4 : Définissez la taille des titres <h1><h2><h3> avec des pourcentages décroissants et une police de la famille sans-serif. La taille fixée pour <h1> doit être l'équivalent de la taille 2 em.

Exercice 5 : Faites la même opération que pour l'exercice 4 en utilisant des tailles absolues, puis des tailles en pixel.

Exercice 6 : Créez une classe qui permette d'obtenir un texte plus grand que celui de l'élément parent.

Exercice 7 : Définissez explicitement le style qui permet d'obtenir le contenu des éléments <b> en gras et plus grand que son contexte.

Exercice 8 : Définissez les styles afin que le contenu de l'élément <code> soit dans une police à espacement fixe, plus grande et plus grasse que celle de son parent.

Exercice 9 : Définissez explicitement le style de l'élément <i> en italique gras.

Exercice 10 : Créez les styles de façon à ce que les titres <h3> et la première ligne des paragraphes apparaissent en petites majuscules.

Exercice 11 : Définissez les styles pour obtenir l'affichage du contenu des éléments <code> et <samp> en minuscules et dans une police monospace

Exercice 12 : Créez les styles de façon à ce que les éléments <h3> soient affichés avec un texte clignotant.

Exercice 13 : Réglez l'interligne des éléments à 1,2 fois la valeur normale et celui des paragraphes à 80 % (il y a deux possibilités).

Exercice 14 : Définissez le style du texte de l'élément <blockquote> avec la police système message-box

Exercice 15 : Définissez l'alignement des éléments <code> à gauche avec un espacement de 0,2 em et ceux des éléments <h2> au centre.

Exercice 16 : Créez le style permettant la préservation des espaces dans l'élément <blockquote>

Exercice 17 : Créez les styles pour que les liens vierges soient rouges et en gras, que les liens visités soient marrons sur fond gris clair et en graisse normale. Les liens survolés doivent apparaître entourés d'une bordure bleu vif.

13

Créer une mise en page : le dimensionnement et le positionnement

Les méthodes de mise en page au moyen de cadres (telles que nous les avons vues au chapitre 8) ou à l'aide de tableaux (chapitre 6), outre qu'elles sont aujourd'hui considérées comme obsolètes, ne manquent pas de présenter des inconvénients importants. Si les cadres gênent considérablement l'indexation correcte des pages web, les mises en page à base de tableaux, pour pratiques qu'elles aient l'air à première vue, impliquent des structures trop complexes qui nuisent à la lisibilité de l'organisation et à la maintenance des sites. Nous allons voir dans ce chapitre que tous ces problèmes peuvent être contournés en utilisant conjointement le dimensionnement et le positionnement des éléments qui contiennent les différentes zones d'une page. Ces méthodes apportent enfin des réponses simples permettant de créer les mises en page les plus diverses, et qui plus est en respectant le principe, fondamental en XHTML, de séparation du contenu et de la présentation. Nous pouvons par exemple obtenir des présentations différentes à partir du même code XHTML en modifiant simplement les styles CSS attachés à ces différents composants.

Le dimensionnement des éléments

Au chapitre 11, dans la définition du modèle de boîte de CSS, nous avons vu que les propriétés width et height permettent de déterminer respectivement la largeur et la hauteur d'un élément. Elles s'appliquent à tous les éléments sauf les éléments en ligne non remplacés et les lignes et groupes de lignes des tableaux. Nous rappelons leurs syntaxes :

```
width: <longueur> | NN% | auto | inherit
height: <longueur> | NN% | auto | inherit
```

Si la définition de la largeur ne pose généralement pas de problème, celle de la hauteur peut engendrer des effets particuliers, voire conflictuels entre les boîtes créées pour les éléments successifs et leurs contenus. L'exemple 13-1 en est une illustration concrète. Le corps du document contient une division (repère ¿) qui inclut successivement un titre <h1> (repère ´), deux paragraphes <p> (repères ² et ¶). Les paragraphes incluent pour leur part du texte ordinaire. L'élément est suivi d'un second paragraphe ne contenant que du texte (repère º).

Les définitions de styles fixent les dimensions de l'élément qui a une largeur de 80 % et une hauteur de 500 pixels (repère ³). Les éléments <p> ont une largeur de 75 % de celle de leur conteneur (repère ·), soit 75 % de 80 % (donc 60 % de la largeur de la page) pour le premier et 75 % de la page pour le second. Ces largeurs sont effectivement respectées par tous les navigateurs (voir la figure 13-1). De même, l'élément <h1> a une largeur réelle de 80 % de celle de son parent , donc 80 % de 80 %, soit 64 % de celle de la page.

Mais c'est au niveau de la définition des hauteurs que se posent les problèmes. Nous pouvons remarquer d'emblée que la hauteur de l'élément est supérieure à la somme des hauteurs de ses éléments enfants <p> (200 pixels chacun) et <h1> (50 pixels). Le résultat obtenu visible à la figure 13-1 montre tout d'abord que la hauteur définie pour est bien respectée. Celles des deux paragraphes sont bien de 200 pixels, la couleur de fond qui leur est attribuée nous permettant de le vérifier. Il en est de même pour la hauteur de l'élément <h1> En revanche, si le texte du premier paragraphe s'affiche bien dans la boîte assignée à l'élément <p> celui du deuxième déborde, et pire encore il se superpose au contenu du troisième. Le résultat obtenu est évidemment catastrophique au niveau de la présentation.

Exemple 13-1. Les problèmes de dimensionnement en hauteur

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Dimensionnement des éléments</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
```

```
 <style type="text/css" >
  body {font-family: Arial sans-serif;font-size: 16px;}
  div{width:80%;height:500px;background-color:#EEE;}
  p{width:75%;height:200px;background-color:#BBB;}
  h1{width: 80%; height: 50px;background-color:#888;color:white;} »
 </style>
</head>
<body>
 ¿ <div>
    <h1>Dimensionnement des éléments</h1>
  ² <p>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
  ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
 </p>
  ¶ <p>Deus duo magna luminaria luminare maius ut praeesset diei et luminare minus
  ➥ ut praeesset nocti et stellas et posuit eas in firmamento caeli . . .</p>
  </div>
  º <p>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
  ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas . . .
 </p>
</body>
</html>
```

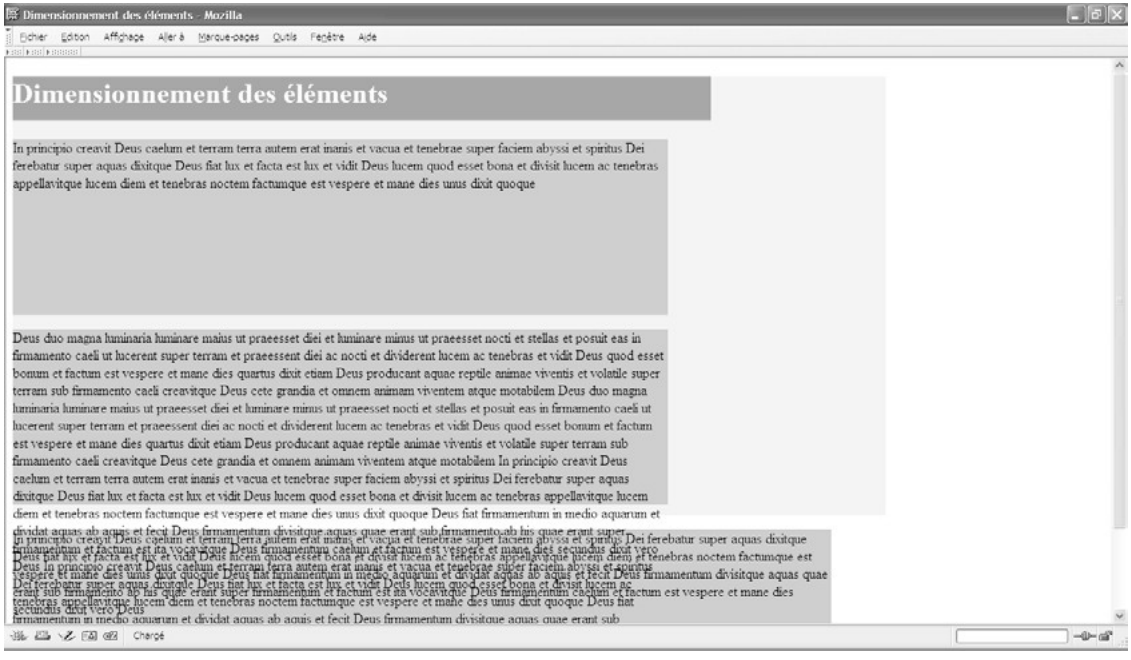


Figure13-1

Les problèmes dus au dimensionnement

Nous pouvons corriger facilement ce résultat en donnant la valeur auto à la propriété height des éléments et <p> (repères ³ et ·). L'élément <style> devient donc :

```
<style type="text/css" >
 body {font-family: Arial sans-serif;font-size: 16px;}
 div{width:80%;height:auto ³  ;background-color:#EEE;}
 p{width:75%;height:auto ·  ;background-color:#BBB;}
 h1{width: 80%; height: 50px;background-color:#888;color:white;}
</style>
```

Nous obtenons alors le dimensionnement présenté à la figure 13-2, dans lequel chaque paragraphe occupe la place qui lui est nécessaire pour afficher son contenu.

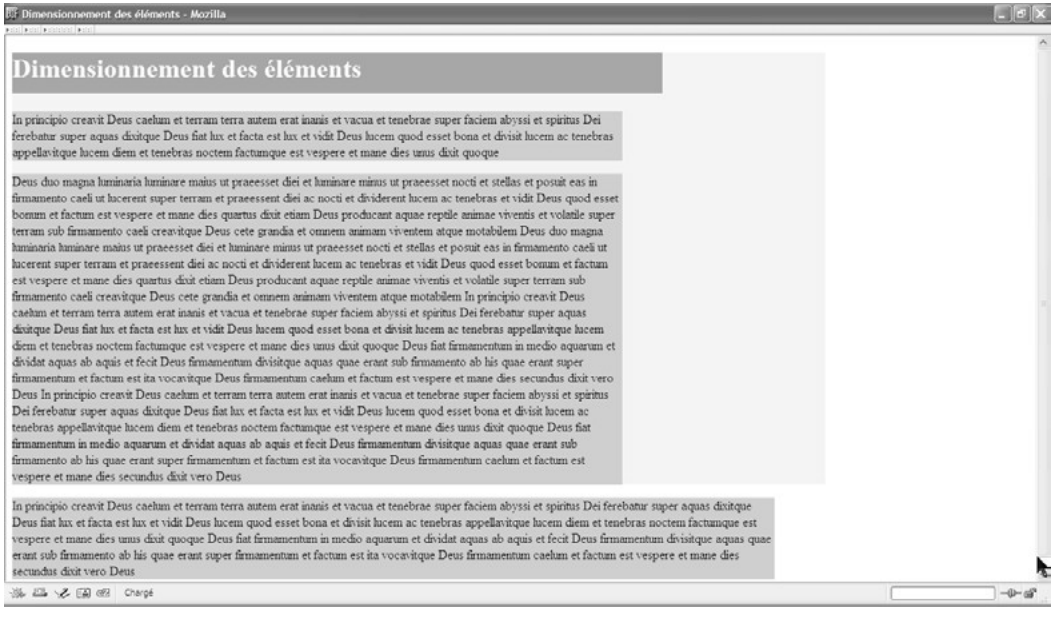


Figure 13-2

Le dimensionnement avec la valeur auto

L'utilisation de la valeur auto ne permet pas une mise en page précise des différents éléments dans le document, en particulier si leur contenu est créé dynamiquement, en provenance d'une base de données, suite à une recherche déclenchée par le visiteur par exemple. S'il semble nécessaire au programmeur de définir impérativement une hauteur à chaque élément, il est possible de conserver cette présentation en appliquant la propriété overflow pour gérer les éventuels débordements des contenus. Sa syntaxe est la suivante :

```
overflow : visible | hidden | scroll | auto | inherit
```

Voici le rôle imparti à chacune de ses valeurs :

- visible : le contenu débordant est affiché.

- hidden : le contenu débordant est caché et donc illisible.
- scroll : des barres de défilement horizontales et verticales apparaissent sur les côtés droit et bas de la boîte de l'élément, ce qui permet d'accéder au contenu débordant. Cette valeur a l'inconvénient de laisser apparaître ces barres même si le contenu ne déborde pas.
- auto : le navigateur fait apparaître les barres de défilement en cas de débordement uniquement.

En modifiant les styles de l'exemple 13-1 de la manière suivante :

```
<style type="text/css" >
 body {font-family: Arial sans-serif;font-size: 16px;}
 div{width:80%;height:500px;background-color:#EEE;}
 p{width:75%;height: 200px;background-color:#BBB;overflow: auto;}
 h1{width: 80%; height: 50px;background-color:#888;color:white;}
</style>
```

dans laquelle nous définissons la propriété overflow pour les paragraphes avec la valeur auto (repère ③), nous obtenons le résultat présenté à la figure 13-3 qui montre que la hauteur définie pour les éléments <p> est conservée à 200 pixels, mais que le contenu du deuxième paragraphe est lisible en utilisant les barres de défilement. Les intentions de mise en page sont donc préservées et le contenu est entièrement accessible.

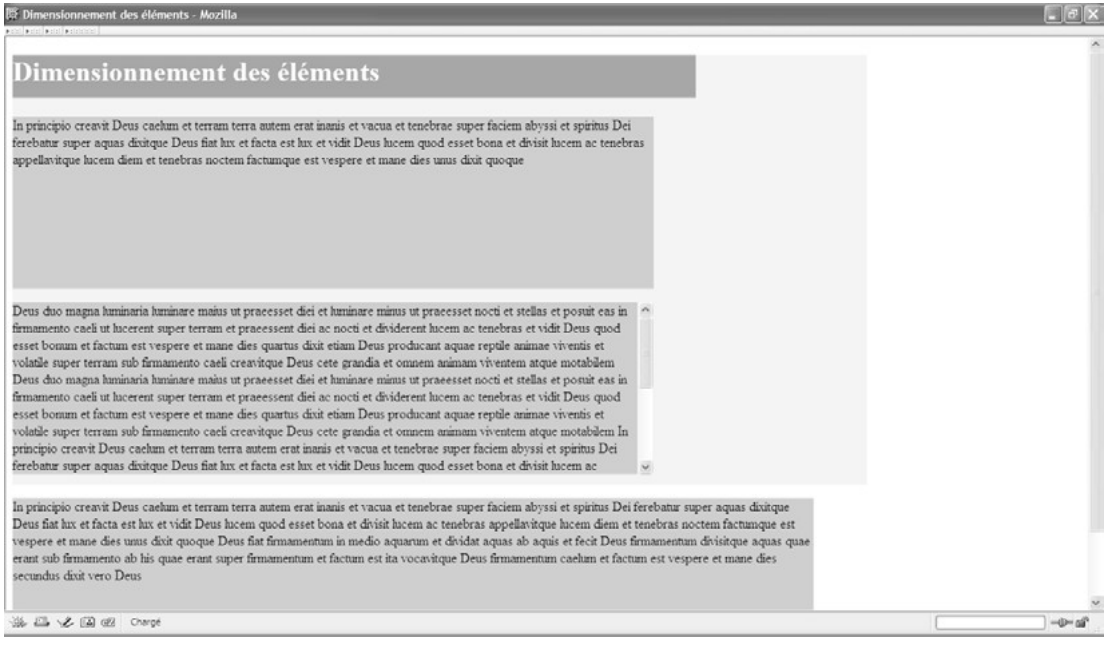


Figure 13-3

Gestion des débordements

Dans les cas précédents, et en gérant les débordements avec la propriété overflow, tous les paragraphes ont les mêmes dimensions quels que soient leurs contenus. Dans notre dernier exemple, le premier et le second ont une hauteur de 200 pixels alors que le premier ne contient que quelques lignes. Pour pouvoir intervenir sur les hauteurs des éléments sans pour autant les fixer de manière absolue, nous pouvons définir une hauteur minimale et une hauteur maximale pour tous les éléments, à l'exception des éléments en ligne non remplacés et des éléments de tableau.

Nous disposons pour cela des propriétés min-height et max-height dont la syntaxe est la suivante :

```
min-height: <longueur> | NN% | inherit
max-height: <longueur> | NN% | none |inherit
```

Le paramètre <longueur> est donné comme d'habitude par un nombre et une unité, et les pourcentages sont donnés en référence à la hauteur du bloc conteneur. La valeur none (qui est la valeur par défaut pour la hauteur maximale) indique que la valeur limite est celle du bloc conteneur.

De même, une largeur minimale et une largeur maximale peuvent être indiquées pour les mêmes éléments à l'aide des propriétés min-width et max-width dont les syntaxes sont similaires aux précédentes :

```
min-width: <longueur> | NN% | inherit
max-width: <longueur> | NN% | none |inherit
```

L'exemple 13-2 nous permet d'affiner les mises en pages obtenues dans les exemples précédents. L'élément se voit affecté une largeur maximale de 900 pixels et une hauteur maximale de 800 pixels (repère ³). Les paragraphes ont une hauteur minimale de 80 pixels et maximale de 250 pixels (repères · et »), ainsi qu'une largeur comprise entre 500 et 600 pixels (repères ¿ et ´). Les éventuels débordements sont gérés en utilisant la propriété overflow (repère ²). L'élément <h1> a pour sa part une largeur comprise entre 400 et 500 pixels au maximum (repères ¶ et º). Comme nous pouvons le constater sur la figure 13-4, le contenu du titre <h1> est affiché sur deux lignes car sa largeur est limitée à 400 pixels. Le premier élément <p> (repère ¾) occupe juste la hauteur nécessaire à son contenu limité à quatre lignes. En revanche, le deuxième (repère µ) a un contenu beaucoup plus long et il apparaît avec une hauteur de 250 pixels, ce qui est sa limite supérieure, et, comme son contenu déborde, des barres de défilement apparaissent automatiquement. Quant au troisième paragraphe (repère ¸), il a le même comportement que le premier et sa hauteur est exactement adaptée à son contenu en respectant les contraintes de hauteur.

Exemple 13-2. Largeurs et hauteurs minimales et maximales

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html  xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Dimensionnement des éléments</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
```

```
 <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 <style type="text/css" >
  body {font-family: Arial sans-serif;font-size: 16px;}
  div{max-width:900px;max-height:800px;background-color:#EEE;}
  p{min-height:80px ; max-height:250px ; min-width:500px ;
    ➥ max-width:600px ; background-color:#BBB;overflow: auto ;}
  h1{min-width: 400px ; max-width: 500px ; background-color:#888;color:white;}
 </style>
</head>
<body>
 <div>
 <h1>Dimensions mini et maxi des éléments XHTML</h1>
  <p> In principio creavit Deus caelum et terram terra autem erat inanis et vacua
  ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas dixitque
  ➥ . . . </p>
  <p>Deus duo magna luminaria luminare maius ut praeesset diei et luminare minus
  ➥ ut praeesset nocti et stellas et posuit eas in firmamento caeli ut lucerent
  ➥ super terram et praeessent diei ac nocti et dividerent. . . </p>
 </div>
  <p> In principio creavit Deus caelum et terram terra autem erat inanis et vacua
  ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas dixitque
  ➥ Deus fiat lux et facta est lux et vidit Deus lucem quod . . .</p>
 </body>
</html>
```

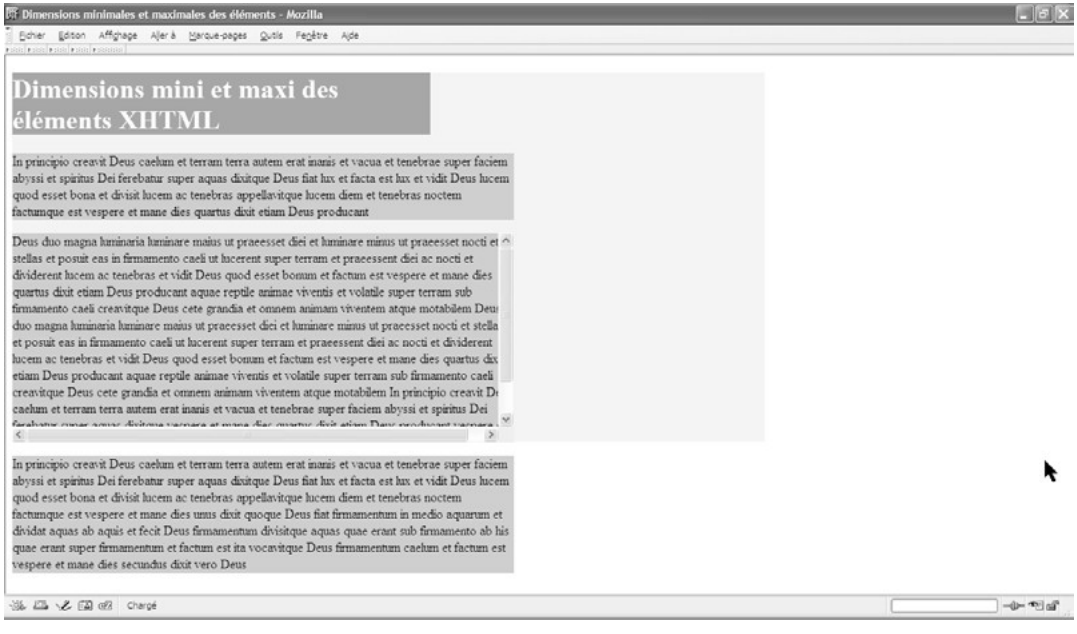


Figure 13-4

Les dimensions minimales et maximales

Le rendu des éléments

Dans la première partie de cet ouvrage, nous avons passé en revue l'ensemble des éléments XHTML et nous avons vu que chacun d'eux est associé par défaut à une présentation spécifique (à l'aide d'une feuille de style par défaut incluse dans les navigateurs : voir l'annexe B). Tous les éléments peuvent donc être classés en grands groupes de mise en page comme les blocs, les éléments en ligne, les listes ou les tableaux avec les caractéristiques qui s'y rattachent. Pour chaque élément, nous pouvons intervenir sur l'appartenance à un de ces groupes et modifier le rendu d'un élément en fonction des besoins. Un élément `<h1>` peut par exemple être transformé en élément en ligne ou en élément de liste alors qu'il est par défaut de niveau bloc.

Ces modifications sont opérées au moyen de la propriété `display` dont la syntaxe est la suivante :

```
display : none |inline | block | list-item |table | inline-table | table-row-group |
➥ table-header-group | table-footer-group | table-row | table-column-group | table-
➥ column | table-cell | table-caption | inherit
```

Sa valeur par défaut est `inline` mais la feuille de style par défaut la modifie et donne à chaque élément le style que nous lui connaissons. Ce sont ces valeurs par défaut sur lesquelles nous intervenons. Les principales valeurs qu'elle peut prendre ont la signification suivante :

- `none` : l'élément n'est pas affiché et la boîte qui lui est associée n'est pas créée. Tout se passe comme s'il n'existait pas dans le code XHTML.
- `inline` : l'élément devient du type en ligne (comme `<span>` par exemple).
- `block` : l'élément devient du type bloc (comme `<h1>`, `<p>`, `<div>`…).
- `list-item` : l'élément devient du type liste (équivalent de `<li>`).

Les autres valeurs sont rarement utilisées en XHTML, mais plutôt réservées à la création de styles destinés à des documents XML pour induire des comportements identiques à ceux des éléments de tableau XHTML. Le tableau suivant donne les correspondances entre les valeurs de la propriété `display` et l'élément XHTML dont le comportement est induit par celles-ci.

Tableau 13-1. Correspondances entre les valeurs de la propriété display et les éléments XHTML

Valeur	Élement	Valeur	Élement
table	<table>	table-row-group	<tbody>
table-header-group	<thead>	table-footer-group	<tfoot>
table-row	<tr>	table-column-group	<colgroup>
table-column	<col />	table-cell	<td> ou <th>
table-caption	<caption>	inline-table	<table> dont la propriété display vaut inline

L'exemple 13-3 offre une illustration de la propriété display. La page comprend un paragraphe <p> qui inclut quatre éléments (repère ¿). Ces éléments, qui sont normalement de type en ligne, sont affichés sous forme de liste en donnant la valeur list-item à la propriété display (repère ³). La page contient également une liste non ordonnée (repère ´). En donnant la valeur inline à display pour les éléments <li> (repère ·), ils constituent un menu sur une seule ligne. La page contient enfin une division (repère ²) incluant une image (repère ¾), deux titres <h1> (repères ² et ¸) et un paragraphe (repère ¹). Pour ces derniers, la propriété display prend la valeur inline . Le contenu des titres est donc affiché comme du texte en ligne dans le paragraphe. Quant à l'image incluse dans , le gestionnaire d'événements onclick permet de la faire disparaître quand le visiteur clique sur la division en donnant à l'aide de code JavaScript la valeur none à la propriété display (repère ¶) ; en gérant l'événement ondblclick , un double-clic la fait réapparaître en lui octroyant la valeur inline (repère º).

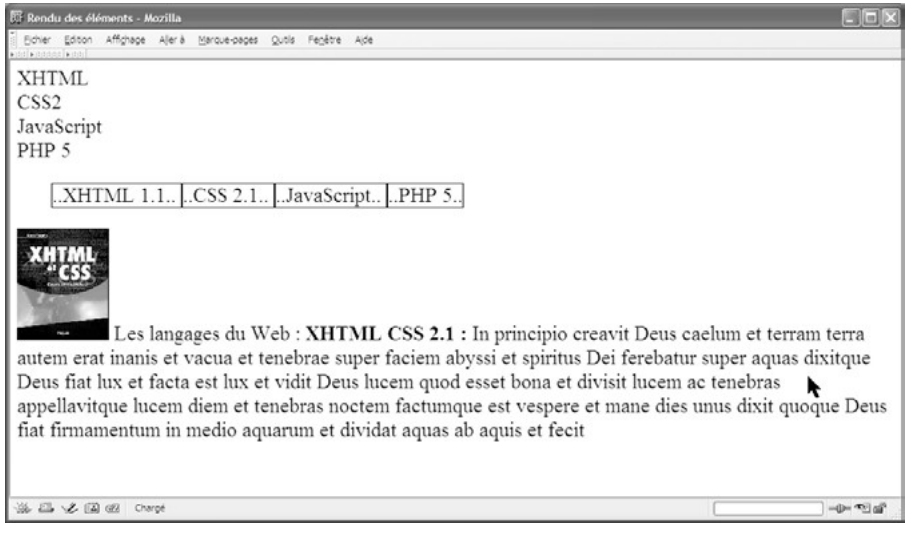
Exemple 13-3. Modification du rendu des éléments

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html  xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Rendu des éléments</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" >
   body,h1,p{font-size: 24px;}
   span{display: list-item; }      ³
   li{display: inline;border: solid 1px black;}          ·
   h1,p{display: inline;}     »
  </style>
 </head>
 <body>
   ¿  <p><span> XHTML </span> <span> CSS2 </span> <span> JavaScript </span>
   ➥ <span> PHP 5 </span>  </p>
   ´    <ul>
   <li><a  href="lien1.html" ></a>..XHTML 1.1.. </li>
   <li><a  href="lien2.html" ></a>..CSS 2.1.. </li>
   <li><a  href="lien3.html" ></a>..JavaScript.. </li>
   <li><a  href="lien4.html" ></a>..PHP 5.. </li>
  </ul>
   ²    <div id="division"
   ➥ ¶  onclick="document.getElementById('couv').style.display='none'"
   ➥ º    ondblclick="document.getElementById('couv').style.display='inline'" >
    ¾ <img id="couv"  src="couvxhtml_gd.jpg" alt="XHTML CSS" height="130"
     ➥ width="107" /> Les langages du Web :
```

```
    <h1>XHTML </h1>
    <h1> CSS 2.1 : </h1>
    <p> In principio creavit Deus caelum et terram terra autem erat inanis et vacua
    ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas dixitque
    ➥ Deus fiat lux et facta est lux et vidit Deus lucem quod esset bona et divisit
    ➥ lucem ac tenebras appellavitque lucem diem et tenebras noctem factumque est
    ➥ vespere et mane dies unus dixit quoque Deus fiat firmamentum in medio aquarum
    ➥ et dividat aquas ab aquis et fecit </p>
  </div>
 </body>
</html>
```

La figure 13-5 montre le résultat obtenu avant la disparition de l'image.

Figure 13-5
Les modifications du rendu normal des éléments



Le positionnement des éléments

Le principe du positionnement permet de définir l'emplacement des boîtes générées pour chaque élément XHTML présent dans le code de la page. Il existe plusieurs schémas de positionnement en CSS 2.1.

Le positionnement selon le flux normal : il inclut le positionnement par défaut opéré par les navigateurs sans définition de styles particuliers. Avec ce positionnement, les éléments <p> ou <h1> par exemple, sont associés à une boîte de bloc, et les éléments <img /> ou à une boîte en ligne. Il inclut également le positionnement relatif des éléments de type bloc ou en ligne.

Le positionnement flottant : dans ce cas, la boîte de l'élément est générée comme dans le flux normal. Elle conserve pour exemple les dimensions qui lui ont été attribuées, soit par

ses attributs, soit par les propriétés `width` et `height`. En revanche, cette boîte est déplacée afin d'être positionnée le plus haut et le plus à gauche possible dans le contenu.

Le positionnement absolu : avec ce type de positionnement, le bloc généré par l'élément devient complètement indépendant du flux normal. Sa position est en particulier sans aucun rapport avec son ordre d'apparition dans le code XHTML. Des propriétés particulières doivent être définies pour placer la boîte de l'élément par rapport à son contenu, qu'il s'agisse de `<body>` ou d'un autre élément.

Le positionnement fixe : dans ce cas, un élément occupe une position fixe dans la fenêtre du navigateur et ne défile pas avec le reste de la page.

Le flottement

Le concept du modèle de boîtes vu au chapitre 11 suppose que les blocs apparaissent dans la page les uns à la suite des autres, de haut en bas, selon leur ordre d'apparition dans le code XHTML. De même, un élément remplacé, comme une image incluse dans un paragraphe au milieu d'un texte par exemple, est affiché comme un sous-bloc en provoquant un retour à la ligne avant et après l'image. Si sa largeur est inférieure à celle du paragraphe, cela entraîne l'apparition d'une zone blanche à sa droite, car elle est alignée à gauche par défaut. Cela crée un effet évidemment disgracieux dans la page. Il en est de même pour un élément de bloc, par exemple un tableau dont la largeur est faible par rapport à celle de son conteneur.

Pour améliorer cet état de fait, nous pouvons rendre n'importe quel élément flottant. Quand un élément est déclaré flottant, le contenu des autres éléments voisins se dispose autour de lui comme l'eau s'écoule autour d'un rocher placé au milieu d'une rivière. L'espace laissé libre autour de cet élément est comblé, créant ainsi une meilleure occupation de la surface de la page.

Quand un élément est déclaré flottant, la boîte qui correspond à l'élément est déplacée vers la gauche ou la droite de son conteneur selon la valeur choisie pour le flottement. L'alignement vertical est tel que la boîte est déplacée sur le haut de la ligne qui contient l'élément, ou sur le bas du bloc qui le précède. On procède à la définition du flottement d'un élément au moyen de la propriété `float` qui peut s'appliquer à tous les éléments XHTML et dont la syntaxe est la suivante :

```
float:left|right|none|inherit
```

Les valeurs qu'elle peut prendre ont la signification suivante :

- `left` : la boîte de l'élément est déplacée en haut et à gauche de son conteneur ;
- `right` : la boîte de l'élément est déplacée en haut et à droite ;
- `none` : la boîte ne flotte pas (c'est la valeur par défaut) ;
- `inherit` : la boîte hérite du comportement de son élément parent (ce qui n'est pas le cas par défaut).

Pour qu'un élément soit flottant, il est nécessaire qu'il soit dimensionné explicitement avec la propriété `width`, ou implicitement par ses dimensions intrinsèques (celles d'une image par exemple), ou par ses attributs `width` et `height` dans le cas d'un élément `<img />` ou `<object>`. Dans le cas contraire, le navigateur risque de l'afficher avec une largeur minimale, si ce n'est nulle.

Si un élément flottant possède des marges, ces dernières ne fusionnent pas avec celles des éléments voisins, mais elles s'ajoutent à celles-ci. Quand plusieurs éléments sont flottants du même côté, celui qui apparaît en seconde position dans le code peut être amenée à butter sur le bord droit ou gauche du premier selon que la propriété `float` a respectivement la valeur `left` ou `right`.

Dans l'exemple 13-4, la page contient un élément `<div>` (repère ¿) qui inclut du texte brut puis une image (repère ´), à nouveau du texte puis un paragraphe (repère ²). La division a une couleur de fond, des marges de 15 pixels et un alignement justifié (repère ³). L'image est positionnée flottante à gauche et a des marges de 20 pixels. Le paragraphe est flottant à droite, dimensionné avec une largeur de 200 pixels, et est muni d'une bordure qui permet de bien le distinguer du reste du texte. La figure 13-6 montre le résultat obtenu. Nous y constatons que les éléments flottants n'apparaissent que relativement à l'ordre dans lequel ils sont écrits dans le code XHTML. Par exemple, l'image n'apparaît qu'une fois que la ligne dans laquelle l'élément `<img />` est situé a été complétée par du texte qui se situe après cet élément.

Exemple 13-4. Les éléments flottants

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les éléments flottants</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" >
   div{background-color: #EEE;margin: 15px;text-align: justify;}    ³
   img{float:left;margin:20px;}    ´
   p.droit{float: right;border:2px solid red;width: 200px;margin:10px;
    ➥ padding: 10px;}»
  </style>
 </head>
 <body>
  ¿ <div>XHTML ET CSS : In principio creavit Deus caelum et terram terra autem erat
    ➥ inanis et vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur
    ➥ super aquas dixitque Deus fiat lux et facta est lux et vidit Deus
```

```
´  <img src="couvxhtml_gd.jpg" alt="XHTML CSS" />
lucem quod esset bona et divisit lucem ac tenebras appellavitque lucem diem et
➥ tenebras noctem factumque est vespere et mane dies unus dixit quoque Deus fiat
➥ firmamentum . . .
²  <p class="droit">Le W3C <br />aquas quae erant sub firmamento ab his quae erant
➥ super firmamentum et factum est ita vocavitque Deus firmamentum caelum et
➥ factum est vespere et mane dies secundus dixit vero Deus congregentur aquae
➥ quae sub caelo sunt in locum unum et appareat  </p>
in firmamento caeli et inluminent terram et factum est ita fecitque Deus duo
➥ magna luminaria luminare maius ut praeesset diei et luminare minus . . .
 </div>
</body>
</html>
```

Figure 13-6

Le flottement des éléments

L'exemple suivant permet d'appréhender les subtilités du positionnement flottant dans des cas particuliers. L'élément `<div>` (repère ») inclus dans la page contient maintenant trois images incorporées au milieu d'un texte (repères ¿ , ´ et ²). Vient ensuite un paragraphe ne contenant que du texte (repère ¶). Les deux premières images sont flottantes à gauche (repère ³), et la troisième à droite en utilisant la classe `img.droit` (repère ·).

Seule la deuxième image est dimensionnée explicitement avec ses attributs `height` et `width`, les autres ayant les dimensions originales de l'image.

Exemple 13-5. Cas particuliers de flottement

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les éléments flottants</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" >
   div{background-color: #EEE;}
   img{float:left;margin:20px;}
   img.droit{float: right;}
   p{background-color: #DD2;}
  </style>
 </head>
 <body>
   <div>XHTML ET CSS : In principio creavit Deus caelum et terram terra autem erat
    inanis et vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur
    super aquas dixitque Deus fiat lux et facta est lux et vidit Deus
    <img src="couvxhtml_gd.jpg" alt="XHTML CSS" />
   lucem quod esset bona et divisit lucem ac tenebras appellavitque lucem diem et
    tenebras noctem factumque est vespere et mane dies unus dixit quoque Deus
    fiat . . .
    <img src="couvxhtml_gd.jpg" alt="XHTML CSS" height="130" width="107" />
   herbam virentem et adferentem semen iuxta genus suum lignumque faciens fructum et
    habens unumquodque sementem secundum speciem suam et vidit Deus . . .
    <img class="droit" src="couvxhtml_gd.jpg" alt="XHTML CSS" />ut praeesset nocti
    et stellas et posuit eas in firmamento caeli ut lucerent super terram et
    praeessent diei ac nocti et dividerent lucem ac tenebras et vidit Deus . . .
  </div>
  <p>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
   et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas dixitque
   Deus fiat lux et facta est lux et vidit Deus . . . </p>
 </body>
</html>
```

On peut le vérifier à la figure 13-7, la première image est positionnée comme dans l'exemple précédent, et la deuxième flottante à gauche vient se positionner le plus à gauche possible, selon les principes énoncés plus haut. Mais comme l'espace est occupé par la première, elle vient se placer contre le bord droit de cette dernière, et non plus sur le

bord gauche de son conteneur. Nous pouvons remarquer à cette occasion que les marges de chacune des images sont conservées et ne fusionnent pas. Le cas de la troisième image flottante à droite est le plus particulier. En effet, l'élément ne pouvant la contenir en entier, elle déborde et empiète sur le paragraphe qui suit et se comporte comme si elle était flottante dans ce paragraphe, le texte de ce dernier l'entourant.

Figure 13-7
Cas particuliers de flottement

Empêcher le flottement

Nous pouvons empêcher le type de comportement de la troisième image qui flotte sur le paragraphe <p> en appliquant pour ce dernier la propriété clear. Ce procédé n'est possible que pour les éléments de bloc, mais il permet d'éviter le flottement d'un élément sur la gauche ou la droite de celui pour lequel la propriété est définie. Ainsi, l'image a la place nécessaire à son affichage et le texte du paragraphe est repoussé vers le bas, laissant apparaître un espace vide. Cela peut constituer un choix de présentation dans certains cas, mais la solution précédente est en l'occurrence plus compacte. Voici la syntaxe de la propriété clear :

```
clear: none | left | right | both | inherit
```

Les valeurs qu'elle peut prendre ont pour signification :

- none : le flottement est autorisé ;
- left : le flottement d'un élément sur la gauche est interdit ;
- right : le flottement d'un élément sur la droite est interdit ;
- both : le flottement d'un élément est interdit à gauche et à droite.

En conservant le code XHTML de l'exemple 13-5 et en modifiant seulement les styles de la manière suivante (repère ³) :

```
<style type="text/css" >
 div{background-color: #EEE;}
 img{float:left;margin:20px;}
 img.droit{float: right;}
 p{background-color: #DD2;clear:right;} ³
</style>
```

dans laquelle la propriété clear est définie avec la valeur right pour l'élément <p> nous obtenons le résultat présenté à la figure 13-8 qui montre que la troisième image ne flotte plus sur le paragraphe.

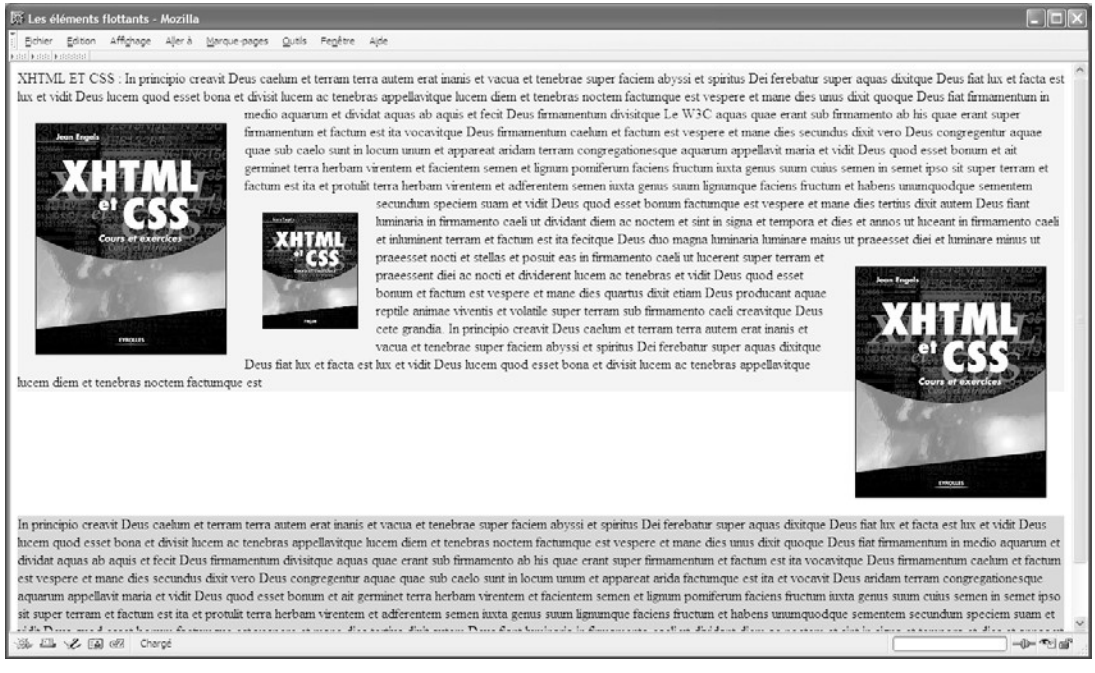


Figure 13-8

Suppression du flottement sur le côté d'un élément

Le positionnement relatif

Dans le positionnement relatif, les navigateurs analysent un document et déterminent un emplacement pour chacun des éléments comme dans le flux normal, puis déplacent ceux qui ont un positionnement relatif par rapport à la position qu'ils auraient dû occuper, mais en ne déplaçant pas les autres éléments. En conséquence, l'emplacement initial reste vide et il en résulte des chevauchements.

En positionnement relatif, l'ordre d'écriture des éléments est important car il conditionne l'emplacement de chaque élément (avant le déplacement relatif) et l'ordre de superposition en cas de chevauchement des boîtes ; la dernière écrite dans le code se superpose à celle de l'élément écrit avant.

Le positionnement relatif est spécifié au moyen de la propriété `position`, munie de la valeur `relative`. Prise isolément, cette propriété n'a aucun effet. Il faut ensuite préciser le décalage voulu au moyen des propriétés `left`, `top`, `right`, `bottom` dont les significations sont les suivantes :

- `left` : décale l'élément vers la droite (si sa valeur est positive) ou vers la gauche (si sa valeur est négative) ;
- `top` : décale l'élément vers le bas (si sa valeur est positive) ou vers le haut (si sa valeur est négative) ;
- `right` : décale l'élément vers la gauche (si sa valeur est positive) ou vers la droite (si sa valeur est négative) ;
- `bottom` : décale l'élément vers le haut (si sa valeur est positive) ou vers le bas (si sa valeur est négative).

Les quatre propriétés `left`, `top`, `right`, `bottom` ont la même syntaxe. Par exemple :

```
left: <longueur> | NN% | auto | inherit
```

Le paramètre `<longueur>` est donné comme à l'habitude par un nombre et une unité ; les pourcentages se réfèrent aux dimensions du bloc conteneur. Avec le mot-clé `auto`, la valeur de la propriété est calculée en fonction de la valeur de celle qui lui est complémentaire (les couples `left/right` et `top/bottom` sont complémentaires) de la manière suivante :

- Si les deux propriétés complémentaires ont la valeur `auto`, leur valeur calculée est `0`.
- Si l'une vaut `auto` et que l'autre a une valeur numérique explicite, la première prend la valeur opposée à celle de la seconde.
- Si les deux propriétés complémentaires sont définies avec des valeurs numériques (donc différentes de `auto`) et que ces deux valeurs ne sont pas opposées, la valeur qui l'emporte dépend du sens de lecture du texte définie par l'attribut `dir`. S'il vaut `ltr`, c'est la propriété `left` qui l'emporte, et `right` prend la valeur opposée ; sinon, quand il vaut `rtl`, c'est `right` qui l'emporte.

L'exemple 13-6 illustre le positionnement relatif. L'élément `<div>` (repère ´) inclut un paragraphe qui contient du texte brut, puis trois images (repères ¶, º et ¾) dans un ordre donné. Si aucun de ces éléments n'est positionné, nous obtenons le résultat habituel présenté à la figure 13-9. Si nous positionnons chacune de ces images en leur appliquant

respectivement les classes img.un img.deux et img.trois , nous obtenons les déplacements relatifs suivants :

- image 1 : déplacement de 30 pixels vers le bas (top vaut 30 px) et de 40 pixels vers la droite (left vaut 40 px) (repère ³) ;
- image 2 : déplacement de 50 pixels vers le haut (top a une valeur négative de – 50 px) et de 40 pixels vers la gauche (left a une valeur négative de -40 px) (repère ·) ;
- image 3 : déplacement de 80 pixels vers le haut (top a une valeur négative de – 80 px) et de 40 pixels vers la droite (left vaut 40 px) (repère »).

Le paragraphe est quant à lui déplacé de 20 pixels vers le bas (bottom a une valeur négative de – 20 px équivalente au style top:20 px) et de 6 % de la largeur de son conteneur (l'élément) vers la droite. Notons ici que les valeurs négatives de pourcentage ne fonctionnent pas.

Exemple 13-6. Le positionnement relatif

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html  xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Positionnement relatif</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" >
   div{background-color: #EEE;}
   img{margin:20px;}
   img.un{position:relative;top:30px;left:40px;}     ³
   img.deux{position:relative; top:-50px; left:-40px;}      ·
   img.trois{position:relative; top:-80px; left:40px;}        »
   p{background-color: #DD2;position:relative; bottom:-20px; left: 6% ;}    ¿
  </style>
 </head>
 <body>
    ´  <div>
    ²   <p><big>XHTML </big>: In principio creavit Deus caelum et terram terra autem
    ➥ erat inanis et vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur
    ➥ super aquas dixitque Deus fiat lux et facta est lux et . . . </p>
    ¶   <img src="couvxhtml_gd.jpg" alt="XHTML CSS" class="un"/>
    º   <img src="couvxhtml_gd.jpg" alt="XHTML CSS" height="130" width="107"
    ➥ class="deux"/>
    ¾ <img class="trois" src="couvxhtml_gd.jpg" alt="XHTML CSS" />
  </div>
 </body>
</html>
```

La figure 13-10 donne le résultat obtenu après le positionnement relatif. Nous constatons que les éléments déplacés relativement à leur position dans le flux normal peuvent se superposer aux autres éléments voisins, la deuxième image étant placée au dessus de la première et la troisième se superposant au texte du paragraphe.

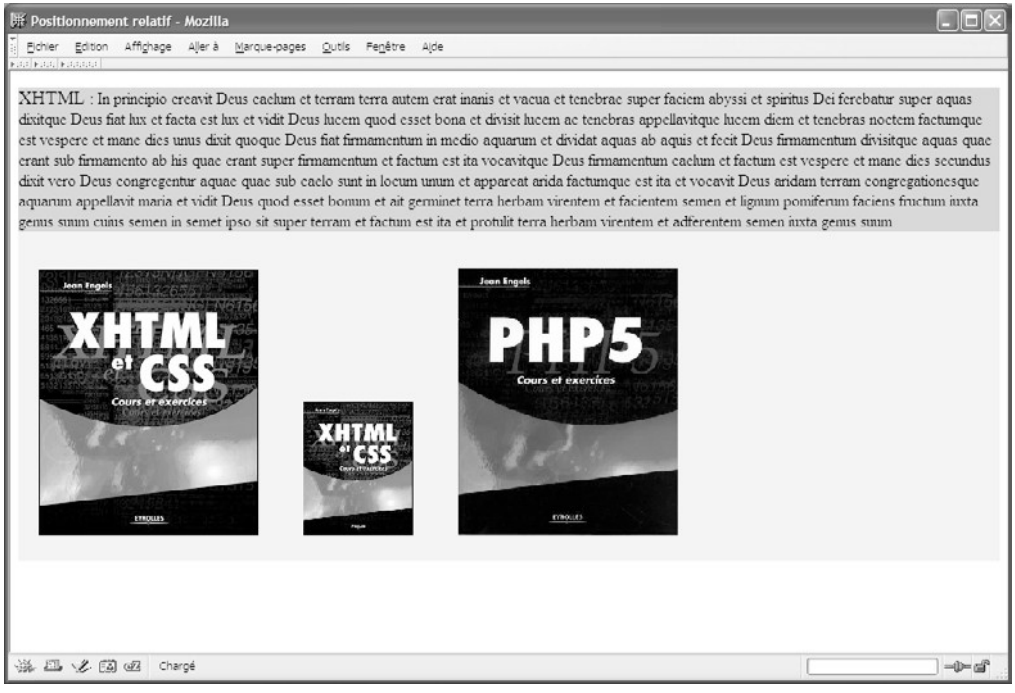


Figure 13-9

La page sans positionnement

Figure 13-10

La page avec positionnement

Le positionnement absolu

Dans le positionnement absolu, qui peut s'appliquer à tous les éléments, la boîte créée pour l'élément concerné n'apparaît plus dans le flux normal du document. Autrement dit, si un bloc `<div>` est positionné de manière absolue, il peut être écrit n'importe où dans le document XHTML sans que cela ne modifie la position qui va lui être assignée avec un style CSS. Le positionnement d'un élément est effectué par rapport au bloc de son conteneur (il s'agit souvent de l'élément `<body>` mais pas nécessairement). Chaque bloc d'un élément positionné de manière absolue devient à son tour le conteneur de ses éléments enfants. Si ces éléments sont eux-mêmes en position absolue, ils le sont par rapport à leur bloc parent direct et non par rapport au bloc qui contient leur parent. Nous pouvons donc créer des blocs `<div>` et les positionner, puis écrire normalement leur contenu.

L'indépendance de la position par rapport au flux normal fait que deux éléments en position absolue peuvent occuper la même zone dans leur bloc parent commun. Dans ce cas, aucun d'eux ne repousse l'autre comme il en va pour les éléments flottants. Le bloc d'un élément peut alors recouvrir l'autre en fonction de leur ordre d'empilement (voir la propriété `z-index`). Cette propriété peut alors être utilisée pour créer des effets dynamiques d'apparition et de disparition gérés par des scripts JavaScript.

Comme le positionnement relatif, le positionnement absolu est défini en utilisant encore la propriété `position`, mais en lui donnant cette fois la valeur `absolute` (ou `fixed` sur laquelle nous reviendrons par la suite). Il faut ensuite définir la position de l'élément par rapport à son conteneur à l'aide des propriétés `left`, `top`, `right`, `bottom` qui définissent la position des bords de l'élément respectivement par rapport aux bords gauche, haut, droit et bas du conteneur. Nous ne définissons généralement que deux de ces propriétés, le plus souvent `left` et `top` (les propriétés symétriques `right` et `bottom` prenant une valeur opposée), et la boîte de l'élément doit être dimensionnée avec les propriétés `width` et `height`. L'utilisation conjointe des propriétés `position` et `float` est impossible, et si c'est le cas la propriété `float` prend automatiquement la valeur `none`.

Dans les exemples suivants, nous allons étudier divers cas de mise en page correspondant à des types de présentation couramment rencontrés sur le Web.

Le premier cas présenté dans l'exemple 13-7 consiste à diviser la page en deux zones horizontales. La zone supérieure est un bandeau contenant le titre du site et un menu composé de liens vers les différentes pages. Pour permettre une navigation aisée, toutes les pages contiennent ce même bandeau dont il suffit de copier le code et les styles qui lui sont associés dans chacune des pages. Il est contenu dans un élément `<div>` muni d'un attribut `id` (repère ´) et il inclut un titre `<h1>` (repère ²) et le menu créé par une liste non ordonnée `<ul>` (repère ¶) dont chaque item contient un lien vers les différentes pages.

Ce premier élément `<div>` est positionné de manière absolue en haut et à gauche de la page en définissant les propriétés `top` et `left` avec la valeur `0`. Il est dimensionné à 100 % en largeur et à 110 pixels en hauteur (repère ³). La propriété `display` appliquée à l'élément `<li>` permet d'obtenir le menu des liens en ligne sous forme horizontale (repère »). Les items sont, de plus, munis de bordures pour en améliorer l'aspect.

La zone de contenu est également incluse dans un second élément qui contient un titre <h1>, un paragraphe <p> et une image (repère º), ces éléments n'étant pas positionnés. Comme la première, cette zone est dimensionnée à 100 % de la largeur de la page et positionnée de manière absolue en étant placée à 110 pixels du bord supérieur de la page (correspondant à la hauteur du bandeau) et collée à son bord gauche en mettant la propriété left à la valeur 0 (repère ·).

Exemple 13-7. Positionnement en deux blocs horizontaux

```
<head>
 <title> La référence des éléments  XHTML 1.1 </title>
 <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
 <style type="text/css">
  body{font-size: 18px;}
  h1{font-size: 2em;margin-top: 5px;}
  div#menu {position: absolute ; width:100%; height: 110px; left:0px; top:0;
    ➥ background-color:rgb(255,102,5);color:white; } ³
  div#corps { position: absolute ;width:100%; left:0; top:110px;background-color:
    ➥ #EEE;color:black;}·
  li {display:inline;  border: solid 1px white;padding: 0 10px 0 10px;}        »
  img {float:right;}      ¿
  a{text-decoration: none;color: white;}
 </style>
</head>
<body>
 <div id="menu">´
  <h1>   Référence XHTML 1.1 <⁄h1>
  <ul> ¶
   <li> <a  href="page1.html" tabindex="1" accesskey="A"
     ➥ title="Structure">Structure</a> </li>
   <li> <a  href="page2.html" tabindex="2" accesskey="B" title="Texte">Texte</a>
     ➥ </li>
  <!-- Suite de la liste -->
  </ul>
 </div>
 <div id="corps"> º
  <h1>La déclaration &lt;!DOCTYPE&gt;</h1>
  <img src="couvxhtml_gd.jpg" alt="XHTML CSS" />
  <p>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
    ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas dixitque
    ➥ Deus fiat lux et facta est lux et vidit Deus lucem quod esset bona et divis
    ➥ . . . </p>
 </div>
</body>
</html>
```

La figure 13-11 montre le résultat obtenu.

Figure 13-11

Le positionnement en deux blocs horizontaux

Nous pouvons également envisager une variante de ce premier cas qui va nous montrer que nous pouvons positionner des éléments de manière absolue à l'intérieur d'un élément lui-même positionné de cette façon. Nous réalisons cette opération à l'intérieur du second élément de l'exemple 13-7 en conservant intégralement son code XHTML.

Pour positionner les éléments enfants du second élément nous pouvons par exemple modifier simplement les styles CSS de la manière suivante :

```
<style type="text/css">
  body{font-size: 18px;}
  h1{font-size: 2em;margin-top: 5px;}
  div#menu {position: absolute ; width:100%; height: 110px; left:0px; top:0;
    ➥ background-color:rgb(255,102,5);color:white; }
  div#corps { position: absolute ;width:100%; left:0; top:110px;color:black;}
  li {display:inline;  border: solid 1px white;padding: 0 10px 0 10px;}
  div#corps h1{position: absolute; width:600px; height:40px;top: 20px;
    ➥ left: 300px;background-color: #AAA;margin: 0;}
  img {position: absolute; top: 70px;right:20px;}
  p {position: absolute;width: auto; top: 70px; right:260px; left:30px;margin: 0;
    ➥ text-align: justify;background-color: #BBB;}
```

```
    a{text-decoration: none;color: white;}
</style>
```

Le second élément `<div>` garde son positionnement par rapport à la page (repère ³), mais les éléments qu'il contient sont à leur tour positionnés. Ses éléments enfants `<h1>` `<img />` et `<p>` étant eux-mêmes positionnés de manière absolue, ils ne le sont pas par rapport à la page mais par rapport à leur parent. L'élément `<h1>` a une largeur de 600 pixels et une hauteur de 40 pixels ; il est placé à 300 pixels du bord gauche de son parent et à 20 pixels de son bord supérieur (soit 110 + 20 = 130 pixels du bord supérieur de la page) (repère ·).

L'image est placée à 70 pixels du bord haut de son conteneur et à 20 pixels de son bord droit (repère ·). Elle n'est pas dimensionnée explicitement et conserve ses dimensions intrinsèques qui sont celles du fichier image. Le paragraphe `<p>` n'est pas non plus dimensionné explicitement et sa largeur est fixée avec la valeur `auto`. Il sera ainsi redimensionné automatiquement sans empiéter sur l'image si la fenêtre est elle-même redimensionnée. En pratique, c'est son positionnement à 260 pixels du bord droit et à 30 pixels du bord gauche qui conditionne sa largeur.

Notons que si nous déplacions le conteneur de tous ces éléments en modifiant les propriétés `left` et `top` du second élément `<div>`, la position de ces trois éléments à l'intérieur de leur conteneur resterait inchangée. La figure 13-12 montre le résultat obtenu avec ces positionnements.

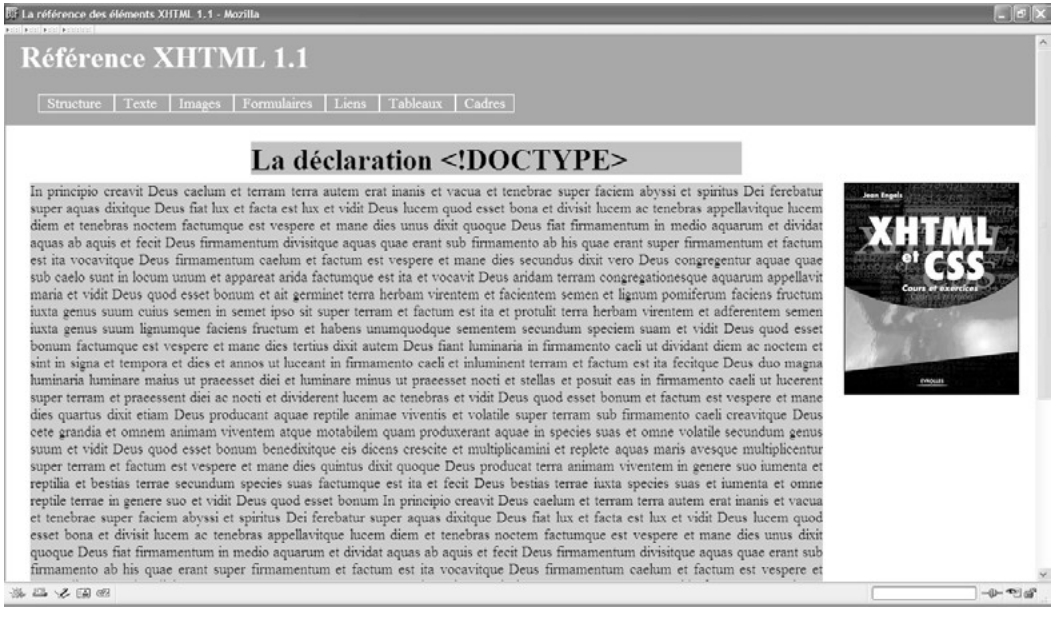


Figure 13-12

Positionnement dans un élément lui-même positionné

L'exemple suivant permet de créer un autre type de présentation très classique puisqu'il permet de diviser la page en trois zones. La première est un bandeau qui peut contenir le titre du site ou une publicité, l'espace restant étant divisé en deux colonnes de la même façon que nous l'avons réalisé avec des cadres au chapitre 8. Le contenu est identique à celui des exemples précédents mais il est structuré différemment. À chaque zone correspond un élément `<div>` qui va être positionné de manière absolue dans la page. Nous n'avons pas défini ici d'attributs `id` pour chacun d'eux, mais nous appliquons des classes différentes à chaque division.

Le premier élément `<div>` (repère ¿) qui contient un titre `<h1>` (repère ´) est positionné comme précédemment en haut de la page après avoir été dimensionné à 100 % en largeur et 100 pixels en hauteur. On évite ici de définir une hauteur en pourcentage pour qu'elle ne dépende pas du contenu de la page. Ces styles sont définis dans la classe `div.tete` (repère ³).

Le reste de la page est partagé en deux colonnes. La colonne de gauche (repère ?e) contient le même menu (repère º) que dans l'exemple 13-7, mais dans un affichage vertical, soit son style par défaut. Cet élément `<div>` est d'abord dimensionné avec une largeur de 20 % de celle de la page et une hauteur de 100 %. Il est positionné de manière absolue au moyen de la classe `div.menu` à 100 pixels du bord haut de la page et à 0 pixel de son bord gauche (repère ·).

Le contenu éditorial de la page constitue la colonne de droite créée avec le troisième élément `<div>` (repère ¾). Il contient un titre (repère µ), une image (repère ¸) et un paragraphe (repère ¹) comme dans les exemples précédents. Son traitement est assuré par la classe `div.contenu` dans laquelle il est dimensionné en largeur à 78 % et avec la valeur `auto` en hauteur. Son positionnement est absolu et il est placé à 20 % de la largeur de la page du bord gauche et à 100 pixels du bord supérieur (repère »). Avec leurs dimensions et positionnement respectifs, les trois zones sont bien collées les unes aux autres.

Exemple 13-8. Division de la page en trois zones

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <title> La référence des éléments  XHTML 1.1 </title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
   <style type="text/css">
   body{font-size: 18px;}
   div.tete {position: absolute ; left:0px;top:0px; width:100%; height:100px;
     ➥ background-color:rgb(0,0,153);margin:0; }
   div.menu {position: absolute ; width:20%; height: 100%; left:0px; top:100px;
     ➥ background-color:rgb(255,102,51);color:white; }
```

```
  div.contenu {margin: 1% ; position: absolute ;width:78%; left:20%;
    ➥ top:100px;background-color:white;color:black;}
  div h1 {font-size:50px;font-style:italic;color:rgb(255,102,51);
    ➥ margin-top:0px;margin-left:200px;}
  li {padding: 15px;}
  p{text-align: justify;}
  img{float: left;margin: 0 10px 0 0;}
 </style>
</head>
<body>
 <div class="tete">
  <h1>   Référence XHTML 1.1 </h1>
 </div>
 <div class="menu">
  <ul>
 <li> <a  href="page1.html" tabindex="1" accesskey="A"
  ➥ title="Structure">Structure</a> </li>
 <li> <a  href="page2.html" tabindex="2" accesskey="B" title="Texte">Texte</a>
  ➥ </li>
 <li> <a  href="page3.html" tabindex="3" accesskey="C" title="Images">Images</a>
  ➥ </li>
 <li> <a  href="page4.html" tabindex="4" accesskey="D"
  ➥ title="Formulaires">Formulaires</a> </li>
 <li> <a  href="page5.html" tabindex="5" accesskey="E" title="Liens">Liens</a>
  ➥ </li>
 <li> <a  href="page6.html" tabindex="6" accesskey="G" title="Tableaux">Tableaux
  ➥ </a> </li>
 <li> <a  href="page7.html" tabindex="7" accesskey="H" title="Cadres">Cadres</a>
  ➥ </li>
  </ul>
 </div>
 <div class="contenu">
  <h1>La déclaration &lt;!DOCTYPE&gt;</h1>
  <p>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
    ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas dixitque
    ➥ Deus fiat lux et facta est lux et vidit Deus lucem quod esset bona et divisit
    ➥ lucem . . .</p>
 </div>
</body>
</html>
```

La figure 13-13 montre le résultat obtenu avec ces définitions de styles. Nous pouvons remarquer une fois de plus l'intérêt des styles CSS car la présentation n'est pas la même que celle de la figure 13-12 alors que le contenu est le même.

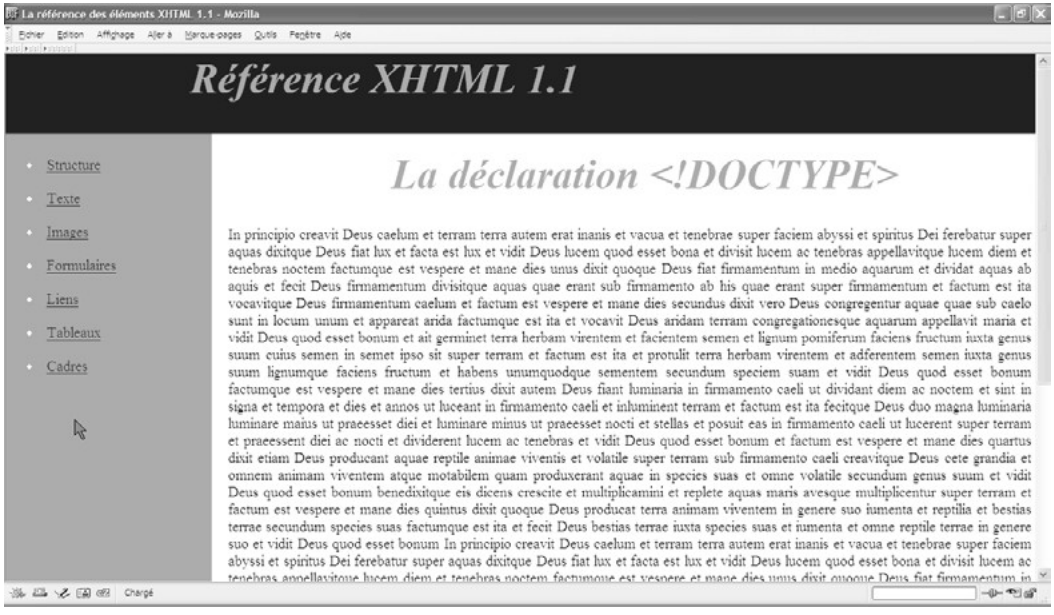


Figure 13-13

Positionnement en trois zones

Notre dernier exemple constitue une structure de page complexe comportant un bandeau, trois colonnes, la première contenant le menu, la deuxième un contenu textuel et un pied de page qui inclut l'adresse de contact. La troisième colonne contient une liste de liens utiles. La figure 13-14 montre la présentation de la page que nous allons obtenir.

Chacune de ces zones est créée par un élément `<div>` dimensionné puis positionné de manière absolue en leur appliquant les classes `div.haut`, `div.gauche` `div.droit`, `div.contenu` et `div.bas`. La seule division qui représente un élément nouveau par rapport aux cas antérieurs est celle qui permet de placer la division d'adresse en bas de la zone centrale de contenu. Cette division est incluse dans la division précédente et non plus directement dans `<body>`

Dans le code de l'exemple 13-9, les éléments `<div>` sont volontairement placés dans le désordre afin de démontrer que leur position dans le code XHTML n'a aucune importance. Les définitions et les rôles des classes sont les suivants :

- `div.haut` : (repères ³ et ¹) largeur 100 %, hauteur 70 pixels, positionné en haut et à gauche (`top:0` et `left:0`).

- div.gauche : (repères · et ¸) largeur 15 %, hauteur 100 % (ou auto), positionné à 70 pixels du haut et à 0 du bord gauche.
- div.droit : (repères» et ¶) largeur 15 %, hauteur 100 % (ou auto), positionné à 70 pixels du haut et à 0 du bord droit.
- div.contenu : (repères ¿ et °) largeur 70 %, hauteur auto car la hauteur du paragraphe peut varier si on réduit la fenêtre du navigateur.
- div.bas : (repères ´ et µ) elle s'applique à la division incluse dans le contenu. Sa largeur est de 100 % de celle de son parent (soit 70 % de celle de la page) et sa hauteur de 60 pixels. Elle est positionnée à gauche et en bas de son parent.

Afin que son contenu ne cache pas la fin du texte, on notera qu'il faut que le paragraphe (repère ¾) ait une marge basse d'au moins 60 pixels (repère ²).

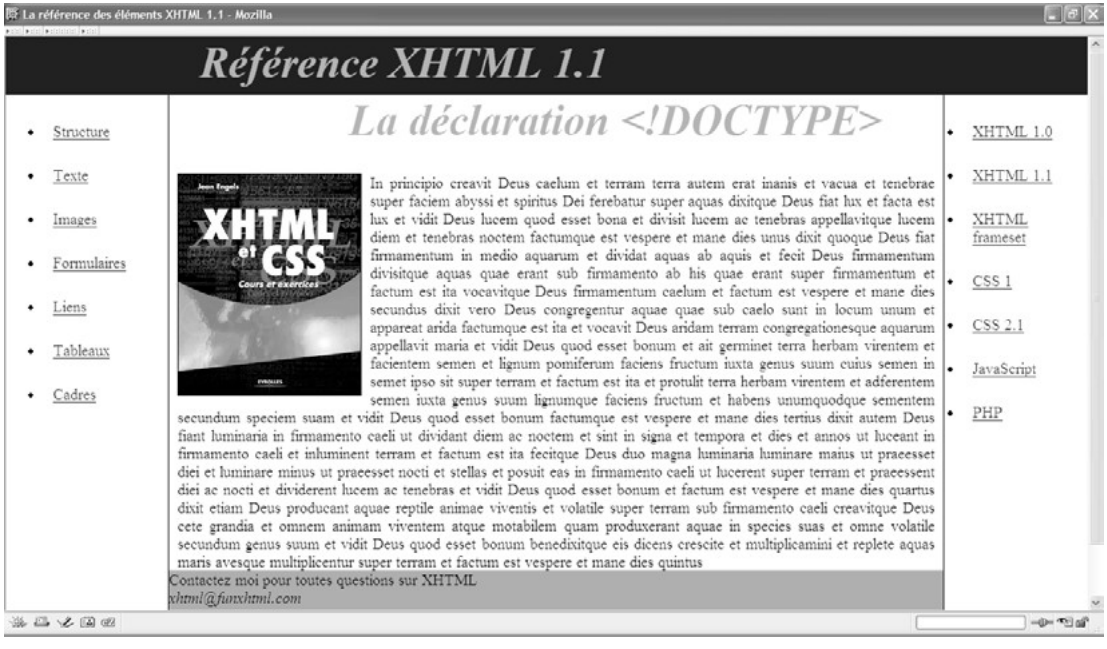


Figure 13-14

Organisation d'une page complexe en cinq zones

Exemple 13-9. Création d'une structure complexe en positionnement absolu

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
```

```
<head>
 <title> La référence des éléments  XHTML 1.1 </title>
 <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
 <style type="text/css">
  body{font-size: 18px;background-color:white;color:black;}
  div.haut {position: absolute ; left:0px;top:0px; width:100%; height:70px;
    ➥ background-color:rgb(0,0,153);margin:0; }
  div.gauche {position: absolute ; width:15%; height: auto; left:0px; top:70px;}
  div.droit{position: absolute ; width:15%; height: 100%; right:0; top:70px; }
  div.contenu {position: absolute ;width:70%; height:auto;left:15%;
    ➥ top:70px;padding: 0 10px 0 10px;border-left: 1px solid black;
    ➥ border-right: 1px solid black; }
  div.bas{position: absolute ; left:0px; bottom:0px; width:100%;
    ➥  height:60px;background-color:rgb(0,220,153);}
  div h1 {font-size:50px;font-style:italic;color:rgb(255,102,151);
    ➥ margin-top:0px;margin-left:200px;}
  li {padding: 15px;}
  p{text-align: justify;margin-bottom:60px;}
  img{float: left;margin: 0 10px 0 0;}
 </style>
</head>
<body>
 ¶  <div class="droit">
  <ul>
  <li> <a  href="xhtml10.html" tabindex="1" accesskey="A" title="Structure">XHTML
    ➥ 1.0</a> </li>
  <li> <a  href="page2.html" tabindex="2" accesskey="B" title="Texte">XHTML 1.1
    ➥ </a> </li>
  <li> <a  href="page3.html" tabindex="3" accesskey="C" title="Images">XHTML
    ➥ frameset</a> </li>
  <li> <a  href="page4.html" tabindex="4" accesskey="D" title="Formulaires">CSS
    ➥ 1</a> </li>
  <!—Suite de la liste -->
 </ul>
</div>
  <div class="contenu">
 <h1>La déclaration &lt;!DOCTYPE&gt;</h1>
 <img src="couvxhtml_gd.jpg" alt="XHTML CSS" />
  ¾ <p>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
    ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas dixitque
    ➥ Deus fiat lux et facta est lux et vidit Deus lucem quod esset . . .</p>
 µ  <div class="bas">
 Contactez-moi pour toutes questions sur XHTML <address>xhtml@funxhtml.com
    ➥ </address>
```

```
</div>
</div>
 ¸ <div class="gauche">
<ul>
<li> <a  href="page1.html" tabindex="1" accesskey="A"
➥ title="Structure">Structure</a> </li>
<li> <a  href="page2.html" tabindex="2" accesskey="B" title="Texte">Texte</a>
➥ </li>
<li> <a  href="page3.html" tabindex="3" accesskey="C" title="Images">Images</a>
➥ </li>
<!—Suite de la liste -->
</ul>
</div>
 ¹ <div class="haut">
<h1>   Référence XHTML 1.1 </h1>
</div>
</body>
</html>
```

Le positionnement fixe

Au chapitre 8, nous avons envisagé la manière de créer des cadres pour diviser la page en zones qui demeurent fixes dans la fenêtre du navigateur. Outre que cette méthode, considérée aujourd'hui comme obsolète, présente plusieurs inconvénients déjà abordés, elle nécessite l'emploi de la DTD XHTML 1.0 frameset. Comme nous l'avions signalé au chapitre 8, en utilisant des styles CSS appropriés, il est possible de créer une présentation quasi similaire.

Le positionnement fixe n'est qu'un cas particulier du positionnement absolu. La détermination des dimensions et de la position d'un élément se réalisent de la même façon à l'aide des propriétés `width` et `height` pour les dimensions, et `top`, `left` , `right` et `bottom` pour déterminer la position. La seule différence réside dans le fait que le bloc conteneur utilisé n'est plus forcément l'élément parent mais la fenêtre du navigateur. De cette façon, un élément positionné de manière fixe reste à sa place même en cas de défilement de la page quand la hauteur de celle-ci dépasse celle de la fenêtre du navigateur. En d'autres termes, les propriétés de positionnement sont appliquées non plus par rapport au conteneur mais par rapport aux bords de la fenêtre.

Ce type de positionnement peut être réalisé en utilisant encore la propriété `position` en lui donnant la valeur `fixed`. Elle est bien gérée par les navigateurs actuels comme Mozilla, Firefox, Netscape, ce qui n'a pas toujours été le cas, mais très incomplètement par Internet Explorer.

Si nous reprenons exactement le même code XHTML que celui de l'exemple 13-9, nous pouvons réaliser un positionnement fixe des zones haute, gauche, droite et basse qui sont

des parties de navigation qu'il peut être utile de voir en permanence. Pour ce faire, nous écrivons les styles suivants :

Exemple 13-10. Le positionnement fixe

```
<style type="text/css">
  body{font-size: 18px;background-color:white;color:black;}
  div.haut {position:  fixed ³  ; left:0px;top:0px; width:100%; height:10%;
    ➥ background-color:rgb(0,0,153);margin:0; }
  div.gauche {position:  fixed ·  ; width:15%; height: auto; left:0px; top:10%;}
  div.droit{position:  fixed »  ; width:15%; height: 100%; right:0; top:10%; }
  div.contenu {position:  absolute¿  ;width:70%; height:auto;left:15%;
    ➥ top:10%;padding: 0 10px 0 10px;border-left: 1px solid black;
    ➥ border-right: 1px solid black; }
  div.bas{position:  fixed ´  ; left:0px; bottom:0px; width:100%;
    ➥ height:60px;background-color:rgb(0,220,153);}
  div h1 {font-size:50px;font-style:italic;color:rgb(255,102,151);
    ➥ margin-top:0px;margin-left:200px;}
  li {padding: 15px;}
  p{text-align: justify;margin-bottom:60px;}
  img{float: left;margin: 0 10px 0 0;}
 </style>
```

Les seules modifications effectuées consistent à remplacer le mot-clé `absolute` par `fixed` dans toutes les classes sauf celle du contenu qui doit pouvoir défiler pour être lisible. La figure 13-15 montre le résultat obtenu dans une fenêtre redimensionnée et après avoir effectué un défilement vertical. Nous y constatons que seule la zone centrale a défilé alors que les autres n'ont pas bougé et restent entièrement visibles. Nous pouvons remarquer également que la zone basse est maintenant bien positionnée par rapport à la fenêtre car elle occupe toute la largeur de l'écran contrairement au cas précédent de positionnement absolu.

Le positionnement fixe représente donc une alternative crédible à la création de cadres comme nous l'avons fait au chapitre 8. Nous parlerons plutôt de simili-cadres qui donnent l'illusion d'une page avec cadres et en présente les avantages visuels, mais sans interactivité entre les différentes zones comme c'est le cas des cadres. La procédure à suivre, relativement simple, est la suivante :

- Créer autant d'éléments `<div>` que l'on désire obtenir de zones différentes dans la page. Chaque zone est l'équivalent des éléments `<frame />` utilisés dans la méthode avec cadres. Pour chacune de ces zones, définir la propriété `position` avec la valeur `fixed`.
- Dimensionner chacune de ces divisions en utilisant les propriétés `width` (largeur) et `heigth` (hauteur).
- Positionner les éléments `<div>` en définissant les propriétés `left`, `top`, `right` et `bottom` qui nous permettent de placer les éléments `<div>` par rapport aux bords de la fenêtre.
- Définir éventuellement les propriétés de couleur de fond, de bordure, de marge et d'espacement pour améliorer la présentation du contenu de chaque élément `<div>`

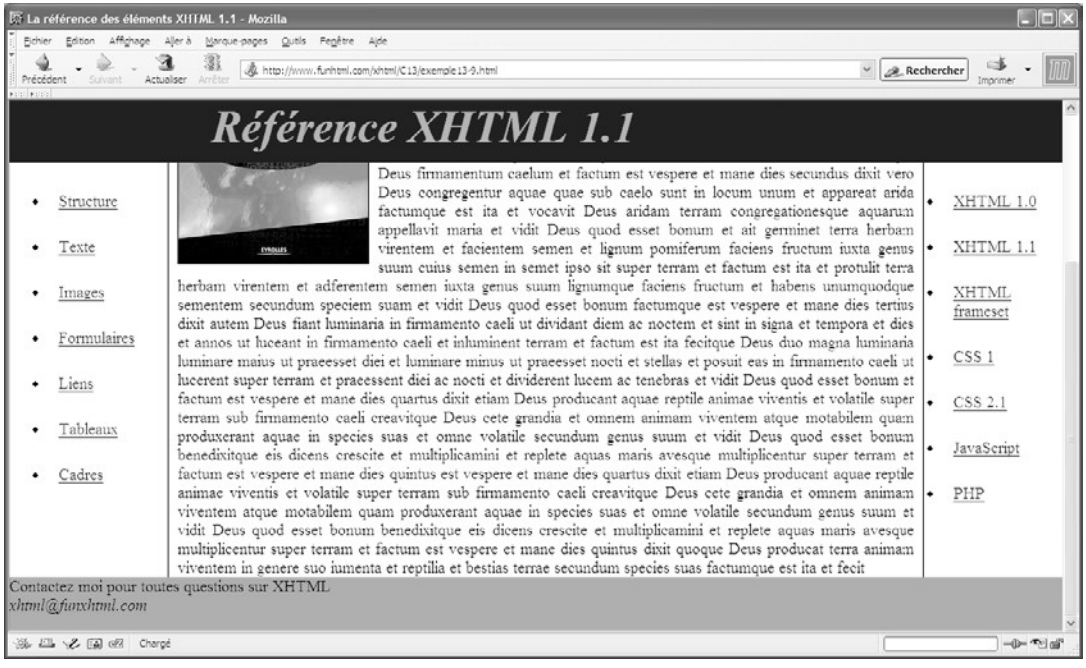


Figure 13-15
Le positionnement fixe

Pour retrouver une interactivité entre les « simili-cadres », par exemple afin que le clic sur un lien du menu affiche un contenu adapté dans la zone centrale, nous pouvons créer plusieurs pages qui ont toutes en commun les divisions positionnées de manière fixe et dont le contenu de la zone centrale est variable. L'illusion d'un site avec cadres est alors complète.

Visibilité et ordre d'empilement

Nous avons pu constater qu'en positionnant des éléments de manière relative, absolue ou fixe, il était possible que plusieurs éléments occupent partiellement ou totalement le même espace dans la fenêtre du navigateur. Dans ce cas, le dernier apparu dans l'ordre du code XHTML se superpose au précédent. Pour pouvoir intervenir sur cet état de fait et gérer volontairement ces superpositions, CSS définit un placement des éléments selon trois dimensions, les deux premières dans le plan de l'écran sur lesquelles nous pouvons intervenir avec les propriétés `left`, `top`, `right` et `bottom` comme nous l'avons déjà vu, et la troisième dimension est définie selon un « axe des z » perpendiculaire à l'écran et dirigé vers le spectateur. Nous pouvons gérer cet ordre d'empilement au moyen de la propriété `z-index` dont la syntaxe est la suivante :

- `z-index : auto | <Nombre> | inherit`

Elle ne s'applique qu'aux éléments positionnés, et les valeurs qu'elle peut prendre sont les suivantes :

- auto : l'élément a la même valeur que celle de son parent direct, qu'il soit défini explicitement par un nombre ou implicitement par son ordre d'apparition dans le code XHTML (le dernier ayant la priorité).

- <Nombre> un nombre entier positif ou négatif, sachant que plus le nombre est grand, plus l'élément est placé en avant, et se superpose à ceux dont la valeur est inférieure.

Nous pouvons intervenir dynamiquement sur l'ordre d'empilement au moyen de code JavaScript en modifiant la valeur de la propriété CSS z-index (qui en JavaScript porte le nom de zIndex) en réponse à une action du visiteur (survol de l'élément par la souris, clic…). Cette modification est gérée par les attributs gestionnaires d'événements correspondants (onmouseover ou onclick…) ou par la pseudo-classe :hover par exemple.

Dans le même ordre d'idées, nous pouvons appliquer à tous les éléments la propriété visibility qui permet de les cacher ou de les rendre visibles. Sa syntaxe est la suivante :

```
visibility : visible | hidden | collapse | inherit
```

- visible : l'élément est visible normalement et c'est la valeur par défaut.

- hidden : l'élément est caché mais la différence de comportement avec la propriété display, quand elle prend la valeur none c'est que la boîte de l'élément est ici maintenue dans la page mais qu'elle est simplement vide et non retirée de la page comme avec display.

- collapse : son comportement est identique à la valeur hidden mais elle s'applique particulièrement aux cellules des tableaux.

Cette propriété est héritée par défaut et elle s'applique donc aux éléments enfants.

L'exemple 13-11 donne une illustration de la gestion de l'ordre d'empilement et de la visibilité de plusieurs éléments. La page comporte une division (repère ²) qui contient un paragraphe <p> positionné (repères ´ et ¶), lui-même incluant du texte et une image flottante (repères º et »). La division inclut également deux images (repères ¾ et µ) qui y sont positionnées de manière absolue (repères ³ et ·). La figure 13-16 montre le résultat obtenu initialement lors de l'affichage de la page.

Compte tenu de l'ordre d'apparition des éléments enfants de la division , l'ordre d'empilement sans utilisation de la propriété z-index devrait être de l'arrière vers l'avant, le paragraphe, l'image « un » puis l'image « deux » au premier plan. Cependant, comme nous attribuons à la propriété z-index de la première image la valeur 2 et à la suivante la valeur 1, cet ordre est inversé.

Figure 13-16

L'état initial de la page avec des éléments empilés

Si la propriété `z-index` ne servait qu'à définir un ordre d'empilement fixe, elle n'aurait qu'un intérêt limité car il suffirait de placer en dernier dans le code XHTML l'élément que l'on veut voir se positionner au premier plan. En gérant l'événement `onclick` par exemple pour le paragraphe ou chacune des images avec le code JavaScript suivant :

```
onclick="this.style.zIndex++"
```

nous permettons au visiteur de placer au premier plan l'élément qu'il désire et éventuellement d'inverser cet ordre à chaque nouveau clic. Notons de nouveau qu'en JavaScript le nom de la propriété devient `zIndex` et que l'opérateur `++` signifie simplement qu'il faut augmenter la valeur de la propriété de 1 unité. De même le texte du paragraphe, recouvert partiellement par l'image « un », peut être mis au premier plan pour être entièrement lisible.

La visibilité de l'image incluse dans le paragraphe (repère °) est contrôlée par la pseudo-classe `:hover`. Si le visiteur positionne le curseur sur cette image (en le laissant immobile, sinon on obtient un effet de clignotement), elle devient invisible en laissant l'espace qu'elle occupait vide dans le paragraphe. Cet effet est obtenu en donnant à la propriété `visibility` la valeur `hidden` en cas de survol (il est annulé automatiquement quand le curseur quitte la zone de l'image).

Exemple 13-11. Visibilité et empilement

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Visibilité et ordre d'empilement</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" >
   body{font-size:18px;}
   img.un{position:absolute;top:160px;left:350px;}
   img.deux{position:absolute;top:260px;left:450px;}
   img#couv{float: left;}
   img#couv:hover{visibility:hidden;}
   p{background-color: #EEE; position:absolute; top:40px;  left:50px;}
  </style>
 </head>
 <body>
    <div>
     <p onclick="this.style.zIndex++"><big>XHTML </big>:
     <img id="couv" src="couvxhtml_gd.jpg" alt="XHTML CSS" height="130"
     ➥ width="107" />
    In principio creavit Deus caelum et terram terra autem erat inanis et vacua et
      ➥ tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas dixitque
      ➥ Deus fiat lux et facta est lux et vidit Deus lucem quod esset bona et divisit
      ➥ lucem ac tenebras appellavitque lucem diem et tenebras noctem factumque est
      ➥ vespere et mane dies unus dixit </p>
     <img src="couvxhtml_gd.jpg" alt="XHTML CSS" height="285" width="240" class="un"
     ➥onclick="this.style.zIndex++" />
     <img src="couvphp5gd.png" alt="XHTML CSS" height="285" width="240" class="deux"
     ➥onclick="this.style.zIndex++" />
 </div>
 </body>
</html>
```

La figure 13-17 donne le résultat obtenu après avoir successivement mis le paragraphe devant la première image (en cliquant sur le paragraphe), la seconde image devant la première (en cliquant sur la seconde), puis en plaçant le curseur sur l'image incluse dans le paragraphe. Par une série de nouveaux clics, il est possible de revenir en arrière.

Figure 13-17

L'état de la page après la modification de l'empilement et de la visibilité de certains éléments

Exercices

Exercice 1 : Incorporez trois images dans une page en définissant pour l'élément `<img />` les propriétés CSS `width` et/ou `height` (sans utiliser les attributs de même nom). Que se passe-t-il si les dimensions intrinsèques des images dépassent ces valeurs ?

Exercice 2 : Dans l'exercice précédent, comment procéder au dimensionnement CSS de façon que chaque image occupe la place qui lui est nécessaire d'après les dimensions du fichier image ?

Exercice 3 : Si les dimensions d'une image sont définies en pourcentage de celles de son conteneur, et que ses proportions largeur/hauteur sont inconnues, comment faire pour qu'elle ne soit pas déformée ?

Exercice 4 : Incluez trois éléments `<div>` contenant du texte, l'un dans l'autre, et définissez la largeur à 70 % du précédent pour chacun d'eux. Le premier doit avoir 800 pixels de haut et les suivants doivent correspondre à 80 % de la hauteur du précédent.

Exercice 5 : Créez deux paragraphes d'une hauteur de 300 pixels et d'une largeur de 700 pixels. Incluez-y un texte très long et gérez le débordement du texte afin qu'il soit entièrement lisible.

Exercice 6 : Créez cinq titres de niveau 2 et affichez les sous-formes de liste (indice : voir la propriété `display`).

Exercice 7 : Dans un élément `<div>`, incluez un élément `<span>` contenant du texte et donnez-lui le style bloc.

Exercice 8 : Créez un menu vertical dont les éléments sont des liens `<a>`

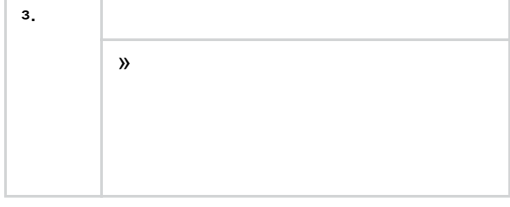
Exercice 9 : Créez une page contenant un paragraphe incluant du texte et deux éléments `<img />` qui se suivent. Faites flotter les images, la première à gauche et la seconde à droite.

Exercice 10 : Reprenez l'exercice précédent et empêchez le flottement de la deuxième image.

Exercice 11 : Placez trois images de tailles initiales différentes dans une page. Écrivez les styles pour qu'elles s'affichent avec la même taille. Ensuite, positionnez-les afin d'obtenir un effet de cascade avec un décalage horizontal et vertical constant pour chaque image par rapport à la précédente. L'utilisateur doit pouvoir mettre chacune d'elles au premier plan en cliquant dessus (voir la propriété `z-index`).

Exercice 12 : Créez une mise en page à trois colonnes de largeurs respectives de 20 %, 65 % et 15 %. La première et la troisième doivent contenir respectivement un menu et une liste de liens créés à partir d'images. La colonne centrale doit avoir un contenu éditorial.

Exercice 13 : Créez une mise en page selon le modèle de la figure ci-dessous :



La colonne de gauche (repère ³) a une largeur de 200 pixels et le bandeau (repère ·) une hauteur de 150 pixels. Le bandeau contient le titre du site, la colonne de gauche un menu et la zone principale (repère ») du texte et des images au choix. Le premier lien du menu doit afficher une page ayant la même structure, le même contenu dans les zones ³ et · , mais un contenu éditorial différent dans la zone » .

Exercice 14 : Reprenez l'exemple précédent de façon à ce que les zones ³ et · soient fixes dans la fenêtre du navigateur.

14

Le style des tableaux

Nous avons constaté dans les chapitres précédents que les tableaux peuvent avoir des applications très variées, qu'il s'agisse d'organiser des données de manière structurée, de présenter un formulaire ou encore, malgré les réserves indiquées, d'ordonner toute une page. Cela dit, les styles par défaut appliqués aux tableaux et à leur contenu dans les éléments `<td>` ou `<th>` ont un rendu particulièrement pauvre et austère du point de vue esthétique. Nous allons voir maintenant qu'un certain nombre de propriétés, déjà étudiées par ailleurs, peuvent s'appliquer aux tableaux et à leurs composants, comme les bordures, les couleurs et images de fond. De plus, le modèle CSS de gestion des tableaux possède des caractéristiques spécifiques qui permettent de gérer les dimensions des cellules, des largeurs de tableaux entiers ou des colonnes qui les composent. De nouvelles propriétés fournissent également des outils pour gérer les tableaux et les cellules vides des tableaux symétriques ou irréguliers.

Le modèle de gestion des tableaux

Les tableaux permettent principalement de créer un modèle de gestion des données. En structurant un tableau, on peut organiser ces informations en lignes et en colonnes, de façon qu'elles entretiennent des relations entre elles. C'est ce type de relations que nous avons créées dans les tableaux statistiques du chapitre 6. Les différents éléments abordés nous permettent de structurer un tableau en groupes de lignes ou de colonnes. Avec certains attributs des éléments XHTML de création des tableaux, il est aussi possible de mettre en exergue la structure définie par le concepteur. Cependant, leurs effets sont relativement limités, et l'application de styles CSS à ces éléments donne les moyens de diversifier considérablement le rendu final des tableaux.

Le modèle CSS de gestion des tableaux se conforme à celui défini dans XHTML, ce qui établit une cohérence entre ces deux systèmes complémentaires. Mis à part le titre du tableau créé avec l'élément `<caption>` et situé en dehors du quadrillage visible ou non du tableau, tout le contenu est inclus prioritairement dans des lignes. Dans le modèle XHTML et CSS, nous ne pouvons pas créer de colonnes indépendamment des lignes, autrement dit le modèle donne la priorité aux lignes sur les colonnes. Une colonne n'a d'existence que parce qu'elle contient des cellules qui appartiennent à une suite de lignes créées antérieurement. Une des premières conséquences de cet état de fait est que si nous définissons un style pour un élément `<tr>` particulier, en écrivant une classe ou un sélecteur d'attribut `id` par exemple, ce style va s'appliquer à toutes les cellules de la ligne en question et pas aux cellules de la ligne suivante.

Les couleurs des cellules

Les différents éléments XHTML propres aux tableaux s'inscrivent dans une hiérarchie bien précise, l'élément parent de tous les autres étant `<table>` et ceux de plus bas niveau `<td>` et `<th>` Quand nous voulons appliquer une couleur de fond à un élément de tableau (tableau dans son intégralité, groupe de lignes ou de colonnes, ligne, colonne ou cellule), il peut survenir des conflits dans la définition de ces propriétés. Pour régler ces conflits, le modèle CSS de gestion des tableaux définit un ordre de priorité pour chacun de ces éléments. Dans ce modèle, un tableau est considéré comme un empilement de couches, de façon similaire à ce que l'on trouve dans les logiciels de création d'images comme Photoshop, la couleur de la couche supérieure masquant celle de la couche inférieure, sauf dans les zones transparentes. Ce principe est également identique à celui que nous avons utilisé avec la propriété `z-index`, mais nous nous ne pouvons pas ici modifier cet ordre prédéterminé.

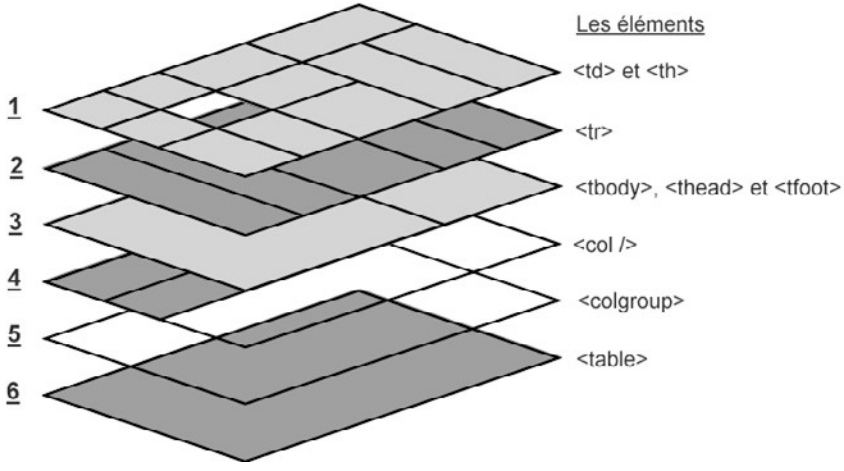
La liste suivante présente cet ordre de priorité, de la couche la plus haute à la plus basse. Pour bien gérer les couleurs de fond de ces éléments, il faut bien noter que, pour chacun d'entre eux, si aucune valeur n'est attribuée à la propriété `background-color` cette dernière a la valeur `transparent`, ce qui implique que l'élément concerné laisse voir la couleur définie pour son parent situé à la couche inférieure. De plus, l'ordre de priorité l'emporte toujours sur celui de la définition des styles.

1. Les cellules créées avec les éléments `<td>` et `<th>` ont la plus haute priorité. La couleur de chaque cellule peut donc être définie individuellement en utilisant un sélecteur d'attribut `id` ou une classe. Les cellules vides déclarées, cachées avec la propriété `empty-cells` (voir plus loin), sont toujours transparentes.
2. Les lignes créées avec `<tr>` viennent ensuite.
3. Les groupes de lignes créés avec `<tbody>` `<thead>` et `<tfoot>` constituent la troisième couche.
4. Les définitions de colonnes appliquées dans l'élément `<col />` figurent dans la couche suivante.
5. Les groupes de colonnes créés dans l'élément `<colgroup>` sont dans la cinquième couche.

6. La couleur de fond définie pour l'élément `<table>` parent de tous les précédents a la plus basse des priorités.

La figure 14-1 résume cet ordre de priorité des couches des éléments de tableau.

Figure 14-1
L'empilement des couches pour les éléments de tableau



Dans l'exemple de la figure 14-1, la couleur de fond pour l'élément `<table>` est gris clair (`#EEE` repère ³). L'en-tête `<thead>` (repère ¶) et le pied de tableau `<tfoot>` (repère º), dont les couleurs de fond sont un gris plus foncé (`#DDD` repère ·), vont l'emporter sur la couleur de l'élément `<table>` car ils sont sur une couche supérieure. En revanche, les lignes de données contenues dans l'élément `<tbody>` (repère ¾), pour lequel aucune couleur n'est définie explicitement, ont la couleur de fond de l'élément `<table>` La couleur de fond de l'élément `<col />` (repères ¿ et ²), qui s'applique aux deux premières colonnes, s'impose également à celle de `<table>` pour la même raison et aussi parce qu'aucune couleur n'est définie pour `<tbody>` Cependant, comme la couleur de fond définie dans la classe `.fonce` (repère ») est appliquée au troisième élément `<tr>` (repère ¸), celle-ci s'impose au-dessus de celle des éléments `<table>` et `<col />` dont les ordres de priorité sont inférieurs.

Pour personnaliser et mettre en évidence la cellule particulière qui contient la variation maximale, nous créons un style avec un sélecteur d'id (repère ´) qui sera appliqué au seul élément `<td>` dont l'attribut `id` vaut `"max"` (repère µ).

Exemple 14-1. L'empilement des couleurs de fond des tableaux

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
➥ "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Couleur de fond des tableaux</title>
```

```
 <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
 <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
 <style type="text/css">
  table{background-color: #EEE;}
  thead, tfoot{background-color:#DDD;color: blue;}
  tr.fonce {background-color: #333;color: white;}
  col{background-color: orange;}
  #max{background-color: red;color: yellow; }
 </style>
</head>
<body>
 <table border="0" summary="Valeur de l'indice" >
  <caption>Indice du coût de la construction</caption>
  <col span="2" width="25%" align="center"/>
  <thead>
   <tr>
    <th>Trimestre</th>
    <th>Indice </th>
    <th>Moyenne </th>
    <th>Variation annuelle en % </th>
    <th>Date de parution</th>
   </tr>
  </thead>
  <tfoot>
   <tr>
    <th>Trimestre</th>
    <th>Indice </th>
    <th>Moyenne </th>
    <th>Variation annuelle en % </th>
    <th>Date de parution</th>
   </tr>
  </tfoot>
  <tbody>
   <tr>
    <td>1er trimestre 2005</td>
    <td>1270</td>
    <td>1269,50</td>
    <td id="max">+4,83</td>
    <td>08/07/2005</td>
   </tr>
   <tr>
    <td>4ème trimestre 2004</td>
    <td>1269</td>
    <td>1258,25</td>
    <td>+4,81</td>
    <td>08/04/2005</td>
   </tr>
```

```
  <tr class="fonce">  ,
   <td>3ème trimestre 2004</td>
   <td>1272</td>
   <td>1244,50</td>
   <td >+4,58</td>
   <td>12/01/2005</td>
  </tr>
  <tr>
   <td>2ème trimestre 2004</td>
   <td>1267</td>
   <td>1227,25</td>
   <td>+3,85</td>
   <td>15/10/2004</td>
  </tr>
  < !-- Suite des données -->
 </tbody>
</table>
</body>
</html>
```

La figure 14-2 montre le résultat obtenu. On remarque bien l'empilement des couleurs de fond des différents éléments.

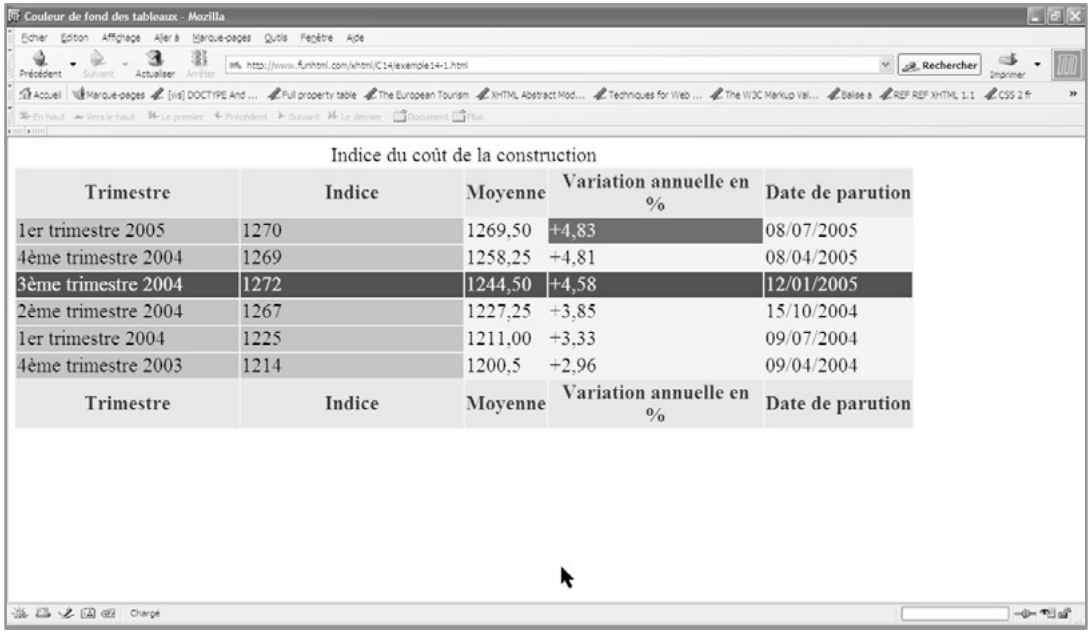


Figure 14-2

L'empilement des couleurs de fond

Dans l'exemple 14-2, nous verrons qu'il est possible de créer des styles spécifiques pour mettre en évidence des colonnes ou des groupes de colonnes.

Les titres des tableaux

L'élément <caption> vu au chapitre 6, contient le titre du tableau. Par défaut, en XHTML pur, il est placé au-dessus du tableau. Nous pouvons déterminer explicitement sa position à l'aide de la propriété caption-side dont la syntaxe est la suivante :

```
caption-side: top | bottom | inherit
```

Elle n'est applicable qu'à l'élément <caption> ou à ceux dont la propriété display a la valeur table-caption et elle est héritée pour les tableaux inclus l'un dans l'autre. Avec la valeur top, le titre est placé au-dessus du tableau, et en dessous avec la valeur bottom. Les valeurs qui permettaient de placer le titre à gauche ou à droite ayant été supprimées de la version CSS 2.1 par manque d'implémentation, nous pouvons créer ces positions en ajoutant une colonne à gauche ou à droite du tableau, en fusionnant toutes ces cellules et y écrivant le titre sans utiliser l'élément <caption>. On procède dans ce cas à l'application de styles particuliers en utilisant un sélecteur d'id ou une classe pour cette cellule de titre.

Que le titre soit situé au-dessus ou en dessous du tableau, les marges haute et basse définies pour les éléments <caption> et <table> fusionnent au profit de la plus grande, sauf dans Internet Explorer qui place systématiquement ces marges au-dessus du tableau et non pas entre le titre et le tableau.

L'exemple 14-2 présente une facture incluse dans un tableau. Nous y définissons des styles pour l'élément <caption> (repère º) afin d'améliorer sa présentation. Il s'agit ici de la création d'une bordure, d'une marge basse et du choix de la police et de la taille des caractères (repère ·), mais nous pourrions également lui ajouter une couleur de fond et d'avant-plan par exemple. La marge basse du titre qui est de 20 pixels fusionne avec la marge haute du tableau (repère ³) qui est de 40 pixels, pour obtenir une marge unique de 40 pixels. Notons également que la largeur de bordure définie pour le tableau dans l'attribut border, qui a la valeur 15 (repère ¶), n'est pas prise en compte, mais écrasée par celle qui est définie à 8 pixels dans le style CSS par la propriété border (repère ³). Cela démontre une fois encore que les styles CSS l'emportent toujours sur les définitions du code XHTML.

Nous illustrons par la même occasion l'affectation de couleurs de fond différentes à des groupes de colonnes. Les données sont réparties en trois groupes, créés dans trois éléments <colgroup> (repères ¾, µ et ¸), chacun utilisant une des classes de couleur .gras, .prix et .date (repères », ¿ et ´), ce qui permet d'attribuer une couleur différente à chaque groupe.

Exemple 14-2. Création du titre d'un tableau

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

```
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Titre et groupement de colonnes</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
<style type="text/css" title="tableau">
 table{margin-top: 40px;border: blue double 8px;font-size: 18px;}
 caption{border: blue double 3px;margin-bottom:20px;
  font-family: Arial,sans-serif;font-size: 1.5em;}
 .gras{font-weight:bold; background-color: #CCC;}
 .prix{background-color: yellow;}
 .date{background-color:#AAA;color:blue;}
 col {text-align: center;}
</style>
 </head>
 <body>
  <table border="15" width="100%" rules="groups" summary="Facture de livres"
   cellpadding="5">
   <caption>Facture de votre commande de livres</caption>
   <!-- Groupe 1 : Dates -->
    <colgroup id="date" width="10%" span="1" align="left" class="date">
    </colgroup>
    <!-- Groupe 2 : Titre et Auteur -->
    <colgroup  id="titre" span="2" width="25%" align="left" class="gras">
    </colgroup>
    <!-- Groupe 3: Quantité, Prix unitaire, Prix total -->
    <colgroup id="prix" span="3" align="left" class="prix">
     <col  width="10%" />
     <col span="2" width="15%" />
    </colgroup>
   <thead><tr><th>Date </th><th>Titre </th><th>Auteur</th><th>Quantité </th>
    <th>Prix Unitaire </th><th>Prix Total </th></tr></thead>
   <tfoot><tr><th>Date </th><th>Désignation </th><th>Auteur</th><th>Quantité
    </th><th>Prix Unitaire </th><th>Prix Total </th></tr></tfoot>
      <!-- Données du tableau -->
   <tbody>
    <tr>
     <td>29/05/2005</td><td>XHTML Design</td><td>Jeffrey Zeldman</td><td>3 </td>
      <td>32.00 &euro;</td><td>96.00 &euro;</td>
    </tr>
    <tr>
     <td>15/06/2005</td><td>CSS 2 </td><td>Raphael Goetter</td><td>2</td>
      <td>30.00 &euro;</td><td>60.00 &euro;</td>
    </tr>
    <tr>
     <td>15/12/2005</td><td>XHTML et CSS </td><td>Jean Engels</td><td>3</td>
      <td>29.90 &euro;</td><td>89.70 &euro;</td>
    </tr>
```

```
  </tbody>
 </table>
</body>
</html>
```

La figure 14-3 montre le résultat obtenu pour le titre et les couleurs des colonnes.

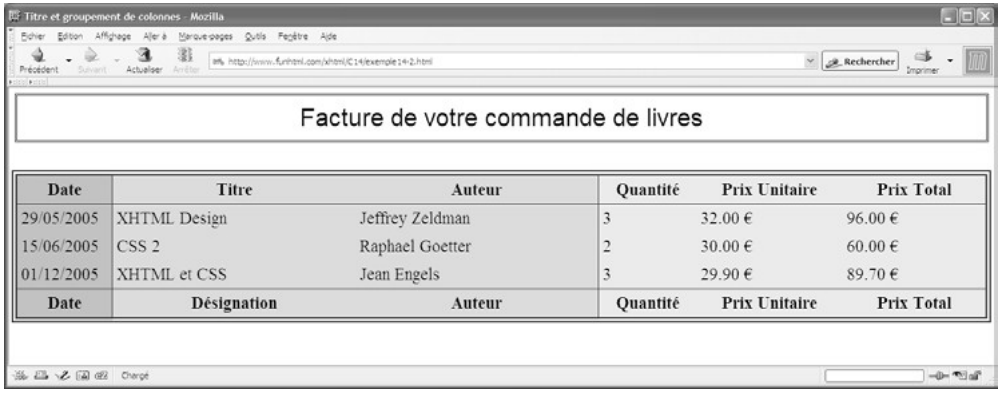


Figure 14-3

Titre de tableau et groupes de colonnes

La gestion des bordures des cellules

Les différentes propriétés de création des bordures vues au chapitre 11 sont applicables aux cellules des tableaux. Pour améliorer la gestion des bordures et en particulier celle des cellules contiguës, CSS fournit la propriété `border-collapse` qui permet de fusionner ou de séparer ces bordures voisines. Sa syntaxe est la suivante :

```
border-collapse : collapse | separate | inherit
```

Elle s'applique à tous les éléments de tableau et elle est héritée par tous les éléments enfants de celui auquel on l'applique.

À la valeur `collapse` correspond la fusion des bordures, et à la valeur `separate`leur séparation.

Les bordures séparées

Avec la valeur `separate` chaque cellule peut avoir une bordure particulière indépendante de celle de ses voisines directes. En particulier, une bordure ne va pas empiéter sur sa voisine même si son épaisseur est importante. En complément, la propriété `border-spacing` permet de définir un espacement entre les bordures voisines. Elle n'est

utilisable que si les bordures sont déclarées séparées avec `border-collapse`. Sa syntaxe est la suivante :

```
border-spacing : <longueur> <longueur> ? | inherit
```

Elle s'applique également à tous les éléments de tableau inclus dans `<table>` et elle est héritée par défaut. Les deux paramètres `<longueur>` définissent respectivement les espacements horizontaux et verticaux entre les cellules. La seconde valeur est facultative, et dans ce cas l'unique valeur indiquée s'applique aux deux espacements. Ces valeurs sont données par un nombre positif et une unité de longueur. Dans l'espacement créé entre les bordures apparaît la couleur de fond définie pour l'élément `<table>` ou un de ses enfants situés plus haut dans l'ordre de priorité défini plus haut.

Une cellule est matérialisée par les bordures entourant son contenu et par sa couleur de fond. Quand elle est vide, on peut soit la matérialiser quand même en affichant ses caractéristiques, soit l'effacer en empêchant l'apparition des bordures et du fond. Une cellule est considérée non vide si elle contient ne serait-ce qu'un seul caractère (mais pas une espace) ou même l'entité ` `. En revanche, si elle contient les codes ASCII hexadécimaux du retour chariot (`\0D`), de nouvelle ligne (`\0A`), de tabulation (`\09`) ou d'espace (`\20`) qui peuvent être générés avec la propriété `content` par exemple, elle est quand même considérée comme vide. De plus, une cellule pour laquelle la propriété `visibility` est définie avec la valeur `hide` est aussi considérée comme vide.

Pour déterminer le comportement des cellules vides, nous utilisons la propriété `empty-cells` dont voici la syntaxe :

```
empty-cells : show | hide | inherit
```

Elle ne s'applique qu'aux éléments `<td>` et `<th>`. La valeur `show` permet l'affichage du fond et des bordures, et `hide` cache l'ensemble de la cellule et laisse donc apparaître la couleur de fond définie pour son premier élément parent (`<tr>`, `<tbody>`, `<tfoot>`, `<thead>`, `<col />`, `<colgroup>` ou `<table>`).

Comportement de Internet Explorer

Même avec la valeur `hide`, Internet Explorer laisse apparaître la couleur de fond de la cellule si elle a été définie explicitement.

L'exemple 14-3 permet de vérifier le modèle des bordures séparées. Pour l'élément `<table>`, la propriété `border-collapse` y est définie avec la valeur `separate` (repère ³), l'espacement horizontal entre les bordures a une valeur de 20 pixels et l'espacement vertical est de 5 pixels (repère ·). Les cellules vides (telle celle du repère ¶) sont définies comme cachées en donnant la valeur `hide` à la propriété `empty-cells` (repère »). Les bordures des cellules ont le style `double` et une largeur de 5 pixels (repère ¿). De plus, une bordure particulière nettement plus large est créée pour la cellule qui contient la valeur maximale de la colonne « Variations » (repères ´ et ²).

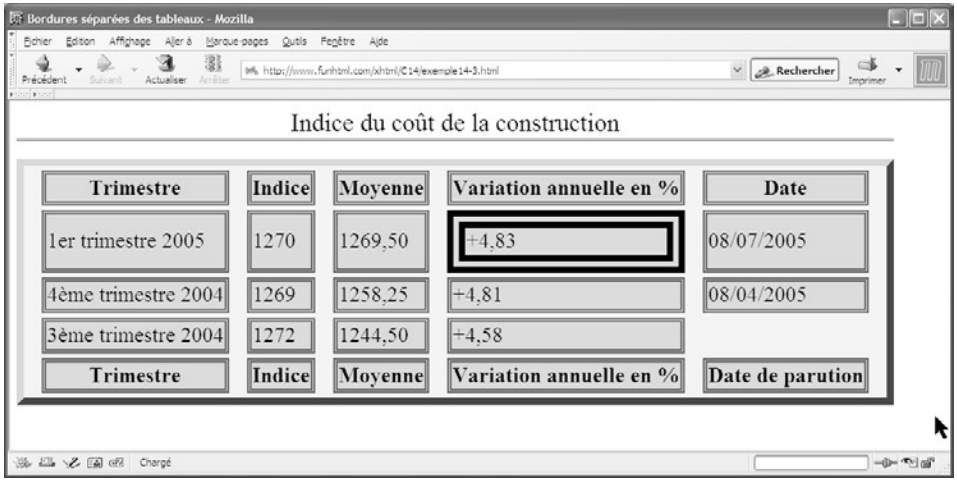
Exemple 14-3. Les bordures séparées

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
➥ "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Bordures séparées des tableaux</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css">
   table{background-color: #EEE; border-collapse: separate³  ;
    ➥ border-spacing: 20px 5px·  ; border: outset 8px red; empty-cells: hide  » ;}
   caption{font-size: 1.2em; border-bottom: red double 3px;margin-bottom: 20px;}
   tr,td,th{border: double 5px red; background-color: #CCC} ¿
   .maxi{border: double black 20px;} ´
  </style>
 </head>
 <body>
  <table summary="Valeur de l'indice">
   <caption>Indice du coût de la construction</caption>
   <thead>
    <tr>
     <th>Trimestre</th>
     <th>Indice</th>
     <th>Moyenne</th>
     <th>Variation annuelle en %</th>
     <th>Date</th>
    </tr>
   </thead>
   <tfoot>
    <tr>
     <th>Trimestre</th>
     <th>Indice</th>
     <th>Moyenne</th>
     <th>Variation annuelle en %</th>
     <th>Date de parution</th>
    </tr>
   </tfoot>
   <tbody>
    <tr>
     <td>1er trimestre 2005</td>
     <td>1270</td>
     <td>1269,50</td>
     <td class="maxi">+4,83</td> ²
```

```
    <td>08/07/2005</td>
   </tr>
   <tr>
    <td>4ème trimestre 2004</td>
    <td>1269</td>
    <td>1258,25</td>
    <td>+4,81</td>
    <td>08/04/2005</td>
   </tr>
   <tr>
    <td>3ème trimestre 2004</td>
    <td>1272</td>
    <td>1244,50</td>
    <td >+4,58</td>
    <td> </td>  ¶
   </tr>
   <!—Suite des données -->
  </tbody>
 </table>
</body>
</html>
```

La figure 14-4 montre le résultat obtenu dans Mozilla. Nous constatons que les bordures des cellules sont bien séparées, qu'à la deuxième ligne la définition d'une seule bordure plus large a entraîné l'augmentation de la hauteur de toute la ligne et que la cellule vide de la dernière colonne est entièrement cachée.

Figure 14-4
Les bordures séparées



Les bordures fusionnées

Quand la propriété `border-collapse` prend la valeur `collapse`, les bordures des cellules mitoyennes ne sont plus indépendantes mais elles fusionnent pour n'en faire plus qu'une. En utilisant le même code XHTML que celui de l'exemple 14-3 et en lui appliquant les styles suivants :

```
<style type="text/css">
  table{background-color: #EEE; border-collapse: collapse;
  ➥ 3  border: outset 8px red; empty-cells: hide;}
  caption{font-size: 1.2em;border-bottom: red double 3px;margin-bottom: 20px;}
  td,th{ border: double 5px red; background-color: #CCC}
  .maxi{border: double black 20px;}
</style>
```

dans lesquels la valeur de la propriété `border-collapse` a été modifiée et prend la valeur `collapse` (repère ³) et la propriété `border-spacing` a été supprimée, car elle n'a aucun effet quand les bordures sont fusionnées, nous obtenons le résultat présenté à la figure 14-5, où nous pouvons remarquer que l'aspect du tableau est très différent de celui de la figure 14-4. Notons également que la bordure appliquée à la cellule contenant la plus grande variation déborde sur l'emplacement des cellules voisines. Dans la pratique, il faudrait diminuer nettement sa largeur pour éviter cet effet disgracieux.

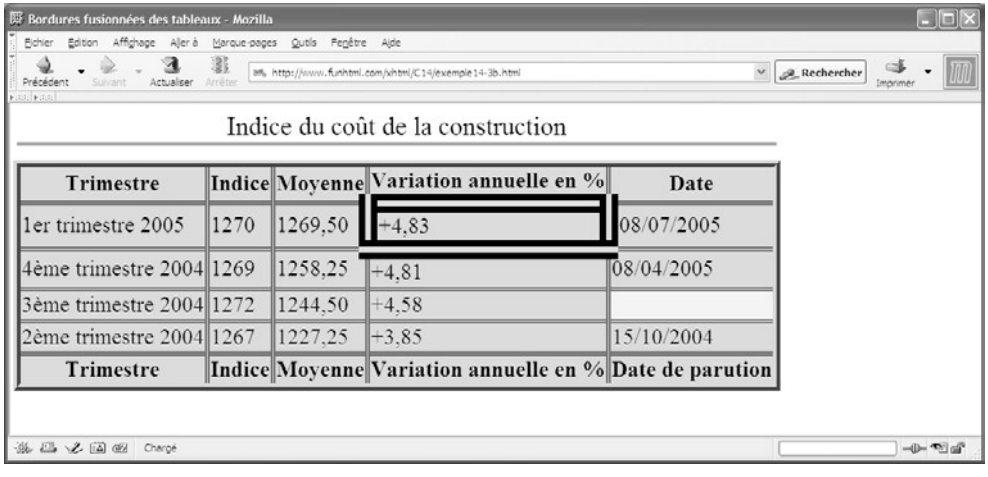


Figure 14-5

Tableau avec des bordures fusionnées

Dans cet exemple, toutes les bordures des cellules ont le même style, et l'affichage obtenu est uniforme. Dans les cas où nous définissons des bordures différentes pour des groupes de lignes, de colonnes ou des cellules particulières, il survient des conflits entre

ces définitions puisqu'une seule bordure sépare désormais ces éléments. Ces cas de conflits sont résolus de la manière suivante :

- Si la propriété `border-style` d'un élément a la valeur `hidden`, la bordure de cet élément est toujours cachée, quelle que soit la définition donnée pour l'élément voisin.
- Si la propriété `border-style` a la valeur `none`, il faut que toutes les autres définitions concernant la même bordure aient aussi cette valeur `none` afin que la bordure soit cachée, sinon elle reste visible. Si un seul élément a une valeur différente de `none`, alors la bordure la plus large est affichée. En cas d'égalité de largeur et quand la propriété `border-style` a une valeur différente pour les deux cellules voisines, les styles sont retenus dans l'ordre de préséance suivant : `double`, `solid`, `dashed`, `dotted`, `ridge`, `outset`, `groove` et `inset`.
- Si les bordures des différents éléments ont toutes en commun la même largeur et le même style, mais simplement une couleur différente, la couleur qui l'emporte est celle de l'élément situé le plus haut dans l'ordre de priorité défini pour les couleurs de fond, que nous avons indiqué dans une section précédente et dont le schéma est représenté à la figure 14-1.

L'exemple 14-4 permet de vérifier la manière dont CSS règle les nombreux conflits existant dans cette page entre les différents styles de bordures affectés aux cellules d'un tableau quand elles sont fusionnées, donc quand la propriété `border-collapse` prend la valeur `collapse` (repère ³). La bordure extérieure de l'ensemble du tableau a une largeur de 15 pixels (repère ·) et elle va donc l'emporter sur celles des cellules qui ont toutes des largeurs inférieures. En revanche, cette bordure disparaît autour de la cellule L1C1 dont la propriété `border` est définie avec la valeur `hidden` (repère »). Cette valeur cause également l'effacement de la bordure gauche de la cellule L1C2 et de la bordure haute de L2C1. La bordure définie pour le sélecteur `td` (repère ¿) devrait s'appliquer à toutes les autres cellules. Il n'en est rien et elle ne s'applique, et encore que partiellement, qu'aux cellules L2C1, L2C2, L3C1 et à la quatrième ligne pour les raisons suivantes :

- L1C2 a une bordure dont le style `double` (repère ´) l'emporte sur le style `solid` à épaisseur égale.
- L1C3 a une bordure gauche écrasée par celle de L1C2 pour la même raison et une bordure basse écrasée par celle qui est définie pour L2C3 dans la classe `.dashed` (repère ²), car son épaisseur est de 12 pixels.
- L2C2 a une bordure haute écrasée par celle de L1C2, une bordure droite dominée par celle de L2C3 pour la même raison que L1C3 et une bordure basse écrasée par celle de L3C2 définie dans la classe `.double` appliquée à L3C2 et L3C3, et qui l'emporte en fonction de son épaisseur de 10 pixels (repère ¶). Il en est de même pour les cellules L4C2 et L4C3.
- Le style défini pour la deuxième colonne (repère º) ne s'applique en définitive à aucune cellule car il est dominé par tous les autres, soit par l'épaisseur supérieure, soit par le style `solid` qui l'emporte sur le style `dotted`.

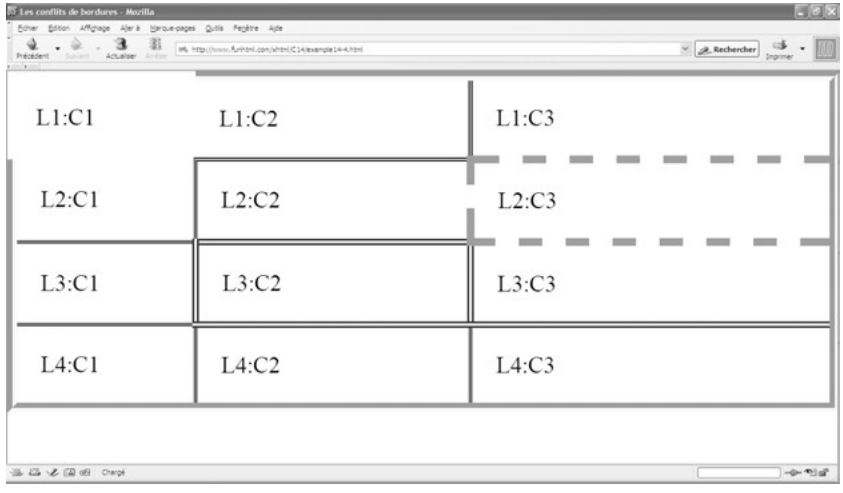
La figure 14-6 montre le résultat obtenu, lequel permet de constater l'application des règles de priorité vues ci-dessus.

Exemple 14-4. Les conflits entre les bordures voisines

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les conflits de bordures</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" >
   table {border-collapse: collapse ³ ; border: 15px inset yellow ˙ ;
    ➥ font-size: 1.5em;}
   #l1c1{border: hidden; » }
   td {border: 6px solid red; padding: 1em;} ¿
   #l1c2 {border: 6px double black;} ´
   td.dashed {border: 12px dashed gray;}²
   td.double {border: 10px double blue;} ¶
   #col2 {border: 6px dotted black;} º
  </style>
 </head>
 <body>
  <table width="100%">
   <col  width="20%"/>
   <col id="col2" width="30%" />
   <col width="40%"/>
   <tr>
    <td id="l1c1"> L1:C1 </td>
    <td id="l1c2"> L1:C2 </td>
    <td> L1:C3 </td>
   </tr>
   <tr>
    <td> L2:C1 </td>
    <td> L2:C2 </td>
    <td class="dashed"> L2:C3 </td>
   </tr>
   <tr>
    <td> L3:C1 </td>
    <td class="double"> L3:C2 </td>
    <td class="double"> L3:C3 </td>
   </tr>
   <tr>
    <td> L4:C1 </td>
    <td> L4:C2 </td>
```

```
    <td> L4:C3 </td>
   </tr>
  </table>
 </body>
</html>
```

Figure 14-6
Résolution des conflits entre les bordures



Déterminer la largeur d'un tableau

La largeur d'un tableau peut être déterminée de plusieurs manières. Celles-ci diffèrent en fonction du contenu de ses cellules ou des dimensions données explicitement à ces dernières. Pour déterminer cette largeur, les navigateurs utilisent différents algorithmes selon la valeur prise par la propriété `table-layout` dont la syntaxe est la suivante :

```
table-layout : auto | fixed | inherit
```

Elle s'applique à l'élément `<table>` et à ceux dont la propriété `display` a les valeurs `table` ou `inline-table` .

En utilisant la valeur `fixed`, la largeur du tableau ne tient pas directement compte du contenu de chacune des cellules, mais seulement de la largeur explicite, différente de `auto`, de l'élément `<table>`, de celles des colonnes `<col />` et des cellules `<td>` ou `<th>` Cette largeur est calculée de la manière suivante :

- La largeur de l'élément `<table>` peut être définie explicitement, soit dans son attribut `width`, soit dans sa propriété `width`. Toutefois, la définition de cette propriété avec la valeur `auto` ne met pas en œuvre l'algorithme `fixed` .
- Si une colonne de tableau a une largeur explicite définie par sa propriété `width` (autre que `auto`), cette largeur est utilisée dans l'affichage du tableau.

- Quand une cellule de la première ligne a une largeur explicite autre que auto, cette valeur est utilisée pour toute la colonne à laquelle elle appartient. Dans le cas où cette cellule est la fusion de plusieurs cellules virtuelles (voir le chapitre 6) réalisée en définissant son attribut colspan et que la ligne suivante contient réellement plusieurs cellules, la largeur de la cellule fusionnée est partagée entre chacune des cellules de la ligne suivante, de manière égale.

- Pour les autres colonnes du tableau qui n'ont pas une largeur définie explicitement par une valeur numérique ou si elles ont la largeur auto, l'espace restant est partagé équitablement entre elles (voir l'exemple 14-5).

Cet algorithme présente l'avantage d'être plus rapide car seules les largeurs des colonnes et des cellules de la première ligne sont prises en compte, et le navigateur n'examine pas toutes les suivantes pour déterminer l'affichage.

L'exemple 14-5 donne une première illustration de ce mécanisme quand la propriété table-layout a la valeur fixed (repère ³). Dans ce tableau, la largeur affectée à l'élément <table> est de 100 % (repère »). La seule cellule qui se voit affectée une largeur est la première de la deuxième colonne par l'intermédiaire de la classe .larg qui la détermine à 150 pixels (repère ·). En conséquence, les colonnes 1, 3 et 4 (repères ¿ , ² et ¶) ont pour largeur commune un tiers de celle de l'écran, diminuée des 150 pixels de la deuxième colonne, soit pour un écran de 1 024 pixels : (1 024 – 150) / 3 = 291 pixels, valeur arrondie au pixel près et diminuée des bordures éventuelles. Remarquons que la dernière colonne, bien qu'ayant un contenu très petit, a la même largeur que les colonnes 1 et 3. Pour un écran plus petit, seule la deuxième colonne garderait une largeur fixe de 150 pixels, les trois autres se partageant l'espace restant.

Dans la mesure où la deuxième cellule de la première ligne est la fusion de deux colonnes (repère ´), les cellules 2 et 3 (repères º et ¾) de la seconde ligne se partagent sa largeur et font donc 75 pixels chacune. L'inconvénient de ce partage automatique est qu'il ne tient pas compte du contenu des cellules. Dans notre exemple la deuxième cellule de la deuxième ligne voit son contenu coupé (dans Opera par exemple) ou débordant (dans Mozilla et FireFox) sur la cellule suivante selon les navigateurs, car aucune césure des mots n'est effectuée. Ce problème se règle théoriquement à l'aide de la propriété overflow qui permet de cacher ou de montrer le débordement (voir le chapitre 13), mais les navigateurs gèrent très mal ou de façon discordante cette propriété pour les cellules.

Exemple 14-5. Dimensionnement des colonnes dans l'algorithme fixed

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
```

```
 <title>La largeur des tableaux</title>
 <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1"/>
 <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico"/>
 <style type="text/css">
  table {table-layout: fixed;}
  td.larg {width: 150px;}
 </style>
</head>
<body>
 <table width="100%" border="1">
  <tr valign="top">
    <td> In principio creavit Deus caelum et terram terra autem erat inanis et
    ➥ vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
    ➥ dixitque Deus fiat lux et facta est lux et vidit Deus lucem quod esset bona
    ➥ et divisit lucem ac tenebras appellavitque lucem diem et tenebras noctem
    ➥ factumque est vespere et </td>
    <td class="larg" colspan="2">XHTML</td>
    <td> In principio creavit Deus caelum et terram terra autem erat inanis et
    ➥ vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
    ➥ dixitque Deus fiat lux et facta est lux et vidit Deus lucem quod esset bona
    ➥ et divisit lucem ac tenebras appellavitque lucem diem et tenebras noctem
    ➥ factumque est vespere et </td>
    <td> CSS 2 </td>
  </tr>
  <tr valign="top">
   <td> In principio creavit Deus caelum et terram terra autem erat inanis et
    ➥ vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
    ➥ dixitque Deus fiat lux et facta est lux et vidit Deus lucem quod esset bona
    ➥ et divisit lucem ac tenebras appellavitque lucem diem et tenebras noctem
    ➥ factumque est vespere et  </td>
    <td> XHTML 1.0abcdefghijk</td>
    <td> XHTML 1.1</td>
   <td> In principio creavit Deus caelum et terram terra autem erat inanis et
    ➥ vacua et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas
    ➥ dixitque Deus fiat lux et facta est lux et vidit Deus lucem quod esset bona
    ➥ et divisit lucem ac tenebras appellavitque lucem diem et tenebras noctem
    ➥ factumque est vespere et </td>
   <td>CSS 2</td>
  </tr>
 </table>
</body>
</html>
```

La figure 14-7 montre le résultat obtenu dans Opera, navigateur dans lequel le contenu débordant est coupé.

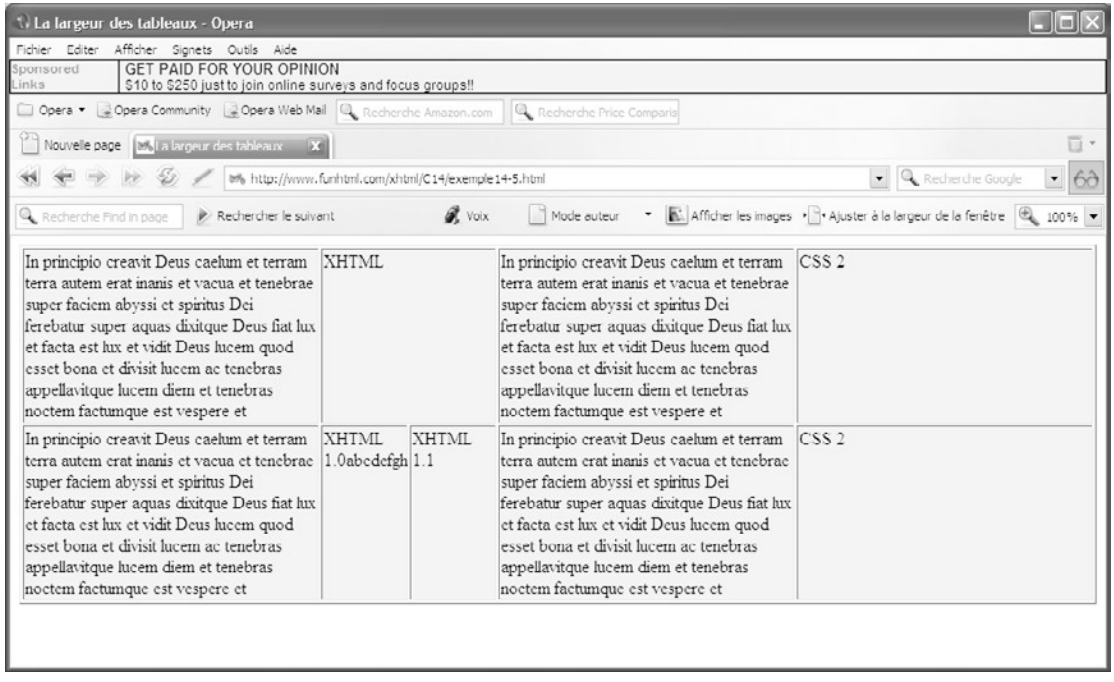


Figure 14-7

Largeurs de cellules dans l'algorithme fixed

Quand la propriété `table-layout` prend la valeur `auto`, l'algorithme utilisé par le navigateur procède de la manière suivante :

- Il calcule pour toutes les cellules la largeur minimale nécessaire pour afficher leur contenu. Cette largeur minimale peut par exemple correspondre à celle qu'il faut pour afficher en entier le mot le plus long d'un texte. Si la propriété `width` est définie pour une cellule et qu'elle est supérieure à la largeur calculée précédemment, c'est elle qui devient la largeur minimale.
- Il détermine la largeur minimale de chaque colonne en fonction de celles des cellules qu'elle contient, la plus grande devenant celle de la colonne. Si une propriété `width` est définie pour un élément `<col />` et qu'elle est supérieure à cette valeur minimale, c'est elle qui devient la largeur minimale. Il suffit donc qu'une cellule contienne un objet très large (une image par exemple) pour que la colonne entière ait cette largeur minimale. Dans ce cas, il est préférable de fusionner des cellules pour en créer une seule qui aura ce contenu sans obliger toutes celles de la même colonne à avoir la même largeur.

En reprenant exactement le même code contenu dans l'élément <body> que celui de l'exemple 14-5 et en modifiant les styles de la manière suivante :

```
<style type="text/css">
table {table-layout: auto;}
td.larg {width: 150px;}
</style>
```

dans lesquels la propriété table–layout a maintenant la valeur auto, le navigateur détermine la largeur des colonnes de la façon suivante :

- La largeur de la deuxième colonne est déterminée en fonction du contenu des cellules 3 et 4 de la seconde ligne. La somme de ces largeurs donne celle de la cellule fusionnée située au-dessus des cellules 3 et 4. Cette somme est supérieure à la largeur de 150 pixels définie explicitement pour la cellule fusionnée (repère · et repère ´ de l'exemple 14-5).
- La largeur de la dernière colonne est la largeur minimale pour contenir le texte « CSS 2.1 ».
- Le reste de la largeur disponible dans le tableau est ensuite partagé en deux parties égales pour contenir les textes des cellules des colonnes 1 et 3.

La figure 14-8 montre le résultat obtenu, lequel est très différent de celui de la figure 14-7 illustrant l'algorithme fixed.

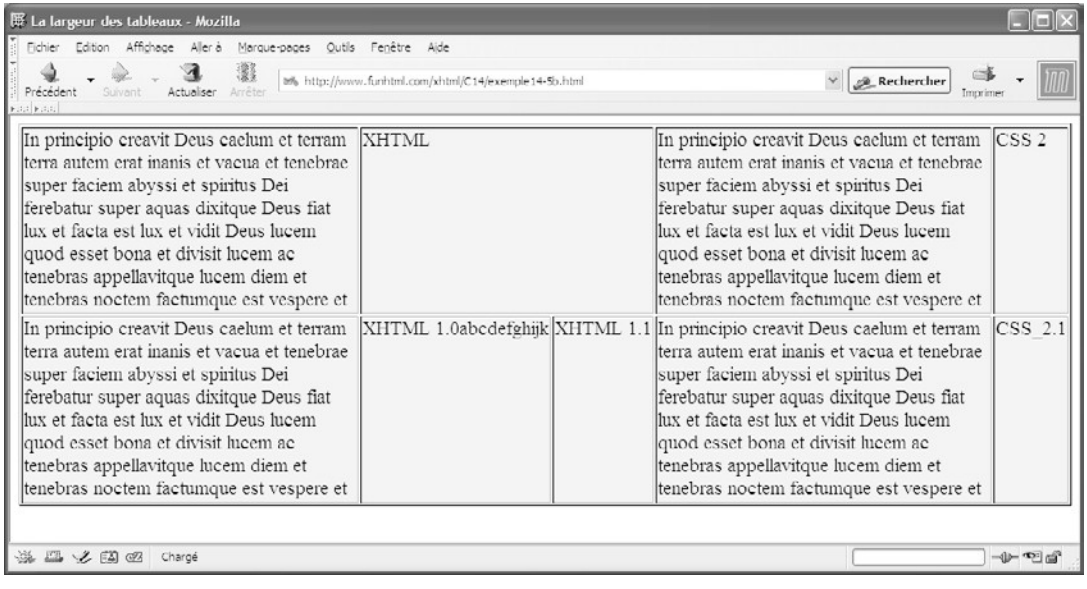


Figure 14-8

Largeurs de cellules dans l'algorithme auto

Présentation d'un formulaire

Les tableaux constituent un bon modèle de conteneur pour les éléments d'un formulaire. En effet, un formulaire présente souvent un aspect général assez similaire, reposant de façon classique sur une structure à deux colonnes, la première contenant les libellés des champs et expliquant la saisie attendue, et la seconde contenant les composants eux-mêmes.

L'exemple 14-6 présente une illustration de la création de styles visant à rendre plus attractif un formulaire au niveau de sa présentation.

Au niveau de la structure XHTML, l'élément `<form>` contient un tableau (repère µ) qui inclut lui-même un titre dans l'élément `<caption>` (repère ¸), puis la définition des deux colonnes et en particulier de leurs largeurs respectives (repères ¹ et), données ici en pourcentage. Chaque ligne du tableau contient un libellé dans l'élément `<label>` et un composant `<input>` ou `<select>` (il pourrait également s'agir de `<textarea>`). Les styles choisis ici sont arbitraires et chacun peut y apporter sa touche personnelle.

Dans la partie qui définit les styles, le tableau est dimensionné à 800 pixels de large, centré dans la page en définissant ses marges gauche et droite avec la valeur `auto` et ses bordures extérieures sont supprimées (repère ³). Le titre du tableau est affiché dans une taille supérieure à celle définie pour le tableau et il est muni de bordures haute et basse (repère ·). On définit les marges gauche et droite pour le titre à l'attention des navigateurs qui traitent le titre séparément du tableau.

Toutes les cellules présentent uniquement une bordure basse (repère »). Les colonnes des libellés et des composants sont mises en évidence par une couleur de fond différente (repères ¿ et ´) et les éléments `<label>` de la première colonne se distinguent par les styles de police italique et gras (repère ²). L'alignement à droite des libellés dans la première colonne est réalisé par l'attribut `align` de l'élément `<col />` (repère ¹). Les navigateurs de la famille Mozilla ne réalisant pas cet alignement, nous pouvons définir une classe `.droit` (repère ¶) et l'appliquer à chaque élément `<td>` de la première colonne ou définir pour chacun d'eux l'attribut `align` avec la valeur `right` .

Les libellés des champs dans lesquels la saisie est obligatoire sont signalés par des couleurs de fond et de texte différentes du reste de la colonne, en leur appliquant la classe `.oblig` (repère º). Enfin, le bouton d'envoi (créé par l'élément `<input>` de type `submit`) est signalé par une taille de police supérieure et une bordure, en utilisant un sélecteur de valeur d'attribut (repère ¾). Ce type de sélecteur n'est pas reconnu par Internet Explorer.

Exemple 14-6. Présentation d'un formulaire

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
  <title>Formulaire</title>
  <style type="text/css">
```

```
 table {margin-left: auto; margin-right: auto; width: 800px; border: none;
    ➥ table-layout: fixed; font-size: 16px;}
 caption {font-size: 1.4em; border-top: 3px blue double;
    ➥ border-bottom: 3px blue double; margin-bottom: 20px;
    ➥ margin-left: auto; margin-right: auto;}
 td {border-bottom: 2px solid red;}
 #libelle {background-color: #EEE;}
 #composants {background-color: #DDD;}
 label {font-style: italic; font-weight: bold;}
 .droit {text-align: right;}/*Pour Mozilla seul*/
 .oblige {background-color: gray; color: white;}
 input [type="submit"] {font-size: 1.5em; color: red; border: 2px solid blue;}
</style>
</head>
<body>
 <form action="exemple7-14.php" method="post" enctype="multipart/form-data">
  <table cellspacing="0">
   <caption>Vos coordonnées</caption>
   <col id="libelle" width="30%" align="right"/>
   <col id="composants" width="70%"/>
   <tr>
    <td class="oblige" ><label>Nom : </label></td>
    <td><input type="text" name="nom" size="40" maxlength="256"
       ➥ value="votre nom" tabindex="1"/></td>
   </tr>
   <tr>
    <td><label>Prénom : </label></td>
    <td><input type="text" name="prenom" size="40" maxlength="256"
       ➥ value="votre prénom" tabindex="2"/></td>
   </tr>
   <tr>
    <td class="oblige"><label>Mail : </label></td>
    <td><input type="text" name="mail" size="40" maxlength="256"
       ➥ value="votre mail" tabindex="3"/></td>
   </tr>
   <tr>
    <td><label>Sexe : </label></td>
    <td><input type="radio" name="sexe" value="homme" tabindex="4"/>Homme
       ➥ <input type="radio" name="sexe" value="femme" tabindex="5"/>Femme</td>
   </tr>
   <tr>
    <td><label>Votre pays : </label></td>
    <td>
     <select name="pays" size="1" tabindex="6">
      <option value="null"> Votre pays</option>
      <optgroup label="Europe">
       <option value="France"> France</option>
       <option value="Belgique"> Belgique</option>
```

```
        <option value="Italie"> Italie</option>
        <option value="Allemagne"> Allemagne</option>
       </optgroup>
       <optgroup label="Amérique">
        <option value="USA"> USA </option>
        <option value="Canada"> Canada</option>
        <option value="Argentine"> Argentine</option>
       </optgroup>
      </select>
     </td>
    </tr>
    <tr>
     <td class="oblige"><label>Code : </label></td>
     <td><input type="password" name="code" size="40" maxlength="6"
        ➥ tabindex="7"/></td>
    </tr>
    <tr>
     <td><label>Cliquez pour envoyer : </label></td>
     <td><input type="submit" id="envoi" value="Envoyer" tabindex="8"/></td>
    </tr>
   </table>
  </form>
 </body>
</html>
```

La figure 14-9 montre le résultat obtenu dans Opera.

Figure 14-9

Présentation d'un formulaire

Exercices

Exercice 1 : Si on définit une couleur de fond rouge pour l'élément `<table>` et bleue pour l'élément `<tr>`, quelle est la couleur réellement obtenue ?

Exercice 2 : Créez les styles pour que les couleurs de chaque ligne d'un même tableau soient différentes.

Exercice 3 : Créez les styles pour que les lignes paires d'un tableau aient un fond gris clair et un texte noir, et que les lignes impaires aient un fond gris foncé et un texte jaune.

Exercice 4 : Créez un tableau à sept colonnes réparties en trois groupes contenant respectivement 2, 3 et 2 colonnes. Appliquez des styles de couleur de fond et de texte différents pour chaque groupe.

Exercice 5 : Mettez en évidence trois cellules d'un tableau dont la couleur de fond est verte en définissant des styles propres pour ces cellules au moyen d'une classe, puis de sélecteurs d'attribut `id`.

Exercice 6 : Placer le titre d'un tableau en dessous de celui-ci en l'entourant d'une bordure double bleue de 3 pixels. Le texte doit être écrit dans une police Arial de 20 pixels.

Exercice 7 : Créez un tableau dont les bordures des cellules sont de type `solid` de 2 pixels de large et sont séparées les unes des autres. Cachez les cellules vides.

Exercice 8 : Créez un tableau pour lequel une ligne sur deux est munie de bordures, en utilisant le modèle des bordures fusionnées.

Exercice 9 : Incluez les éléments d'un formulaire dans un tableau à deux colonnes et écrivez les styles pour que chaque ligne ait une couleur différente selon qu'elle est paire ou impaire.

15

Le style des listes

De prime abord, les listes peuvent sembler être des éléments mineurs dans la structuration du contenu d'une page. Cependant, leur utilisation ne se limite pas au simple affichage d'un sommaire. Au niveau de la présentation, qu'elle soit ordonnée ou non, une liste a un aspect pauvre, et nous pouvons nous en satisfaire dans la mesure où son rôle est avant tout d'ordre sémantique, comme nous l'avons vu au chapitre 3. L'aspect esthétique d'une liste doit donc être élaboré à l'aide de styles CSS. Nous allons voir dans ce chapitre que les différents rendus visuels d'une liste peuvent être très variés tant au niveau des différentes possibilités de numérotation que s'agissant du choix des puces dans les listes non ordonnées. Les listes sont également très en vogue pour la création des menus d'un site, et c'est sur ce point que nous terminerons notre étude.

La numérotation des listes

Par défaut, la numérotation des items `<li>` inclus dans un élément `<ol>` s'effectue en chiffres arabes. Nous pouvons faire varier cette numérotation en attribuant un style à l'élément `<ol>` à l'aide de la propriété `list-style-type` dont la syntaxe est :

- `list-style-type :<type>|none|inherit`

Plus généralement, cette propriété ne s'applique qu'aux éléments de listes (`<ol>`, `<ul>` ou `<li>`) ou à ceux pour lesquels on a défini la propriété `display` avec la valeur `list-item`.

Pour les listes ordonnées, le paramètre <type> peut prendre une des valeurs suivantes :

- decimal : numérotation en chiffres arabes : 1, 2, 3… C'est la valeur par défaut.
- decimal-leading-zero : idem, mais les nombres inférieurs à 10 sont précédés d'un zéro : 01, 02, 03… Cette valeur n'est pas prise en compte par Internet Explorer.
- upper-latin ou upper-alpha : numérotation alphabétique en majuscules : A, B, C…
- lower-latin ou lower-alpha : numérotation alphabétique en minuscule : a, b, c…
- upper-roman : numérotation en chiffres romains majuscules I, II, III, IV…
- lower-roman : numérotation en chiffres romains minuscules i, ii, iii, iv…
- lower-greek : numérotation en caractères grecs minuscules , a, b, y… Cette valeur n'est pas prise en compte par Internet Explorer.
- georgian : numérotation en caractères géorgiens. Cette valeur n'est pas prise en compte par Internet Explorer.
- armenian : numérotation en caractères arméniens. Cette valeur n'est pas prise en compte par Internet Explorer.

La valeur none permet de supprimer toute numérotation. On peut l'utiliser pour conserver la sémantique des listes sans aucune numérotation. La valeur inherit permet d'obtenir explicitement la même numérotation que celle de l'élément parent, ce qui est appliqué par défaut car la propriété est héritée. L'exemple 15-1 nous permet de mettre en œuvre toutes les différentes possibilités de numérotation pour des listes ordonnées imbriquées sur deux niveaux, en appliquant des numérotations différentes pour chacun d'eux. Il contient quatre fois le même code XHTML de création des mêmes listes imbriquées (repères µ, ¸, ¹ et), chacune d'elles ayant un premier élément <ol> muni d'un identifiant id spécifique afin de lui appliquer un style propre, indépendamment des autres listes. Dans la définition des styles, le premier sélecteur (par exemple #deci) permet de créer le style des items de premier niveau, et le second (par exemple #deci li ol) celui des items imbriqués. La première liste est numérotée avec les styles decimal et decimal-leading-zero (repères · et »). La deuxième avec les styles upper-roman et lower-roman (repères ¿ et ´). La troisième utilise les styles upper-latin et lower-greek (repères ² et ¶) et la dernière les styles exotiques armenian et georgian (repères º et ¾). Le contenu des éléments de premier niveau est également mis en évidence en utilisant la propriété font-weight pour les afficher en gras. Notez que l'utilisation des attributs id ne serait pas nécessaire s'il n'y avait qu'une liste dans la page ou si nous voulions les numéroter toutes de la même façon. Nous pourrions en effet écrire simplement la définition des styles suivante (repère ³) :

```
ol {list-style-type:decimal; font-weight:bold; }
ol li ol{list-style-type:decimal-leading-zero; font-weight:lighter;}
```

Exemple 15-1. La numérotation des listes ordonnées

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les styles des listes ordonnées</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="Listes">
   /*Si la liste est unique dans la page, on peut écrire:
   ol {list-style-type:decimal; font-weight:bold; }
   ol li ol{list-style-type:decimal-leading-zero; font-weight:lighter;}*/
   #deci{list-style-type:decimal;font-weight:bold; }
   #deci li ol{list-style-type:decimal-leading-zero;font-weight:lighter; }
   #roman{list-style-type:upper-roman;font-weight:bold; }
   #roman li ol {list-style-type:lower-roman;font-weight:lighter; }
   #grec{list-style-type:upper-latin;font-weight:bold; }
   #grec li ol{list-style-type:lower-greek;font-weight:lighter; }
   #armenie {list-style-type:armenian;font-weight:bold; }
   #armenie li ol {list-style-type:georgian;font-weight:lighter; }
  </style>
 </head>
 <body>
  <h1>Liste ordonnée decimal puis decimal-leading-zero</h1>
   <ol id="deci">
   <li> XHTML 1.0
     <ol>
      <li> DTD transitional </li>
       <li> DTD strict </li>
       <li> DTD frameset </li>
     </ol>
   </li>
   <li> XHTML 1.1
    <ol>
      <li>DTD unique </li>
     </ol>
   </li>
    <li> CSS
    <ol>
     <li> CSS 1 </li>
      <li> CSS 2 </li>
```

```
    <li> CSS 2.1 </li>
   </ol>
  </li>
 </ol>
 <h1>Liste ordonnée upper-roman puis lower-roman</h1>
 <ol id="roman">,
  <!—MÊMES ÉLÉMENTS DE LISTES -->
 </ol>
 <h1>Liste ordonnée upper-latin puis lower-greek</h1>
 <ol id="grec">  1
  <!— MÊMES ÉLÉMENTS DE LISTES -->
 </ol>
 <h1>Liste ordonnée armenian puis georgian</h1>
 <ol id="armenie">
  <!— MÊMES ÉLÉMENTS DE LISTES -->
 </ol>
</body>
</html>
```

Les figures 15-1 et 15-2 illustrent les résultats obtenus pour ces différents styles de numérotation.

Figure 15-1
Les listes numérotées (décimales et en chiffres romains)

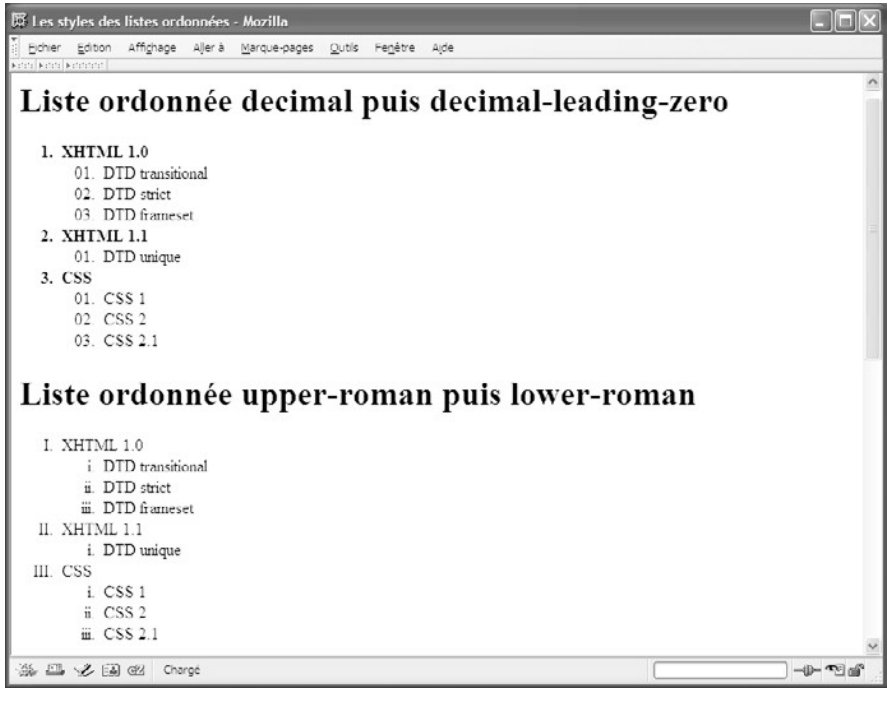
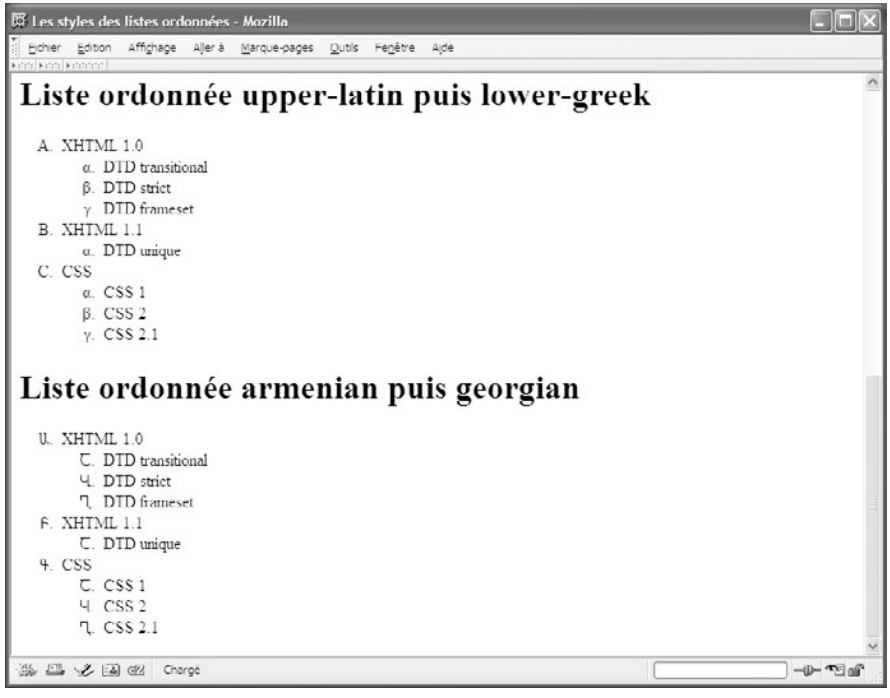


Figure 15-2
Les listes numérotées (latin-grec et en arménien-géorgien)



Les items d'une liste sont affichés par défaut en retrait par rapport au bord gauche de leur conteneur, laissant apparaître une marge gauche. Les caractères utilisés pour les numéroter font normalement partie intégrante du texte des items comme s'ils étaient écrits en tant que contenu des éléments <li> . Il est possible d'intervenir sur la position horizontale de ces caractères (chiffres ou lettres) en définissant la propriété list-style-position dont la syntaxe est la suivante :

list-style-position :inside|outside|inherit

- Avec la valeur inside , les caractères de la numérotation sont placés dans la marge de retrait du texte de l'item.
- Avec la valeur outside, qui est la valeur par défaut, ces caractères sont intégrés au texte de l'item.
- La valeur inherit permet de définir explicitement le style de l'élément parent mais la propriété est de toute façon héritée par défaut.

Dans l'exemple 15-2, le code des listes imbriquées est semblable à celui de l'exemple 15-1 (repères ´ et ²). Le style du premier niveau de liste est upper-roman et celui du second est lower-latin (repères ³ et · , puis » et ¿). Nous définissons successivement la position inside pour la première liste (repère ³) et outside pour la seconde (repère »).

Les définitions d'alignement des éléments de liste de second niveau sont ici explicites bien que la propriété list-style-positio n soit héritée, car certains navigateurs gèrent mal cet héritage.

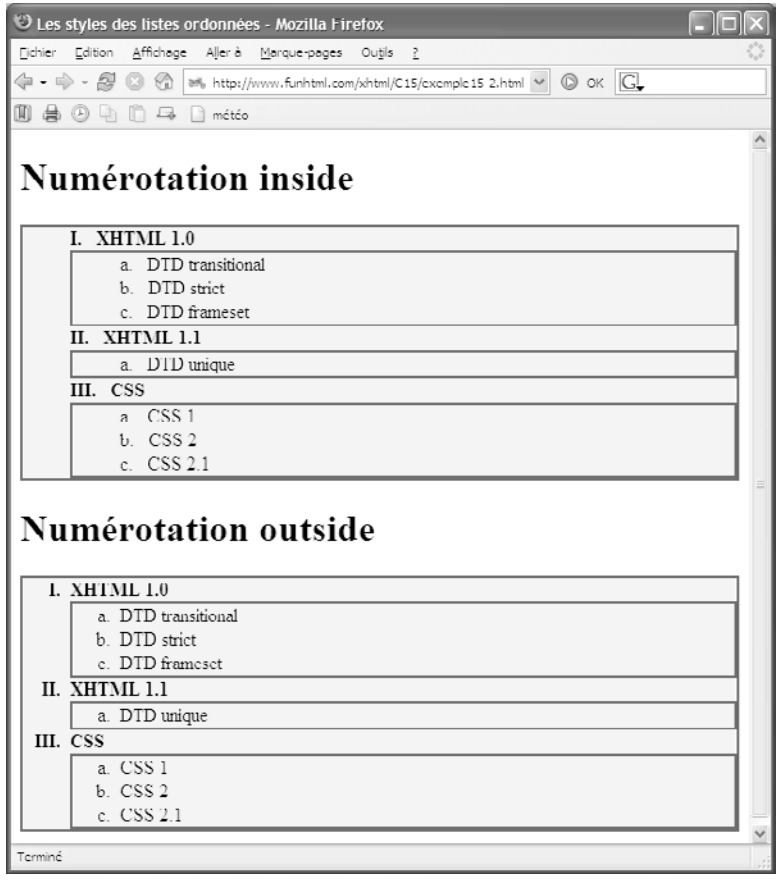
Exemple 15-2. Le positionnement des numérotations

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les styles des listes ordonnées</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="Listes">
   ol {list-style-type:upper-roman;border:solid red 2px;font-weight:bold;
    ➥list-style-position:inside;background-color:#EEE;}
   ol li ol{list-style-type:lower-latin;font-weight:lighter;
    ➥list-style-position:inside }
   #out{list-style-type:upper-roman;border:solid red 2px;font-weight:bold;
    ➥list-style-position:outside;background-color:#EEE;}
   #out li ol {list-style-type:lower-latin;border:solid red 2px;
    ➥font-weight:lighter;list-style-position:outside;}
  </style>
 </head>
 <body>
  <h1>Numérotation inside</h1>
  <ol>
   <li> XHTML 1.0
     <ol>
     <li> DTD transitional </li>
      <li> DTD strict </li>
     <li> DTD frameset </li>
     </ol>
   </li>
   <li> XHTML 1.1
    <ol>
     <li> DTD unique </li>
    </ol>
   </li>
    <li> CSS
    <ol>
     <li> CSS 1 </li>
```

```
    <li> CSS 2 </li>
    <li> CSS 2.1 </li>
   </ol>
  </li>
 </ol>
 <h1>Numérotation outside</h1>
 <ol id="out">
  <!—Les mêmes éléments de liste que ci-dessus-->
 </ol>
</body>
</html>
```

Sur la figure 15-3, nous pouvons constater les différences d'alignement des symboles de numérotation dues à l'utilisation des valeurs `inside` et `outside`.

Figure 15-3
Positionnement de la numérotation des listes



La création de compteurs

Les propriétés que nous venons d'étudier permettent de faire varier le type de numérotation des listes. Elles ne sont cependant pas adaptées, pour les listes imbriquées, à la réalisation de numérotations complexes tenant compte du symbole du niveau précédent. En particulier, elles ne permettent pas de créer des numérotations du type « I.A, I.B, I.C » ou même d'insérer un mot avant la numérotation pour obtenir automatiquement des listes identifiées par des en-têtes du type « Chapitre 1, Chapitre 2… ». Ce type de numérotation est adapté à la création de pages dynamiques dont le contenu répond à une demande du visiteur, et dont le nombre d'items ne peut être connu à l'avance.

Pour parvenir à ce type de résultats et à d'autres variantes, CSS offre plusieurs propriétés qui doivent être utilisées de concert.

Prise en charge par les navigateurs

Notez cependant qu'à ce jour, très peu de navigateurs implémentent complètement ces propriétés. Pour les exemples qui vont suivre, seul le navigateur Opera est capable de restituer l'effet recherché, Mozilla, FireFox et même Amaya (le navigateur pourtant créé par le W3C !) en étant incapables. Pour obtenir des effets visibles par tous, et en attendant une prise en charge plus complète, il sera préférable d'utiliser une programmation dynamique des pages, avec PHP ou ASP, qui fait appel à des boucles successives. En dépit de ces soucis de prise en charge, l'étude des compteurs n'est pas sans intérêt.

La procédure à suivre comporte plusieurs étapes. Il faut tout d'abord déclarer et initialiser un compteur. L'étape suivante consiste à définir la manière dont le compteur va être incrémenté. La phase finale permet d'afficher le contenu généré dans les listes.

On effectue la déclaration et l'initialisation du compteur à l'aide de la propriété `counter-reset` dont la syntaxe est la suivante :

```
counter-reset :[<identifiant> N ?]+ | none | inherit
```

L'identifiant du compteur est un nom arbitraire choisi par le programmeur comme c'est le cas pour l'attribut `id`. Si l'identifiant est le seul paramètre défini, le compteur est initialisé à 0 et le premier item de la liste est numéroté avec la valeur `1` (ou le caractère correspondant au style alphabétique choisi). S'il est suivi d'un entier N, la numérotation commence avec la valeur N + 1 (ou la N + 1e lettre). Les valeurs négatives de N sont admises à partir de – 32 768. On évitera ces valeurs négatives pour les numérotations alphabétiques ou romaines qui risquent de donner des résultats incohérents. En écrivant par exemple les styles suivants :

```
ol{counter-reset :partie ;}
```

les items de la liste sont numérotés 1, 2, 3… ou A, B, C…

Comme l'indique le symbole + de la syntaxe, il est possible de déclarer plusieurs compteurs dans la même propriété. En écrivant la déclaration suivante :

```
ol{counter-reset :partie 2 chapitre -1 paragraphe 7 ;}
```

le compteur nommé « chapitre » commence au numéro 3, le compteur nommé « chapitre » à 0, et celui nommé « paragraphe » à la valeur 8.

La deuxième étape définit l'incrémentation du compteur. C'est la propriété `counter-increment` qui permet cette opération. Sa syntaxe est la suivante :

```
counter-increment : [<identifiant> <N>]+ | none | inherit
```

L'identifiant est celui d'un des compteurs déclarés et l'entier facultatif qui le suit indique le pas de l'incrémentation (la valeur ajoutée à chaque étape et qui vaut 1 par défaut). Il est possible mais rare de définir plusieurs identifiants dans la même propriété. Le sélecteur utilisé dans la déclaration du style va préciser à quels éléments s'applique le compteur. Il faut en règle générale utiliser un sélecteur par niveau de numérotation, et donc aussi par compteur.

En écrivant les styles suivants :

```
ol{counter-reset : partie ;}
ol li{counter-increment :partie 2 ;}
```

le compteur est remis à 0 pour chaque nouvel élément <ol> et les items <li> sont numérotés de 2 en 2 à partir de 2.

À ce stade, rien ne s'affiche devant chaque item si ce n'est la numérotation habituelle. Si l'on veut créer une numérotation particulière, il est souvent préférable de supprimer la numérotation habituelle en donnant à la propriété `list-style-type` qu'on utilise la valeur `none`

Pour afficher le compteur, nous devons associer aux deux propriétés précédentes la propriété content que nous avons déjà rencontrée et qui va générer le contenu du compteur en étant associée aux pseudo-éléments `:before` ou `:after` .

Pour son utilisation avec des compteurs, sa syntaxe se réduit à la forme suivante :

```
content :[<chaine> | counter() ]+ | inherit
```

La chaîne insérée peut être un texte quelconque entre guillemets doubles, comme nous l'avons déjà vu par ailleurs. C'est la fonction `counter()`, qui reçoit comme premier paramètre l'identifiant du compteur, qui va permettre l'affichage des numérotations et qui peut recevoir un second paramètre facultatif pour désigner le style de la numérotation voulue, avec les mêmes mots-clés que ceux de la propriété `list-style-type` . Dans la même propriété `content`, nous pouvons utiliser plusieurs chaînes et plusieurs fonctions `counter()`.

L'exemple 15-3 constitue notre première illustration des compteurs. Il contient une liste des noms des villes olympiques entre 1992 et 2012 (repère ²). Notre objectif est d'afficher chaque item précédé du mot « Olympiade », puis de l'année des jeux, générée automatiquement et du caractère deux-points (:). Pour y parvenir, nous créons tout d'abord un

compteur nommé « jeux » et initialisé à la valeur 1988 (repère ·). On définit une taille de caractères pour les items de la liste (repère ³) et on attribue à la propriété list-style-type la valeur none pour supprimer la numérotation habituelle non désirée en tête des items (repère »). La liste n'ayant qu'un niveau, le sélecteur utilisé est ol li:before . Le compteur est incrémenté de 4 unités pour chaque item à l'aide de la propriété counter-increment (repère ¿) et la propriété content permet de générer le contenu qui va figurer devant chaque item à l'aide de la fonction counter() (repère ´).

Exemple 15-3. Numérotation automatique

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Numérotation</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="Listes">
   li {font-size: 1.5em;}      ³
   ol{counter-reset: jeux 1988    ·  ;list-style-type: none;    »  }
   ol li:before{counter-increment: jeux 4  ¿  ;content:"Olympiade " counter(jeux) " : "; ´  }
  </style>
 </head>
 <body>
  <h1>Jeux Olympiques</h1>
  <ol>  ²
   <li> Barcelone </li>
   <li> Atlanta</li>
   <li> Sydney</li>
   <li> Athènes</li>
   <li> Pékin</li>
   <li> Londres</li>
  </ol>
 </body>
</html>
```

La figure 15-4 montre le résultat obtenu dans Opera 8.0, qui correspond bien à notre attente. Les autres navigateurs se contentent d'afficher une numérotation de 1 à 6, mais ils ne manqueront pas de se mettre à la page un jour !

Notre second exemple (exemple 15-4) permet de générer une numérotation personnalisée plus complexe car elle s'applique à une liste imbriquée sur deux niveaux (repère µ). Nous voulons faire apparaître le mot « Partie » suivi d'une numérotation en chiffres romains majuscules devant les items du premier niveau, puis le mot « Chapitre » suivi du chiffre précédent et d'une numérotation décimale des items de second niveau (du type I.1, I.2, etc.).

Figure 15-4
Une numérotation simple sur un niveau

Après la définition d'une taille de police différente pour mettre en évidence les items de chaque niveau (repères ³ et ·), nous créons les compteurs nommés « titre1 » (repère ») et « titre2 » (repère ¿), puis nous supprimons la numérotation classique (repère ´). Pour le premier niveau, le sélecteur utilisé est `ol li:before` , pour lequel nous définissons l'incrémentation du premier compteur (repère ²) et le contenu généré par la propriété `content` dans laquelle nous précisons le style de numérotation souhaité comme second paramètre de la fonction `counter()` (repère ¶). Pour les items de second niveau, le sélecteur est `ol li ol li:before` pour lequel nous incrémentons le second compteur (repère °), puis nous créons le contenu généré qui utilise ici deux fois la fonction `counter()` avec deux paramètres, ainsi que deux chaînes de caractères (repère ¾).

Exemple 15-4. Les compteurs sur plusieurs niveaux

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les compteurs automatiques</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="Listes">
   <!-- Fontes -->
   ol li{font-size:1.5em;}      ³
   ol li ol {font-size:0.5em;}       ·
   <!-- Compteurs -->
   ol{counter-reset: titre1      » ;counter-reset:titre2    ¿ ;list-style-type: none;}      ´
   ol li:before{counter-increment: titre1       ² ;content: "Partie "
     ➥counter(titre1,upper-roman) " : ";   ¶ }
   ol li ol li:before{counter-increment: titre2          ° ;content: "Chapitre "
     ➥counter(titre1,upper-roman) "." counter(titre2,decimal) " : ";      ¾ }
  </style>
```

```
</head>
<body>
 <h1>Numérotation automatique</h1>
 <ol> µ
  <li> XHTML 1.0
    <ol>
    <li> DTD transitional </li>
     <li> DTD strict </li>
     <li> DTD frameset </li>
    </ol>
  </li>
  <li> XHTML 1.1
   <ol>
     <li> DTD unique </li>
   </ol>
  </li>
   <li> CSS
   <ol>
    <li> CSS 1 </li>
     <li> CSS 2 </li>
     <li> CSS 2.1 </li>
    </ol>
   </li>
 </ol>
</body>
</html>
```

Là encore, le résultat présenté à la figure 15-5 ne peut être obtenu pour l'instant que dans le navigateur Opera 8.

Figure 15-5
Numérotations automatiques de parties et de chapitres

La méthode précédente nécessite d'écrire un nouveau sélecteur pour chaque niveau de la liste imbriquée, ce qui implique des sélecteurs complexes si on augmente le nombre de niveaux. Si on peut se contenter du même style pour tous les niveaux d'items, il est possible de simplifier la procédure précédente en n'utilisant qu'un seul sélecteur et un seul compteur. Celui-ci est réinitialisé indépendamment à chaque niveau de la liste qui peut alors en contenir autant que souhaité. Pour créer cette fonctionnalité, il faut encore utiliser les mêmes propriétés `counter-reset` et `counter-increment` et simplement remplacer la fonction `counter()` par `counters()` qui peut avoir de un à trois paramètres. Le premier paramètre est à nouveau le nom du compteur, le deuxième est la chaîne qui va séparer chaque numérotation de niveau et le troisième le style de la liste qui est donc unique pour tous les niveaux.

Nous mettons en œuvre cette méthode dans l'exemple 15-5 dont le code XHTML est composé d'une liste numérotée à trois niveaux d'imbrication (les repères ¿ , ´ et ² identifient chacun des niveaux). La déclaration des styles est très simple car elle ne contient que deux éléments. Le premier initialise le compteur nommé « titres » implicitement à 0 avec la propriété `counter-reset` et supprime la numérotation par défaut en donnant la valeur `none` à la propriété `list-style-type` (repère ³). Le second définit le contenu du compteur avec la chaîne `Niveau` suivie de la numérotation créée par la fonction `counters()` utilisée avec trois paramètres, qui sont respectivement le nom du compteur unique, la chaîne de séparation `"."` et le style de la numérotation décimale (repère ·). Le compteur est ici explicitement incrémenté de une unité à chaque étape (repère »).

Exemple 15-5. Numérotation automatique des listes pour un nombre quelconque de niveaux

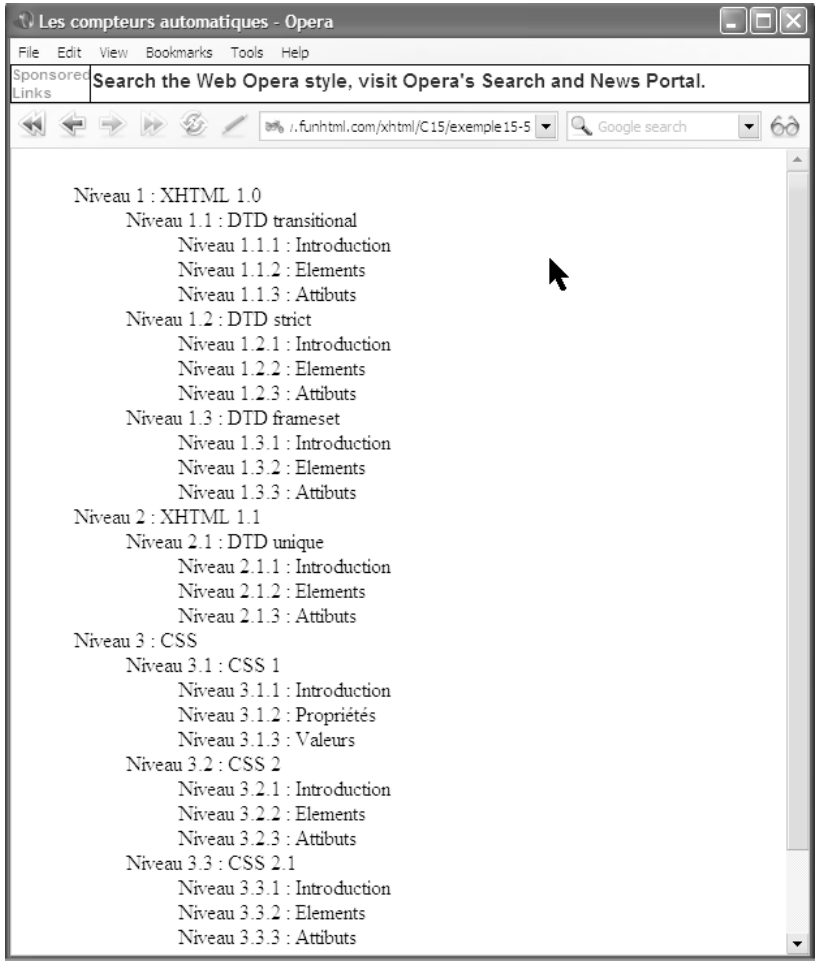
```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Les compteurs automatiques</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="Listes">
   <!-- Compteurs -->
   ol {counter-reset: titres;list-style-type: none;}                ³
   li:before{content: "Niveau " counters(titres,".",decimal) " :
     ➥" ·  ;counter-increment: titres 1;  » }
  </style>
 </head>
 <body>
  <ol> ¿
   <li> XHTML 1.0
     <ol>  ´
     <li> DTD transitional
      <ol>  ²
       <li>Introduction</li>
       <li>Éléments</li>
```

```
    <li>Attributs</li>
   </ol>
  </li>
   <li> DTD strict
   <ol>
    <li>Introduction</li>
    <li>Éléments</li>
    <li>Attributs</li>
   </ol>
  </li>
   <li> DTD frameset
   <ol>
    <li>Introduction</li>
    <li>Éléments</li>
    <li>Attributs</li>
   </ol>
  </li>
 </ol>
</li>
<li> XHTML 1.1
 <ol>
   <li> DTD unique
   <ol>
    <li>Introduction</li>
    <li>Éléments</li>
    <li>Attributs</li>
   </ol>
  </li>
  </ol>
</li>
 <li> CSS
 <ol>
  <li> CSS 1
   <ol>
    <li>Introduction</li>
    <li>Propriétés</li>
    <li>Valeurs</li>
   </ol>
  </li>
   <li> CSS 2
   <ol>
    <li>Introduction</li>
    <li>Éléments</li>
    <li>Attributs</li>
   </ol>
  </li>
```

```
      <li> CSS 2.1
      <ol>
        <li>Introduction</li>
        <li>Éléments</li>
       <li>Attributs</li>
      </ol>
     </li>
     </ol>
    </li>
   </ol>
</body>
</html>
```

La figure 15-6 présente le résultat obtenu pour notre liste à trois niveaux d'imbrication.

Figure 15-6
Numérotation automatique sur trois niveaux ayant un style unique, avec la fonction counters()



Les listes à puces

En XHTML, la disparition des attributs de présentation des listes à puce oblige, et c'est une bonne chose, à utiliser des propriétés CSS pour diversifier les puces affichées dans les listes créées avec l'élément `<ul>`. L'unique puce disponible par défaut est un disque noir mais il est possible de personnaliser les puces grâce à la propriété `list-style-type` déjà utilisée pour les listes numérotées. Les puces prennent toujours la couleur du texte qui les suit, tel que cela est défini par la propriété `color`.

Les puces prédéfinies

Pour les listes à puces, la syntaxe de la propriété `list-style-type` est simplifiée et se résume à ceci :

```
list-style-type:disc | circle | square | none | inherit
```

- `disc` : la puce est un disque plein (c'est la valeur par défaut) ;
- `circle` : la puce est un cercle ;
- `square` : la puce est un carré plein ;
- `none` : pas de puce.

L'exemple 15-6 montre la création de divers styles pour les différents niveaux de listes à puces imbriquées sur deux niveaux. Tous les items de premier niveau disposent d'une puce carrée définie par la valeur `square` de la propriété `list-style-type` pour les éléments `<ul>` (repère ³). S'agissant de définir une puce circulaire pour les items de second niveau, il faut utiliser le sélecteur `ul ul` et la valeur `circle` pour la même propriété (repère). Des couleurs de fonds et des tailles de polices différentes sont également définies pour chacun des sélecteurs afin de mettre en évidence les imbrications.

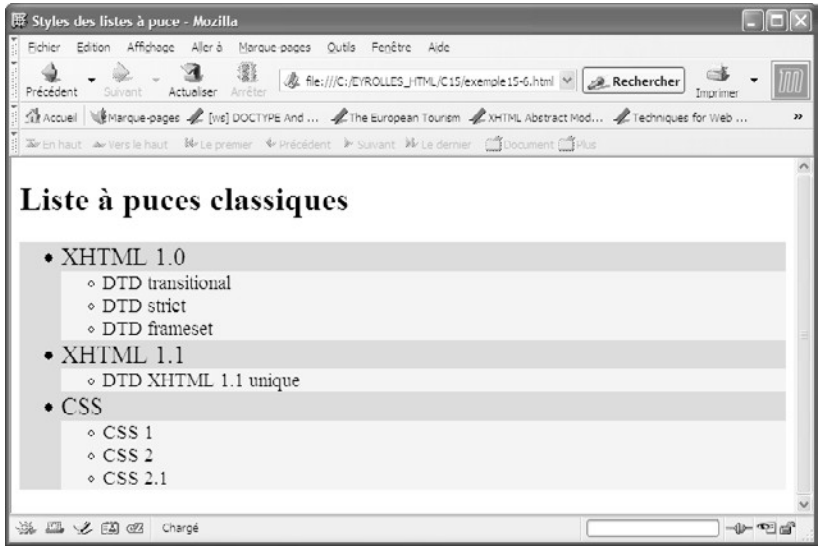
Exemple 15-6. Listes à puces classiques

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <title>Styles des listes à puces </title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <style type="text/css" title="Les listes">
  ul{font-size:1.5em;background-color:#CCC;list-style-type:square;} ³
  ul ul{font-size:0.8em;background-color:#EEE;list-style-type:circle;}
  </style>
 </head>
 <body>
  <h1>Listes à puces classiques</h1>
  <ul>
   <li> XHTML 1.0
```

```
    <ul>
     <li> DTD transitional </li>
     <li> DTD strict </li>
     <li> DTD frameset </p></li>
    </ul>
   </li>
   <li> XHTML 1.1
    <ul>
     <li> DTD XHTML 1.1 unique</li>
    </ul>
   </li>
   <li> CSS
    <ul>
     <li> CSS 1 </li>
     <li> CSS 2 </li>
     <li> CSS 2.1</li>
    </ul>
   </li>
  </ul>
 </body>
</html>
```

La figure 15-7 présente le résultat obtenu.

Figure 15-7
Une liste imbriquée à puces classiques



La propriété `list-style-position` peut également être mise à profit pour les listes à puces. On peut ainsi déterminer si l'alignement des puces des items figurera en retrait (avec la valeur `outside`) ou non (avec la valeur `inside`).

Les puces graphiques

La pauvreté de l'éventail des puces disponibles est telle que la mise en œuvre de puces graphiques personnalisées s'impose vite. Ce ne sont rien d'autres que des images de petite taille (souvent des icônes de type GIF) nous permettant d'utiliser des motifs et des couleurs plus variés. Cette création de style s'effectue avec la propriété `list-style-image`, dont la syntaxe est :

```
list-style-image :<uri>|none|inherit
```

Elle peut être appliquée aux éléments `<ul>` et `<ol>` et à tous ceux dont la propriété `display` a la valeur `list-item` (nous y reviendrons plus en détail dans ce chapitre).

Le paramètre `<uri>` donne l'adresse relative ou absolue de l'image qui va servir de puce. Il est recommandé de toujours faire suivre la propriété `list-style-image` de la propriété `list-style-type`, ce qui peut paraître a priori inutile, alors que cela fournit une solution de remplacement si l'image est introuvable ou corrompue. Dans tous les cas, si l'image est accessible, la seconde définition est ignorée.

Dans l'exemple 15-7, nous créons une liste à puces à deux niveaux (premier niveau, repère ¶, et second niveau, repère º). Le sélecteur du premier niveau est `ul`, et il est mis en évidence par une taille de caractères de 1,5 cm (repère ³) puis par une puce constituée par l'image d'un drapeau (repère ·), et enfin par une définition de style de puce ronde en remplacement éventuel de l'image (repère »). Le sélecteur du second niveau est `ul ul` pour lequel la taille de la police est plus petite (repère ¿). L'image de la puce est différente de celle du niveau supérieur (repère ´) et le style de remplacement est une puce classique carrée (repère ²).

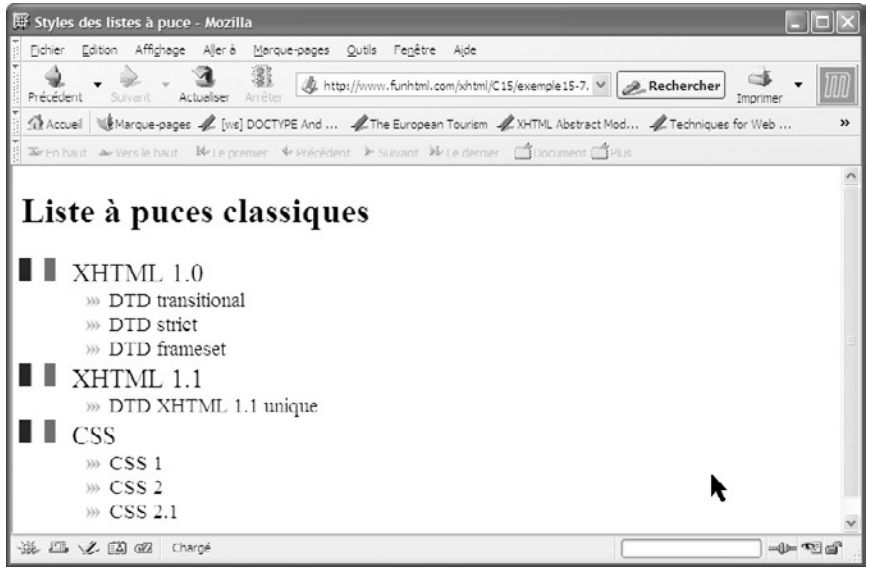
Exemple 15-7. Les listes à puces graphiques

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <title>Styles des listes à puces </title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <style type="text/css" title="Les listes">
  ul{font-size:1.5em ³ ;list-style-image:url(../images/drapeaufr.gif) · ;
    ➥list-style-type:disc; » }
  ul ul{font-size:0.8em ¿ ;list-style-image:url(../images/fleche.gif) ´ ;
    ➥list-style-type:square; ² }
  </style>
 </head>
 <body>
  <h1>Listes à puces classiques</h1>
  <ul> ¶
   <li> &amp;nbsp;XHTML 1.0
     <ul> º
```

```
    <li> DTD transitional </li>
    <li> DTD strict </li>
    <li> DTD frameset </p></li>
   </ul>
  </li>
  <li>  XHTML 1.1
   <ul>  °
    <li> DTD XHTML 1.1 unique</li>
   </ul>
  </li>
  <li>  CSS
   <ul>  °
    <li> CSS 1 </li>
    <li> CSS 2 </li>
    <li> CSS 2.1</li>
   </ul>
  </li>
 </ul>
</body>
</html>
```

La figure 15-8 montre le résultat obtenu. On notera encore que la taille des images utilisées doit être préalablement adaptée au contexte, car il n'est pas possible d'intervenir sur ses dimensions avec une propriété CSS.

Figure 15-8
Listes imbriquées avec puces graphiques



Les listes mixtes

Une liste mixte est une liste à plusieurs niveaux dans laquelle on définit par exemple des puces graphiques pour le premier niveau et des puces classiques pour le second. En reprenant le code XHTML de l'exemple 15-7 en vue de créer ce genre de liste, nous pourrions définir les styles suivants :

```
<style type="text/css" title="Les listes">
 ul{font-size:1.5em;list-style-image:url(../images/drapeaufr.gif)  ³  ;
  ➥list-style-type:disc;}
 ul ul{font-size:0.8em;list-style-type:square;  ·  }
</style>
```

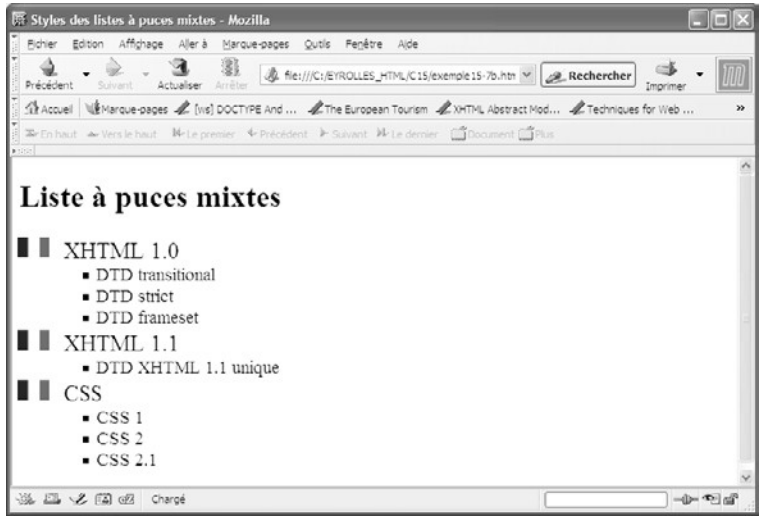
Nous pourrions donc nous attendre à ce que les items de premier niveau soient précédés d'une image GIF (repère ³) et ceux de second niveau d'une puce ordinaire carrée (repère ·). Or, il n'en est rien, et les deux niveaux sont affichés avec la puce graphique de l'image « drapeaufr.gif ». Cela provient de ce que la propriété `list-style-image` est héritée par les éléments enfants, même après la définition d'un style différent par la propriété `list-style-type` dans le sélecteur `ul ul` (repère ·).

Pour pallier cet inconvénient, il faut définir explicitement l'absence d'image pour le second niveau en écrivant les styles suivants dans lesquels la propriété `list-style-image` a la valeur `none` (repère ³), le reste étant inchangé :

```
<style type="text/css" title="Les listes">
 ul{font-size:1.5em;list-style-image:url(../images/drapeaufr.gif);
  ➥list-style-type:disc;}
 ul ul{font-size:0.8em;list-style-image:none  ³  ; list-style-type:square;}
 </style>
```

Dans ce cas seulement, nous obtenons le résultat présenté à la figure 15-9.

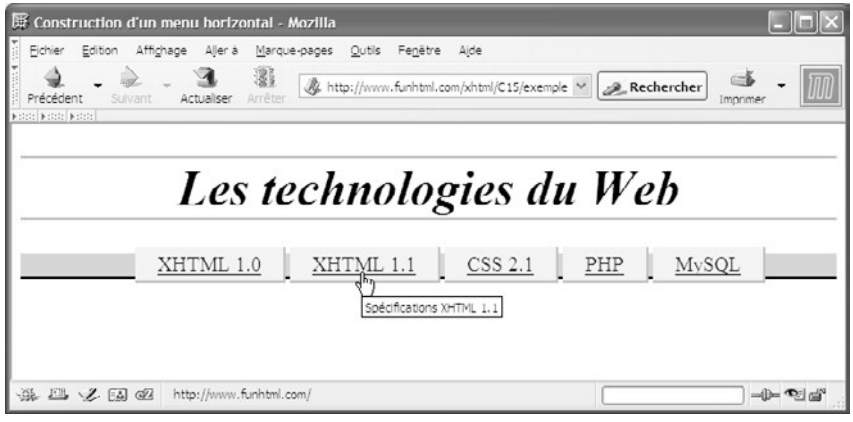
Figure 15-9
Les listes mixtes



Affichage des listes en ligne

Les éléments XHTML de liste <ol> et <ul> employés précédemment sont tous affichés sous forme d'un item par ligne avec un comportement identique aux éléments de type bloc. Il est possible de modifier ce comportement en utilisant la propriété display et en lui attribuant la valeur inline. Dans ce cas, les items d'une liste s'affichent sur la même ligne les uns derrière les autres comme les mots d'un texte. Nous allons utiliser cette technique pour créer un menu horizontal tel que celui présenté à la figure 15-10.

Figure 15-10
Création d'un menu horizontal



L'exemple 15-8 contient le code de création d'un tel menu. La structure est constituée à partir d'une liste ordonnée <ol> (repère) contenant cinq items <li> (repères , , , et) renfermant chacun un lien, comme il se doit dans un menu. Pour le sélecteur ol nous définissons la propriété text-align qui permet le centrage du contenu de la liste dans la page, quelle que soit la largeur de la fenêtre (repère ³). Pour créer un cadre général aux items, nous définissons également une bordure basse (repère ·), une couleur de fond pour ol (repère ») et nous annulons la marge gauche inhérente aux listes (repère ¿). C'est pour le sélecteur li que nous pouvons créer tous les styles avec lesquels nous obtenons le menu horizontal. La propriété display qui prend la valeur inline permet d'obtenir l'affichage des items sur une seule ligne (repère ´) et l'attribution de la valeur none à la propriété list-style-type supprime toute numérotation (repère ²). Les propriétés font-size (repère ¶), background-color (repère º) et padding (repère ¾) concourent à améliorer la présentation du menu mais ne sont pas nécessaires. Il en va de même de la définition des bordures droites (repère µ) et basses (repère ¸) pour chaque élément du menu, qui permettent de donner un aspect de relief à chaque occurrence. La feuille de style se termine par les définitions facultatives de la couleur des liens <a> (repère ¹) et de la présentation du titre principal <h1> qui joue le rôle de bandeau en haut de la page (repère).

Exemple 15-8. Création d'un menu horizontal à partir d'une liste

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <title>Construction d'un menu horizontal</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <style type="text/css" title="Menu">
   ol{text-align: center ³ ; background-color:#FC0 ;border-bottom: 2px
    ➥solid #000» ;margin-left: 0; ¿ }
   li{display:inline ´ ;list-style-type: none ² ; font-size:18px ¶ ;
    ➥background-color:#EE; padding:5px 20px 5px 20px¾ ; border-right:2px solid
    ➥#AAA ;border-bottom:2px solid #AAA; }
   li a{color:navy; ¹ }
   h1{text-align:center; border-top:2px solid #AAA; border-bottom: 2px solid #AAA;
    ➥font-size:44px; font-style:oblique;}
  </style>
 </head>
 <body>
  <h1>Les technologies du Web</h1>
  <ol>
   <li><a href="http://www.w3.org" tabindex="1" accesskey="A" title="Spécifications
    ➥XHTML 1.0" >XHTML 1.0</a></li>
   <li><a href="http://www.funhtml.com" tabindex="2" accesskey="B"
    ➥title="Spécifications XHTML 1.1">XHTML 1.1</a></li>
   <li><a href="http://www.w3.org/TR/CSS21/" tabindex="3" accesskey="D"
    ➥title="Spécifications CSS 2.1">CSS 2.1</a></li>
   <li><a href="http://www.php.net" tabindex="4" accesskey="G" title="PHP">PHP
    ➥</a></li>
   <li><a href="http://www.mysql.org" tabindex="5" accesskey="H"
    ➥title="MySQL">MySQL</a></li>
  </ol>
 </body>
</html>
```

L'aspect obtenu présenté à la figure 15-10 est similaire dans presque tous les navigateurs, à l'exception de Internet Explorer dans lequel la séparation entre chaque item n'existe pas. Le code de l'exemple 15-8 peut facilement être réutilisé. En effet, il suffit d'ajouter des éléments `<li>` pour l'enrichir et obtenir par exemple le résultat présenté à la figure 15-11 sans effectuer aucune modification dans les définitions de styles.

Du point de vue structurel, il serait parfaitement possible de remplacer l'élément `<ol>` par `<ul>`, la liste n'ayant aucun caractère de numérotation.

Figure 15-11
Adaptation du menu horizontal

Affichage d'éléments divers sous forme de liste

Tout comme la propriété `display` permet d'afficher des éléments `<ol>` ou `<ul>` comme des éléments de type en ligne, il est possible de l'utiliser pour afficher sous forme de liste des éléments qui ne le sont pas habituellement. Pour réaliser ce type d'affichage, il faut définir pour l'élément choisi la propriété `display` avec la valeur `list-item`. Nous pouvons ensuite définir des styles spécifiques aux listes pour ces éléments. Cette méthode s'applique bien sûr davantage à des éléments XML dont le rôle est purement sémantique qu'à des éléments XHTML qui ont déjà une signification prédéfinie. L'exemple 15-9 en est une première illustration, qui permet l'affichage de cinq éléments `<h2>` (repères ¸, ¹, et) sous forme d'une liste dont l'aspect est celui d'un menu vertical. Pour parvenir à ce résultat, nous définissons pour le sélecteur `h2` la propriété `display` avec la valeur `list-item` (repère ³). Tous les contenus des éléments `<h2>` s'affichant dès lors sous forme de liste, nous pouvons ensuite leurs appliquer les propriétés spécifiques `list-style-image` (repère ·), `list-style-type` (repère ») et `list-style-position` (repère ¿). Les définitions de styles suivantes (repères´, ², ¶ et º) créent une présentation proche de celle obtenue dans le menu horizontal de l'exemple 15-8. L'inclusion de tous les éléments `<h2>` (repères ¸, ¹, et) dans une division `<div>` (repère µ) permet de positionner l'ensemble sur la partie gauche de la page à l'aide des propriétés `position` et `width` (repère ¾).

Exemple 15-9. Affichage d'éléments divers sous forme de liste

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <title>Construction d'un menu vertical</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
```

```
 <style type="text/css" title="Menu">
  h2{display:list-item ³ ;list-style-image:url(fleche.gif) · ;
   ➥list-style-type:disc » ;list-style-position:inside ¿ ;font-size:18px;
   ➥background-color:#EEE;padding:5px 20px 5px 20px;border-bottom:2px solid #AAA;}
  h1{text-align:center;border-top:2px solid #AAA;border-bottom: 2px solid #AAA;
    ➥font-size:44px;font-style:oblique; }
  a{color:navy;}
  div{position:absolute;width:15%;}
 </style>
</head>
<body>
 <h1>Les technologies du Web</h1>
 <div>
  <h2><a href="http://www.w3.org" tabindex="1" accesskey="A" title="Spécifications
    ➥XHTML 1.0" >XHTML 1.0</a></h2>
  <h2><a href="http://www.funhtml.com" tabindex="2" accesskey="B"
    ➥title="Spécifications XHTML 1.1">XHTML 1.1</a></h2>
  <h2><a href="http://www.w3.org/TR/CSS21/" tabindex="3" accesskey="D"
    ➥title="Spécifications CSS 2.1">CSS 2.1</a></h2>
  <h2><a href="http://www.php.net" tabindex="4" accesskey="G"
    ➥title="Spécifications CSS 2.1">PHP</a></h2>
  <h2><a href="http://www.mysql.org" tabindex="5" accesskey="H"
    ➥title="Spécifications CSS 2.1">MySQL</a></h2>
 </div>
</body>
</html>
```

La figure 15-12 montre le résultat obtenu.

Figure 15-12

Création d'une liste à partir d'éléments <h2>

Dans la même optique, il est possible de réaliser une liste imbriquée sur plusieurs niveaux à partir des éléments <h1>, <h2> et <h3> en définissant pour chacun d'eux (repères ³ , ¿ et º), dans l'exemple 15-10, la propriété display avec la valeur list-item comme précédemment. Nous définissons ensuite les propriétés list-style-image , list-style-type et list-style-position (repères · , » , ´ , ² et ¾) pour créer et positionner des puces différentes pour chaque niveau. En créant des marges gauches (repères ¶ et ¸), on peut afficher chaque niveau en retrait du niveau précédent.

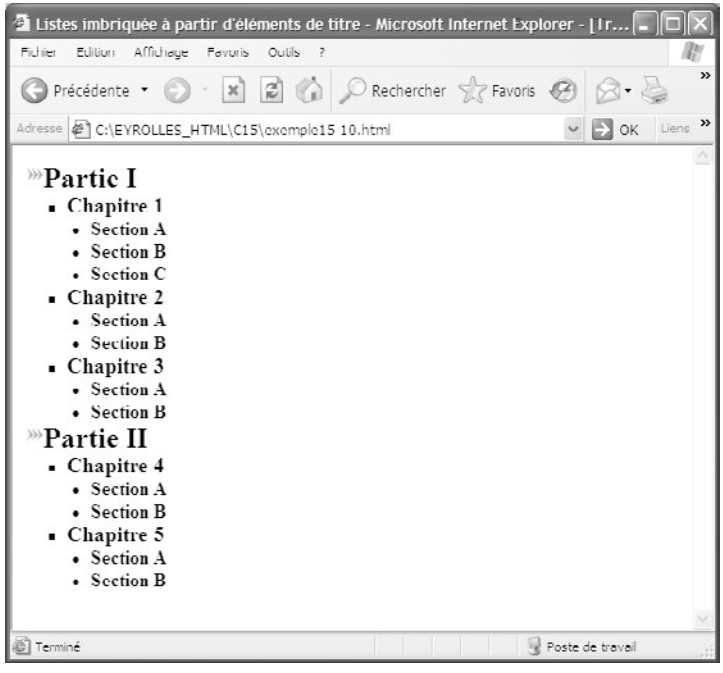
Exemple 15-10. Création d'une liste imbriquée à partir d'éléments de titre

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Listes imbriquées à partir d'éléments de titre</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" >
   h1{display:list-item ³ ;list-style-image:url(../images/fleche.gif) · ;
    ➥list-style-position: inside » ; font-size :24px;}
   h2{ display:list-item ¿ ;list-style-type:square ´ ; list-style-position: inside ² ;
    ➥margin-left: 20px ¶ ; font-size:18px;}
   h3{ display:list-item º ;list-style-type:disc ¾ ; list-style-position: inside µ ;
    ➥margin-left: 40px ¸ ; font-size:16px;}
  </style>
 </head>
 <body>
  <h1> Partie I </h1>
   <h2> Chapitre 1 </h2>
    <h3>Section A</h3>
    <h3>Section B</h3>
    <h3>Section C</h3>
   <h2> Chapitre 2 </h2>
    <h3>Section A</h3>
    <h3>Section B</h3>
   <h2> Chapitre 3 </h2>
    <h3>Section A</h3>
    <h3>Section B</h3>
  <h1> Partie II </h1>
   <h2> Chapitre 4</h2>
    <h3>Section A</h3>
    <h3>Section B</h3>
   <h2> Chapitre 5 </h2>
```

```
    <h3>Section A</h3>
    <h3>Section B</h3>
  </body>
</html>
```

Figure 15-13

Création d'une liste imbriquée à partir de différents niveaux de titre



Notons pour finir que l'utilisation d'autres éléments XHTML aurait le même effet à condition de leur attribuer la propriété `display` avec la valeur `list-item` .

Exercices

Exercice 1 : Créez deux listes et définissez explicitement les styles par défaut des éléments `<ol>` et `<ul>`.

Exercice 2 : Créez une liste numérotée en caractères latins majuscules.

Exercice 3 : Créez une liste sur deux niveaux, numérotée en caractères latins pour le premier et en alphabet grec pour le second. Attribuez une fonte et une taille différentes à chaque niveau.

Exercice 4 : Créez une liste non ordonnée dotée de puces carrées.

Exercice 5 : Créez une liste numérotée en chiffres romains pour le premier niveau et à puces graphiques pour le second.

Exercice 6 : Créez une liste de définition de termes et appliquez-lui les propriétés de numérotation des éléments <ol>.

Exercice 7 : Pour cet exercice, téléchargez et installez tout d'abord le navigateur Opera. Utilisez ensuite des compteurs pour numéroter une liste de format papier, leurs dimensions allant de A0 à A5 (pour mémoire le format A4 correspond à 297 ¥ 210 mm).

Exercice 8 : Transformez le menu horizontal de l'exemple 15-8 en menu vertical en conservant les éléments <ol>.

16

Les médias écrits

Les navigateurs sont les moyens les plus courants pour visualiser les pages web, mais s'ils représentent l'immense majorité des moyens mis en œuvre, ce ne sont pas les seuls. Il est en effet possible d'afficher une page sur des médias très divers, aussi bien des terminaux portables ayant des écrans de petite taille que des médias dits « paginés ». Un navigateur est en effet un média continu, car la page est unique et sa hauteur peut être très grande (en réalité, il n'y a pas de limite à cette hauteur), et parce qu'il suffit à l'utilisateur de la faire défiler, d'où l'introduction systématique des roulettes sur les souris ces dernières années. Dans les médias paginés, le contenu de la page n'est plus visualisé de façon continue mais sous forme de feuilles qui se succèdent. Il s'agit par exemple des imprimantes ou des rétroprojecteurs qui présentent un site sous forme de diapositives comme le fait PowerPoint. Nous nous intéressons particulièrement ici à la manière de créer des styles en vue d'imprimer le contenu d'un site. Compte tenu du peu de prise en charge par les différents navigateurs de certaines possibilités que nous allons aborder, il conviendra de se montrer circonspect dans leur utilisation. Le projet de recommandation CSS 3 devrait cependant reprendre et étendre les possibilités d'impression et il nous semble utile de s'y préparer.

Adapter les styles à l'impression

Cibler un média précis

Plusieurs moyens sont disponibles pour qu'un ensemble de styles s'applique à un média donné et non à d'autres. Si ces styles sont inclus dans un fichier externe ayant l'extension .css, nous pouvons les importer en précisant dans l'élément <link /> le média cible en utilisant l'attribut media de la manière suivante :

```
<link rel="stylesheet" type="text/css" media="screen" href="style_ecran.css" />
<link rel="stylesheet" typ e="text/css" media="print" href="style_imprime.css" />
```

Lors de l'impression, seuls les styles contenus dans le fichier imprime.css sont appliqués.

Il est également possible de définir des styles spécifiques à un média en les écrivant dans plusieurs éléments <style> munis également d'un attribut media adapté à la circonstance, par exemple :

```
<style type="text/css" media="screen">
 body {background-color: yellow; color:blue;}
</style>
<style type="text/css" media="print">
 body {background-color: white; color: black;}
</style>
```

Dans ce cas, le premier élément <style> (repère ³) est adapté aux écrans et le texte de la page est écrit en bleu sur fond jaune (repère). En revanche, le second élément (repère ») est adapté aux imprimantes et va afficher un texte noir sur fond blanc (repère ¿). L'ordre d'apparition de ces deux éléments n'a ici aucune importance.

La distinction entre les différents médias peut aussi être opérée à l'intérieur même d'un unique élément <style>, en divisant son contenu en plusieurs sections, chacune d'elles étant introduite par la directive @media suivie du nom du média, puis de l'ensemble des styles inclus entre accolades.

Le code suivant en donne un exemple :

```
<style type="text/css" >
 @media screen{
  body{background-color: blue; font-family: Arial; color: white;}
 }
 @media print{
  body{background-color: #FFF; font-family: Times; color: #000;}
 }
</style>
```

Dans ce cas, les styles créés dans la première directive @media (repère ³) permettent d'afficher sur l'écran un texte blanc sur fond bleu avec une police Arial. Les styles définis dans la seconde directive (repère ·) permettent quant à eux d'afficher un texte noir sur fond blanc dans une police Times.

Adaptation des styles

Il ne s'agit pas ici d'envisager la simple impression d'une page web en tant que copie d'écran, ce que chaque visiteur peut faire à partir de son navigateur, mais bien plutôt de définir les styles qui permettent une impression correcte des pages ayant un contenu textuel important, et de limiter l'impression à ce contenu réellement informatif, en omettant volontairement les parties non signifiantes comme les menus ou les bandeaux graphiques ou publicitaires présents en haut des pages. Le but est donc de permettre aux visiteurs de récupérer la trace écrite d'un contenu.

La plupart des propriétés que nous avons étudiées sont applicables à l'impression d'un document, mais la gestion des couleurs n'étant pas assurée coté client, il est préférable de créer des styles appropriés à une sortie sur une imprimante noir et blanc, et par exemple de définir des nuances avec des niveaux de gris plutôt qu'avec des couleurs comme pour une sortie sur écran. De même, les polices de la famille `sans-serif` comme Arial, définies avec la propriété `font-family`, sont réputées être plus lisibles sur un écran que les polices de type `serif` comme Times, qui sont réputées plus lisibles sur un support papier. Nous pourrons donc définir différemment la propriété `font-family` selon qu'elle s'applique à l'écran ou à l'impression.

L'exemple 16-1 donne une illustration de ces adaptations en ne redéfinissant des styles qu'en fonction du média cible et en ne permettant de réaliser que l'impression du contenu, en éliminant donc les éléments d'organisation de la page telle qu'elle est vue habituellement.

La page web concernée, représentée à la figure 16-1, comporte trois zones quand elle est affichée à l'écran.

Chacune de ces zones est comprise dans un élément `<div>` (repères º, ¾ et µ) positionné à l'aide des styles que nous avons étudiés au chapitre 13 et qui sont définis dans le bloc `@media screen` (repère ³). En revanche, l'utilisateur qui veut imprimer le contenu du site n'a pas besoin de retrouver sur chaque page le bandeau situé en haut de l'écran, ni le menu situé sur la gauche. Nous devons donc les éliminer dans le bloc `@media print` (repère ·) dédié aux styles d'impression. Pour cela, nous utilisons la propriété `display` en lui affectant la valeur `none` (repères » et ¿) pour les deux premiers éléments `<div>` (repères º et ¾). Nous adaptons également la police de caractères en définissant la propriété `font-family` avec la valeur `Times serif` (repère ´) et la propriété `font-size` en utilisant l'unité `pt` adaptée aux imprimantes pour les éléments `<p>` et `<h2>` (repères ² et ¶). De plus, l'impression en noir sur fond blanc ou gris est définie explicitement par les valeurs des propriétés `color` et `background-color` (repères ´ et ¶).

Exemple 16-1. Adaptation des styles à l'impression

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <title>La référence des éléments XHTML 1.1</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <style type="text/css" >
   @media screen{
    body {background-color: #FFA; font-family: Arial;}
    div#tete {position: fixed ; left: 0px; top: 0px; width: 100%; height: 15%;
      ➥ background-color: rgb(0,0,153); z-index: 5; margin:0; }
    div#menu {position: fixed ; width: 20%; left: 0px; top: 15%; background-color:
      ➥ rgb(255,102,51); color: white;}
    div#contenu {position: absolute ; idth: 80%; left: 20%; top: 17%;
      ➥ background-color: #FFA; color: black;}
    p {margin-left: 20px;}
    div h1 {font-size: 50px; font-style: italic; color: rgb(255,102,51); margin-top:
      ➥ 0px; margin-left: 200px;}
   }
   @media print{
    div#tete {display: none;}
    div#menu {display: none;}
    div#contenu {width: 170mm; background-color: white; color: black; margin-left:
      ➥ 20mm; text-align: justify; font-family: Times serif;}
    p {margin-left: 15mm; font-size: 14pt;}
    h2{font-size:30pt; text-align:center; margin-bottom: 30mm; background-color:
      ➥ #EEE;}
   }
  </style>
 </head>
 <body>
  <div id="tete">
   <h1>funhtml.com&amp;nbsp;&amp;nbsp;Référence XHTML 1.1</h1>
  </div>
  <div id="menu">
   <ul>
    <li> !DOCTYPE</li>
    <li>&amp;lt;html&amp;gt;</li>
    <li>&amp;lt;head&amp;gt;</li>
    <li>&amp;lt;body&amp;gt;</li>
    <li>&amp;lt;a&amp;gt;</li>
    <li>&amp;lt;abbr&amp;gt;</li>
```

```
    <li>&lt;acronym&gt;</li>
   <!--  Suite des éléments XHTML-->
    </ul>
  </div>
  <div id="contenu">µ
   <h2>Le langage XHTML</h2>
   <p>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
     ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas dixitque
     ➥ Deus fiat lux et facta est lux et vidit Deus lucem . . .</p>
  </div>
 </body>
</html>
```

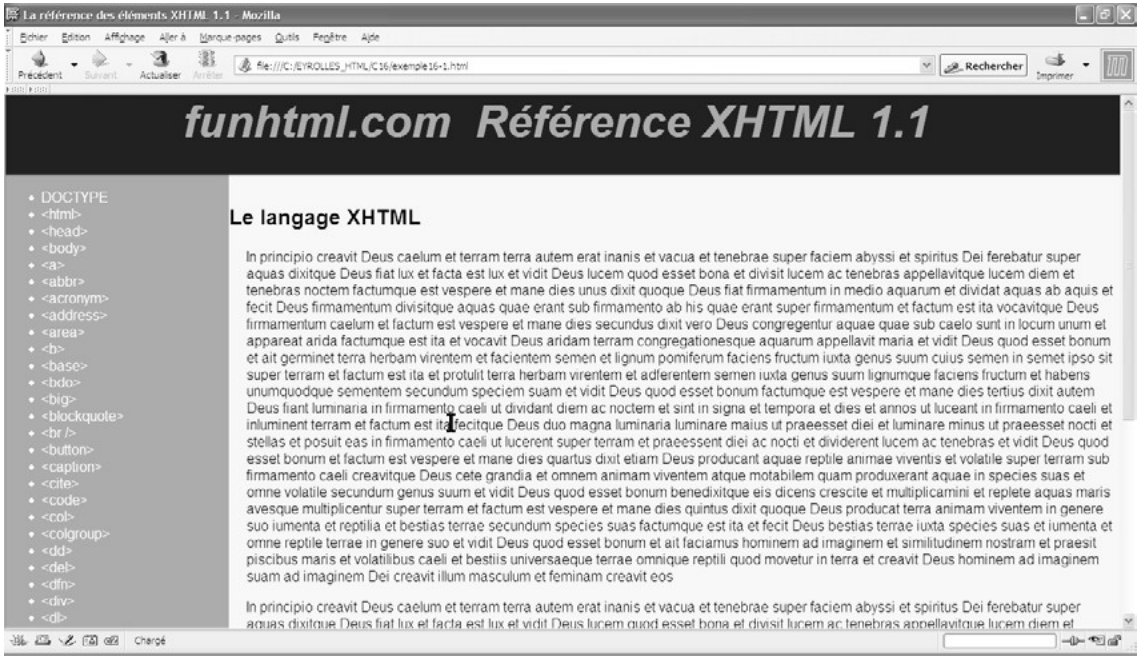


Figure 16-1

La page d'origine dans un navigateur

La figure 16-2 montre l'aperçu avant impression obtenu dans Mozilla ou FireFox. Les résultats obtenus dans les navigateurs Opera ou Internet Explorer sont équivalents, mais chaque navigateur crée des marges différentes dans les paramètres d'impression, et nous ne pouvons pas intervenir sur ces valeurs. Un texte donné n'occupe donc pas le même espace quand il est imprimé, à partir de différents navigateurs, à moins de prévenir le visiteur de mettre manuellement toutes les marges à la valeur 0.

Figure 16-2
Adaptation des styles à l'impression

Créer un en-tête commun aux pages

L'exemple précédent élimine délibérément l'en-tête et le menu de la page, mais vous pouvez trouver utile de faire apparaître un en-tête commun à toutes les pages imprimées, qu'il soit publicitaire ou non, quand le contenu est long. Pour parvenir à ce résultat, nous devons modifier dans le bloc `@media print` (repère ³) les styles existants en créant tout d'abord un espace en haut de chaque page, en définissant une marge haute pour l'élément `<body>` dans laquelle s'inscrira le bandeau (repère ·). Ensuite, nous positionnons le deuxième élément `<div>` de manière fixe à l'aide des propriétés `position`, `left` et `top` (repère »). Les styles suivants restent inchangés (repères ¿ , ´ , ² et ¶).

Exemple 16-2. Création d'un bandeau sur chaque page

```
 <?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
 <head>
  <title>La référence des éléments XHTML 1.1</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <style type="text/css" >
   @media screen{
    body {background-color: #FFA; font-family: Arial;}
```

```
  div#tete {position: fixed ; left: 0px; top: 0px; width: 100%; height: 15%;
    ➥ background-color: rgb(0,0,153); z-index: 5; margin:0; }
  div#menu {position: fixed ; width: 20%; left: 0px; top: 15%; background-color:
    ➥ rgb(255,102,51); color: white;}
  div#contenu {position: bsolute ; width: 80%; left: 20%; top: 17%;
    ➥ background-color: #FFA; color: black;}
  p {margin-left: 20px;}
  div h1 {font-size: 50px; font-style: italic; color: rgb(255,102,51); margin-top:
    ➥ 0px; margin-left: 200px;}
 }
 @media print{
  body{margin-top: 20mm;}
  div#tete {position: fixed; left: 0px; top: 0px; width: 100%; height: 20mm;
    ➥ background-color: #CCC; font-family: Arial; font-size: 16pt;}
  div#menu {display: none;}
  div#contenu {background-color: white; color: black; margin-left: 20mm;
    ➥ text-align: justify;}
  p {margin-left: 15mm;}
  h2{font-size: 30pt; font-family: Times; text-align: center; margin-bottom: 30mm;
    ➥ background-color: #EEE;}
 }
 </style>
</head>
<body>
 <div id="tete">
  <h1>funhtml.com&amp;nbsp;&amp;nbsp;Référence XHTML 1.1</h1>
 </div>
 <div id="menu">
  <ul>
    <li>DOCTYPE</li>
    <li>&amp;lt;html&amp;gt;</li>
   <li>&amp;lt;head&amp;gt;</li>
    <li>&amp;lt;body&amp;gt;</li>
    <li>&amp;lt;a&amp;gt;</li>
    <!-- Suite des éléments -->
  </ul>
 </div>
 <div id="contenu">
  <h2>Le langage XHTML</h2>
  <p>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
    ➥ et tenebrae super faciem abyssi et . . .</p>
  <h2>Les styles CSS 2.1 </h2>
  <p>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
    ➥ et tenebrae super faciem abyssi et . . .</p>
 </div>
</body>
</html>
```

La page contenant plusieurs paragraphes inclus dans le troisième élément `<div>` (repères °, ¾ et µ), nous pouvons vérifier dans les aperçus avant impression des pages 1 et 2 présentées aux figures 16-3 et 16-4 que chacune possède bien le même en-tête.

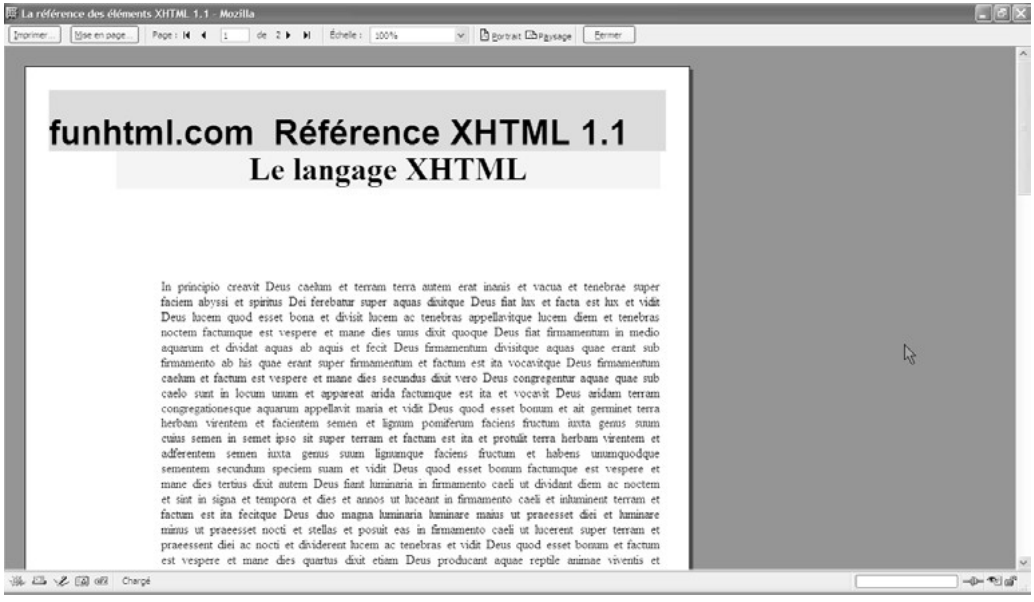


Figure 16-3

La page 1 avec son en-tête

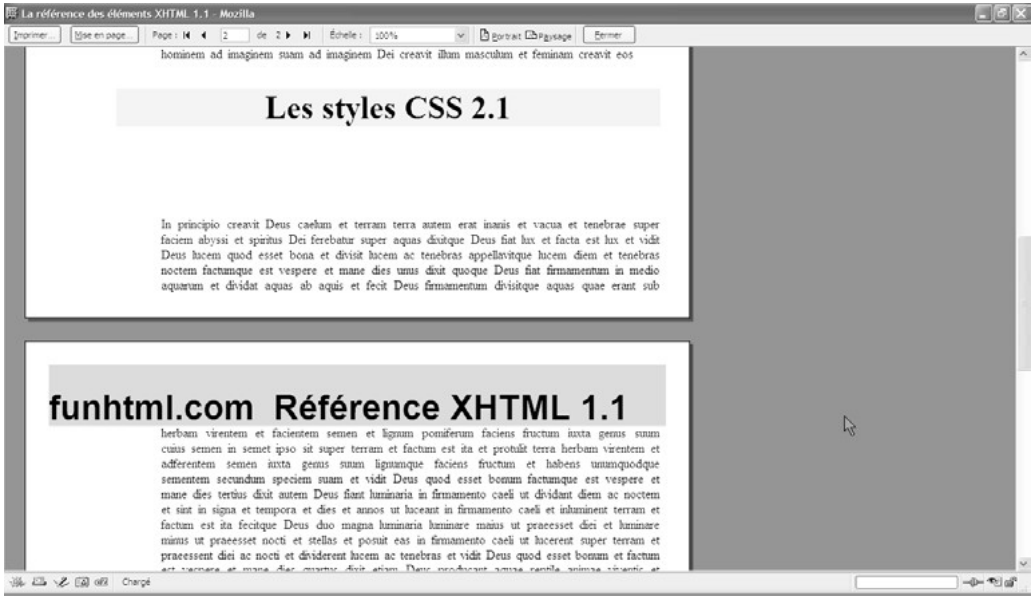


Figure 16-4

La page 2 avec son en-tête

Gestion des sauts de page

Nous pouvons améliorer la présentation d'un contenu imprimé en insérant, comme dans un traitement de texte, des sauts de page uniquement avant et après un élément de bloc, en définissant respectivement les propriétés page-break-before et page–break-after dont la syntaxe commune est la suivante :

```
page-break-before: auto | always | avoid | left | right | inherit
page-break-after: auto | always | avoid | left | right | inherit
```

Voici la signification de leurs paramètres :

- auto : les sauts de page ne sont effectués que si nécessaire ;
- always : le saut de page forcé est toujours autorisé avant ou après un élément quelle que soit sa hauteur ;
- avoid : aucun saut de page forcé n'est autorisé ;
- left : le saut de page n'est autorisé que si l'élément figure dans une page de gauche (page paire ou verso) ;
- right : le saut de page n'est autorisé que si l'élément figure dans une page de droite (page impaire ou recto).

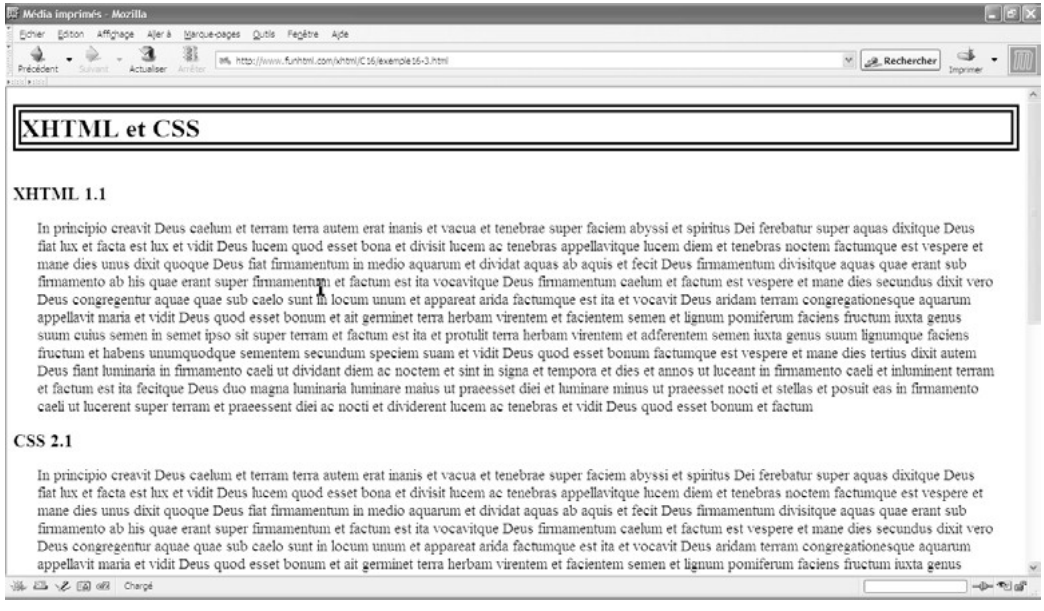


Figure 16-5

La page affichée dans un navigateur

L'exemple 16-3 permet de tester ces propriétés en définissant une mise en page particulière pour un document dans sa version imprimée. La figure 16-5 montre l'aspect de la

page dans un navigateur afin de bien mettre en évidence, par contraste, les différences avec la version imprimée (voir figure 16-6).

Pour la version imprimée, notre objectif est de faire apparaître le titre <h1> (repère ¶) seul au milieu de la première page, le titre <h2> (repère °) et le contenu du paragraphe <p> de la première section (repère ¾) sur une page impaire (la page 3), et chacune des sections suivantes (les titres et les paragraphes) (repères µ et ¸, puis ¹ et) sur une nouvelle page. Pour y parvenir, nous créons des styles propres à l'impression dans un élément <style> spécialisé (repère ³). Pour centrer verticalement le titre <h1> nous lui attribuons une marge haute de 130 mm, et la propriété page-break-after permet d'insérer un saut de page immédiatement après lui (repère »). La propriété page-break-before est appliquée uniquement au premier titre <h2> à l'aide d'un sélecteur d'id (repère °) pour créer le saut de page avant lui, afin qu'il apparaisse avec le premier paragraphe sur la page 3. Pour tous les autres paragraphes, y compris le premier, nous créons un saut de page avec la propriété page-break-after (repère ´) pour que ce qui suit soit imprimé sur une nouvelle page. Afin de ne pas obtenir une page blanche en fin d'impression, nous annulons ce saut de page pour le dernier paragraphe (repère) en définissant la propriété page-break-after avec la valeur avoid (repère ²). Notons que les résultats obtenus présentés en figure 16-6 sont valables aussi bien dans Mozilla, Opera que Internet Explorer, ce qui est assez rare pour être signalé.

Exemple 16-3. Insertion de sauts de pages

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Média imprimés</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" title="impression" media="screen">
   *{font-family: Arial sans-serif;}
   h1 {font-size: 2em; margin-bottom: 40px; border: 10px double;}
   h2 {font-size: 1.4em;}
   p {font-size: 1.2em; margin: 0 30px 0 30px;}
  </style>
  <style type="text/css" title="impression" media="print">    ³
   *{font-family: Times serif;}
   h1 {font-size: 36pt; margin-top: 130mm; text-align: center; page-break-after:
    ➥ always;}»
   h2 {font-size: 20pt;}
   h2#premier {page-break-before: always;}¿
   p {font-size: 12pt; font-style: italic; text-align: justify; page-break-after:
    ➥ always;}´
   p#dernier {page-break-after: avoid;}   ²
  </style>
 </head>
```

```
<body>
  ¶  <h1>XHTML et CSS</h1>
  °  <h2 id="premier">XHTML 1.1</h2>
  ¾ <p>In principio creavit Deus caelum et terram terra autem erat inanis
  ➥ et vacua et tenebrae super faciem abyssi et spiritus . . .</p>
  µ  <h2>CSS 2.1</h2>
  ›  <p> In principio creavit Deus caelum et terram terra autem erat inanis
  ➥ et vacua et tenebrae super faciem abyssi et spiritus . . .    </p>
  ¹  <h2>PHP 5</h2>
     <p id="dernier">In principio creavit Deus caelum et terram terra autem erat
  ➥ inanis et vacua et tenebrae super faciem abyssi et spiritus . . .</p>
 </body>
</html>
```

Figure 16-6

Réalisation des sauts de pages à l'impression

Les sauts de page effectués à l'intérieur d'un élément de bloc sont automatiques quand le contenu est long, la suite du texte étant de fait imprimée sur la page suivante. Pour éviter d'obtenir des coupures qui risquerait de ne faire apparaître que quelques lignes orphelines en haut de la dernière page, CSS2 offre la possibilité de placer des sauts de page à l'intérieur d'un bloc. Cette possibilité reste assez théorique, car pour l'instant elle n'est exploitée, et encore incomplètement, que par Opera 8. Nous la signalons cependant à titre informatif, en espérant que tous les navigateurs la prendront en compte de manière précise dans un avenir proche.

Pour gérer ces sauts de page, nous disposons de plusieurs propriétés qui doivent être utilisées conjointement. Pour autoriser ce type de saut de page, il faut définir la propriété `page-break-inside` dont la syntaxe est la suivante :

```
page-break-inside : auto | avoid |inherit
```

avec la valeur `auto`, la valeur `avoid` interdisant ces sauts.

En complément, la propriété `widows` dont la valeur est un nombre entier (par défaut, elle est de `2`), permet d'indiquer le nombre minimal de lignes d'un bloc qui peuvent être placées

en haut d'une page. Si la coupure automatique d'un texte contenu dans un élément <p> ou par exemple, conduit à faire apparaître moins de lignes que la valeur fixée dans widows alors le paragraphe entier est placé dans la page suivante, créant ainsi un saut de page forcé à la suite de l'élément.

De même, la propriété orphans permet de définir le nombre minimal de lignes d'un bloc qui doivent apparaître en bas d'une page. Si la coupure automatique du texte conduit à afficher un nombre inférieur de lignes, alors tout le paragraphe est imprimé sur la page suivante.

Ces définitions sont celles qui figurent dans la recommandation CSS 2.1 du W3C, mais elles restent pour l'instant lettre morte, et les tests effectués ne fournissent le résultat espéré dans aucun navigateur.

Si nous définissons par exemple les styles suivants :

```
p {orphans: 15; widows: 10; page-break-inside: auto;}
```

et que nous visualisons le résultat de l'aperçu avant impression dans Opera 8, nous obtenons le résultat présenté à la figure 16-7.

Nous remarquons que Opera gère ces propriétés exactement à l'inverse des définitions du W3C. En effet, la propriété widows a la valeur 10 et 10 lignes apparaissent en bas de la première page. De même, la propriété orphans a la valeur 15 et il apparaît 16 lignes sur la deuxième page.

Figure 16-7
Les sauts de pages internes

Les pseudo-classes d'impression

En vue d'améliorer encore la présentation des documents imprimés, les concepteurs de CSS ont défini les pseudo-classes :first, :left et :right, afin d'ajouter des fonctionnalités dignes d'un traitement de texte évolué. Hélas, encore une fois, elles ne sont prises en compte que par Opera 8. Nous ne les citons ici que pour mémoire, leur utilisation n'étant pas recommandée pour l'instant, tant que les autres navigateurs ne les auront pas implémentées.

Elles s'utilisent conjointement à la directive @page de la manière suivante :

- @page:first{Définitions des styles;} pour que les styles ne s'appliquent qu'à la première page imprimée.
- @page:left{Définitions des styles;} pour que les styles s'appliquent aux pages de gauche (les pages paires).
- @page:right{Définitions des styles;} pour que les styles s'appliquent aux pages de droite (les pages impaires).

L'exemple 16-4 utilise ces pseudo-classes pour créer une mise en page particulière qui s'applique à la première page en y créant une marge haute de 100 mm (repère ³), puis des marges gauche et droite différentes selon qu'il s'agit d'une page gauche ou droite, en supposant que les feuilles pourraient être reliées ou agrafées. Pour les pages de gauche (les pages paires), la marge droite est de 40 mm et la marge gauche de 10 mm (repère ·). Pour les pages de droite (les pages impaires), ces valeurs sont inversées (repère »).

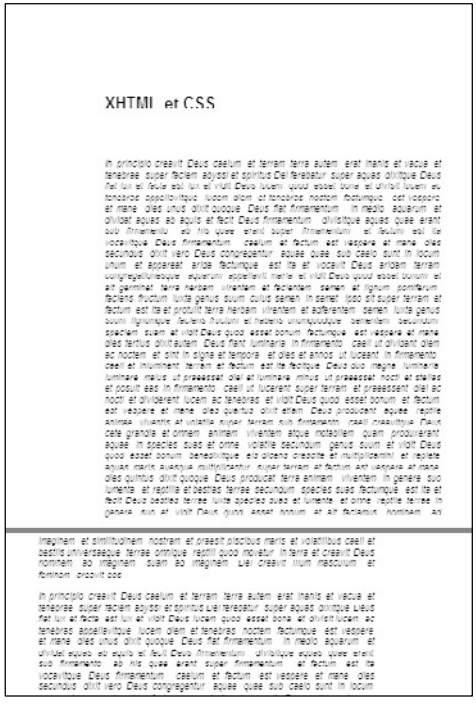
Exemple 16-4. Les pseudo-classes d'impression

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="fr" xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Média imprimés</title>
  <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
  <link rel="shortcut icon" type="images/x-icon" href="../images/favicon.ico" />
  <style type="text/css" media="screen">
   h1 {font-family: Times ; font-size: 2em; margin-bottom: 40px; border: 10px double;}
   p {font-family: Times; font-size: 1.2em; margin: 0 30px 0 30px;}
  </style>
  <style type="text/css" title="impression" media="print">
   @page:first {margin-top: 100mm;}³
   @page:left {margin-right: 40mm; margin-left: 10mm;}·
   @page:right {margin-left: 40mm; margin-right: 10mm;}»
   h1 {font-family: Arial; font-size: 20pt; margin: 0cm 3cm 2cm 0cm;
     ➥ background-color: #EEE;}
```

```
  p {font-family: Arial; font-size: 12pt; font-style: italic; text-align: justify;
    ➥ width: 150mm;}
 </style>
</head>
<body>
 <h1>XHTML et CSS</h1>
 <p>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
    ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas dixitque
    ➥ Deus fiat lux et facta est lux . . . </p>
 <p>In principio creavit Deus caelum et terram terra autem erat inanis et vacua
    ➥ et tenebrae super faciem abyssi et spiritus Dei ferebatur super aquas dixitque
    ➥ Deus fiat lux et facta est lux . . .  </p>
</body>
</html>
```

La figure 16-8 montre le résultat obtenu, qui correspond bien à l'objectif visé, mais seulement dans Opera 8.

Figure 16-8
Utilisation des pseudo-classes d'impression



Exercices

Exercice 1 : Quelles sont les différentes méthodes permettant de créer des styles particuliers pour un média donné ?

Exercice 2 : Créez des styles différents pour l'écran et l'impression en utilisant la directive `@media` pour que le texte soit affiché en jaune sur fond bleu à l'écran, et en noir sur fond blanc à l'impression.

Exercice 3 : Adaptez les polices de caractères et leur taille avec des valeurs différentes pour l'écran et l'impression.

Exercice 4 : Créez un en-tête de page contenant votre nom et votre mail pour qu'il apparaisse en haut de chaque page imprimée mais pas dans un navigateur.

Exercice 5 : Créez un document dont le contenu est inclus dans plusieurs éléments `<div>` et définissez les styles permettant d'obtenir des sauts de page à l'impression après chacun de ces éléments.

Exercice 6 : Dimensionnez et positionnez une page et les paragraphes qu'elle contient pour qu'ils soient imprimés en format paysage.

Partie III

Annexes

Annexe A

Référence des éléments XHTML

Notations

Les attributs en **gras souligné** sont obligatoires

Le symbole | signifie "ou"

Le symbole || signifie "et/ou"

(élément)* signifie 0 ou plusieurs fois l'élément

(élément)? signifie 0 ou une fois l'élément

(élément)+ signifie 1 ou plusieurs fois l'élément

Élément <a>…< /a>

Définition	Lien hypertexte.
Enfants	Texte, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	abbr, acronym, address, b, bdo, big, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.

Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
accesskey	Définit la touche de raccourci clavier à l'aide d'une lettre.
charset	Indique un jeu de caractères (exemple : iso-8859-1).
coords	Crée une zone sensible dans le lien (rectangle, cercle, polygone) en inqiquant des coor-données.
href	Contient l'URL du document cible.
hreflang	Le code de langue du document cible.
onblur	Événement qui se produit quand l'élément perd le focus.
onfocus	Événement qui se produit quand l'élément reçoit le focus.
rel	Type de la relation entre le document et la cible.
rev	Type de la relation entre la cible et le document.
shape	Type de forme de la zone sensible (voir coords).
tabindex	Entier donnant un ordre de tabulation pour l'élément.
target	Indique le nom du cadre dans lequel doit s'afficher la cible du lien (avec la DTD frameset seulement).
type	Type MIME du fichier cible ("text/html" ou "image/gif" par exemple).

Élément <abbr>…< /abbr>

Définition	Abréviation.
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.

Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <acronym>…< /acronym>

Définition	Acronyme.
Enfants	Texte a,abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <address>…< /address>

Définition	
Enfants	Texte a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	blockquote, body, button, dd, del, div, fieldset, form, ins, li, map, noscript, object, td, th.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, événements de base.

Élément <area />

Définition	Définit une zone sensible au clic.
Enfants	Aucun.
Parents	map.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
accesskey	Définit la touche de raccourci clavier à l'aide d'une lettre.
alt	Texte descriptif de la cible du lien.
coords	Liste d'entiers séparés par des virgules définissant la zone.
href	Contient l'URL du document cible.
nohref	"nohref".
onblur	Événement qui se produit quand l'élément perd le focus.
onfocus	Événement qui se produit quand l'élément reçoit le focus.
shape	Définit la forme de la zone (default|rect|circle|poly).
tabindex	Entier donnant un ordre de tabulation pour l'élément.

Élément <b>…< /b>

Définition	Mise en gras du contenu (préférez un style CSS).
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <base />

Définition	Définition de l'adresse de base pour les URL relatives.
Enfants	Aucun.
Parents	head.
Attributs	**Définition et valeurs**
Communs	id.
href	Contient l'URL de base.

Élément <bdo>…< /bdo>

Définition	Indique le sens de lecture du texte.
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, dfn, em, i, img, input, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
dir	Sens de lecture ltr (gauche à droite par défaut) ou rtl (doite à gauche).

Élément <big>…< /big>

Définition	Texte dans une police plus grande (préférez un style CSS).
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <blockquote>…< /blockquote>

Définition	Bloc de citation.
Enfants	address, blockquote, del, div, dl, fieldset, form, h1, h2, h3, h4, h5, h6, hr, ins, noscript, ol, p, pre, script, table, ul.
Parents	blockquote, body, button, dd, del, div, fieldset, form, ins, li, map, noscript, object, td, th.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
cite	URL d'une ressource précisant le contenu.

Élément <body>…< /body>

Définition	Corps du document.
Enfants	address, blockquote, del, div, dl, fieldset, form, h1, h2, h3, h4, h5, h6, hr, ins, noscript, ol, p, pre, script, table, ul.
Parents	html.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
onload	Événement survenant au chargement de la page.
onunload	Événement survenant à la fermeture de la page.

Élément

Définition	Saut de ligne.
Enfants	Vide.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title.

Élément <button>…< /button>

Définition	Crée un bouton personnalisable.
Enfants	Texte, abbr, acronym, address, b, bdo, big, blockquote, br, cite, code, del, dfn, div, dl, em, h1, h2, h3, h4, h5, h6, hr, i, img, ins, kbd, map, noscript, object, ol, p, pre, q, samp, script, small, span, strong, sub, sup, table, tt, ul, var.
Parents	a, abbr, acronym, address, b, bdo, big, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
accesskey	Définit la touche de raccourci clavier à l'aide d'une lettre.
disabled	"disabled".
name	Nom du bouton.
onblur	Événement qui se produit quand l'élément perd le focus.
onfocus	Événement qui se produit quand l'élément reçoit le focus.
tabindex	Entier donnant un ordre de tabulation pour l'élément.
type	(button | submit | reset).
value	Valeur associée au bouton qui sera transmise au serveur en cas de programmation dynamique (PHP par exemple).

Élément <caption>…< /caption>

Définition	Titre d'un tableau.
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, dfn, em, i, img, input, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	table.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <cite>…< /cite>

Définition	Citation courte.
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, dfn, em, i, img, input, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément < code >…< /code>

Définition	Code source.
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, dfn, em, i, img, input, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <col />

Définition	Définition d'une colonne de tableau.
Enfants	vide.
Parents	colgroup, table.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
align=(leftl centerl rightl justify)	Alignement horizontal du contenu de la colonne.
span	Nombre de colonnes identiques dans un groupe.
valign(top l middle l bottom l baseline)	Alignement vertical du contenu de la colonne.
width	Largeur de la colonne.

Élément <colgroup>…< /colgroup>

Définition	Groupe de colonnes.
Enfants	(col)*.
Parents	table.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
align=(leftl centerl rightl justifyl char)	Alignement horizontal du contenu des colonnes.
span	Nombre de colonnes du groupe.
valign (top l middle l bottom l baseline)	Alignement vertical du contenu de la colonne.
width	Largeur des colonnes.

Élément <dd>…< /dd>

Définition	Définition d'un terme dans une liste.
Enfants	Texte, a, abbr, acronym, address, b, bdo, big, blockquote, br, button, cite, code, del, dfn, div, dl, em, fieldset, form, h1, h2, h3, h4, h5, h6, hr, i, img, input, ins, kbd, label, map, noscript, object, ol, p, pre, q, samp, script, select, small, span, strong, sub, sup, table, textarea, tt, ul, var.
Parents	dl.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément ~~…~~

Définition	Signale un contenu modifié.
Enfants	Texte, a, abbr, acronym, address, b, bdo, big, blockquote, br, button, cite, code, del, dfn, div, dl, em, fieldset, form, h1, h2, h3, h4, h5, h6, hr, i, img, input, ins, kbd, label, map, noscript, object, ol, p, pre, q, samp, script, select, small, span, strong, sub, sup, table, textarea, tt, ul, var.
Parents	a, abbr, acronym, address, area, b, bdo, big, blockquote, body, button, cite, code, dd, del, dfn, div, dt, em, fieldset, form, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, noscript, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
cite	URL du fichier donnant des précisions sur la modification.
datetime	Date de modification.

Élément <dfn>…< /dfn>

Définition	Contient une définition.
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément …< /div>

Définition	Crée une division structurelle dans la page.
Enfants	Texte, a, abbr, acronym, address, b, bdo, big, blockquote, br, button, cite, code, del, dfn, div, dl, em, fieldset, form, h1, h2, h3, h4, h5, h6, hr, i, img, input, ins, kbd, label, map, noscript, object, ol, p, pre, q, samp, script, select, small, span, strong, sub, sup, table, textarea, tt, ul, var.
Parents	blockquote, body, dd, del, div, fieldset, form, ins, li, map, noscript, object, td, th.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <dl>…< /dl>

Définition	Liste de définition de termes.
Enfants	(dt I dd)+.
Parents	blockquote, body, button, dd, del, div, fieldset, form, ins, li, map, noscript, object, td, th.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <dt>…< /dt>

Définition	Terme à définir dans une liste.
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	dl.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément …< /em>

Définition	Texte mis en évidence (préférez un style CSS).
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <fieldset>…< /fieldset>

Définition	Groupe de champs d'un formulaire.
Enfants	Texte, a, abbr, acronym, address, b, bdo, big, blockquote, br, button, cite, code, del, dfn, div, dl, em, fieldset, form, h1, h2, h3, h4, h5, h6, hr, i, img, input, ins, kbd, label, legend, map, noscript, object, ol, p, pre, q, samp, script, select, small, span, strong, sub, sup, table, textarea, tt, ul, var.
Parents	blockquote, body, dd, del, div, fieldset, form, ins, li, map, noscript, object, td, th.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <form>…< /form>

Définition	Parent de tous les éléments de formulaire.
Enfants	address, blockquote, del, div, dl, fieldset, h1, h2, h3, h4, h5, h6, hr, ins, noscript, ol, p, pre, script, table, ul.
Parents	blockquote, body, dd, del, div, fieldset, ins, li, map, noscript, object, td, th.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
accept	Types MIME des fichiers acceptés en téléchargement vers un serveur.
accept-charset	Indique un jeu de caractères (exemple : iso-8859-1).
action	Nom du fichier de traitement des données du formulaire coté serveur.
enctype	Type d'encodage des informations à transmettre sur le serveur Valeur par défaut : "application/x-www-form-urlencoded".
method	Méthode de transfert des données : get (par défaut) ou post.
onreset	Événement survenant lors de l'affacemment des données saisies.
onsubmit	Événement survenant lors de l'envoi du formulaire.

Élément <frameset>…< /frameset) (DTD xhtml 1.0 frameset)

Définition	Crée un ensemble de cadres.
Enfants	(frameset I frame I noframes)*.
Parents	html.
Attributs	**Définition et valeurs**
Communs	id, class, title.
cols	Définit le partage de la fénètre en plusieurs colonnes.
onload	Événement survenant lors du chargement de la page.
onunload	Événement survenant lors de la fermeture de la page.
rows	Définit le partage de la fénètre en plusieurs lignes.

Élément <frame />(DTD xhtml 1.0 frameset)

Définition	Crée un cadre.
Enfants	Vide.
Parents	frameset.
Attributs	**Définition et valeurs**
Communs	id, class, title.
frameborder=(1 I 0)	Crée des bordures autour d'un cadre(valeur 1) ou 0 sinon.
longdesc	URL vers un document donnant une description du contenu du cadre.
marginheight	Marge haute entre le contenu et les bords supérieur et inférieur du cadre.
marginwidth	Marge haute entre le contenu et les bords gauche et droit du cadre.
name	Nom du cadre. Utilisé pour créer une communication entre deux cadres.
noresize=(noresize)	Interdit le redimensionnement du cadre par l'utilisateur.
scrolling=(yes I no I auto)	Autorise ou pas le défilement du contenu d'un cadre.
src	URL du fichier à visualiser dans un cadre.

Élément <h1>…< /h1> à <h6>…< /h6>

Définition	Titres de niveau 1 à 6.
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	blockquote, body, button, dd, del, div, fieldset, form, ins, li, map, noscript, object, td, th.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <head>…< /head>

Définition	En-tête du document.
Enfants	(title & base?) + (script l style l meta l link).
Parents	html.
Attributs	**Définition et valeurs**
Communs	xml:lang, dir.

Élément <hr />

Définition	Ligne de séparation.
Enfants	Vide.
Parents	blockquote, body, button, dd, del, div, fieldset, form, ins, li, map, noscript, object, td, th.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <html>…< /html>

Définition	Elément racine du document.
Enfants	head, body.
Parents	Aucun, c'est l'élément racine.
Attributs	**Définition et valeurs**
Communs	xml:lang, dir.

Élément <i>…< /i>

Définition	Texte en italique (préférez un style CSS).
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <img />

Définition	Image.
Enfants	Vide.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
alt	Texte descriptif en cas de non affichage de l'image ou pour les navigateurs oraux.
height	Hauteur de l'image.
ismap (ismap)	Définit s'il existe une carte de zones sensibles.
longdesc	URL du fichier de description de l'image.
src URL	du fichier image.
usemap	URL de la description de la carte des zones sensibles si elle est située coté serveur.
width	Largeur de l'image.

Élément <input />

Définition	Crée un champ de formulaire.
Enfants	Vide.
Parents	a, abbr, acronym, address, b, bdo, big, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Parents	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
accept	Type MIME des fichiers acceptés.
accesskey	Définit la touche de raccourci clavier à l'aide d'une lettre.
alt	Texte explicatif si le type est image.
checked (=checked)	Coche d'office une case à cocher.
disabled (=disabled)	Rend un champ inactif.
ismap(=ismap)	Définit s'il existe une carte de zones sensibles.
maxlength	Nombre maximal de caractères autorisés dans un champ texte.

Élément <input /> *(suite)*

name	Nom du champ.
onblur	Événement qui se produit quand l'élément perd le focus.
onchange	Événement qui se produit quand le contenu d'un élément change.
onfocus	Événement qui se produit quand l'élément reçoit le focus.
onselect	Événement qui se produit quand un texte est sélectionné dans l'élément.
readonly (readonly)	Rend le champ non modifiable.
size	Taille d'un champ de saisie en nombre de caractères.
src	URL du fichier image si type vaut "image".
tabindex	Entier donnant un ordre de tabulation pour l'élément.
type= (text l password l checkbox l radiol submit l reset file l hidden l image l button)	Détermine le type du champ de formulaire. Respectivement zone de texte, mot de passe, case à cocher, bouton radio, bouton d'envoi, bouton de réinitialisation, zone de transfert de fichier, champ caché, image et bouton personalise.
usemap	Si le type est "image", désigne l'URL de la description de la carte des zones sensibles si elle est située coté serveur.
value	Valeur associée au champ.

Élément <ins>…< /ins>

Définition	Contient un texte marque comme nouveau.
Enfants	Texte, a, abbr, acronym, address, b, bdo, big, blockquote, br, button, cite, code, del, dfn, div, dl, em, fieldset, form, h1, h2, h3, h4, h5, h6, hr, i, img, input, ins, kbd, label, map, noscript, object, ol, p, pre, q, samp, script, select, small, span, strong, sub, sup, table, textarea, tt, ul, var.
Parents	a, abbr, acronym, address, area, b, bdo, big, blockquote, body, button, cite, code, dd, del, dfn, div, dt, em, fieldset, form, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, noscript, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
cite	URL donnant la description de la modification.
datetime	Date de la modification.

Élément <kbd>…< /kbd>

Définition	Signale un contenu à saisir au clavier (exemple un raccourci).
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <label>…< /label>

Définition	Définit un libellé, souvent employé dans les formulaires.
Enfants	Texte a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Définition	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
accesskey	Définit la touche de raccourci clavier à l'aide d'une lettre.
for	Contient l'identifiant d'un autre élément auquel il est associé.
onblur	Événement qui se produit quand l'élément perd le focus.
onfocus	Événement qui se produit quand l'élément reçoit le focus.

Élément <legend>…< /legend>

Définition	Contient le titre d'un groupe de champs (de formulaire en général).
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	fieldset.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
accesskey	Définit la touche de raccourci clavier à l'aide d'une lettre.

Élément …< /li>

Définition	Item d'une liste.
Enfants	Texte, a, abbr, acronym, address, b, bdo, big, blockquote, br, button, cite, code, del, dfn, div, dl, em, fieldset, form, h1, h2, h3, h4, h5, h6, hr, i, img, input, ins, kbd, label, map, noscript, object, ol, p, pre, q, samp, script, select, small, span, strong, sub, sup, table, textarea, tt, ul, var.
Parents	ol l ul.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <link />

Définition	Lie un document annexe au document principal (une feuille de style par exemple).
Enfants	Vide.
Parents	head.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
charset	Indique un jeu de caractères (exemple : iso-8859-1).
href	Contient l'URL du document cible.
hreflang	Le code de langue du document cible.
media	Type du media concerné.
rel	Type de la relation entre le document principal et le document cible.
rev	Type de la relation entre le document cible et le document principal.
type	Type MIME du fichier cible ("text/css" par exemple).

Élément <map>…< /map>

Définition	Crée une carte de zones sensibles pour une image.
Enfants	address, area, blockquote, del, div, dl, fieldset, form, h1, h2, h3, h4, h5, h6, hr, ins, noscript, ol, p, pre, script, table, ul.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	**id**, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
name	Nom servant à identifier la carte.

Élément <meta />

Définition	Contient des méta informations sur le document.
Enfants	Vide.
Parents	head.
Attributs	**Définition et valeurs**
Communs	xml:lang, dir.
content	Contenu de l'information.
http-equiv	Équivalent du nom dans les en-têtes http.
name	Nom de l'information.

Élément <noframes>…< /noframes> (DTD xhtml 1.0 frameset)

Définition	Crée un contenu alternatif pour les navigateurs n'acceptant pas les cadres.
Enfants	body.
Parents	frameset.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <noscript>…< /noscript>

Définition	Crée un contenu alternatif pour les navigateurs n'acceptant pas les scripts.
Enfants	address, blockquote, del, div, dl, fieldset, form, h1, h2, h3, h4, h5, h6, hr, ins, noscript, ol, p, pre, script, table, ul.
Parents	area, blockquote, body, button, dd, del, div, fieldset, form, ins, li, noscript, object, td, th.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <object>…< /object>

Définition	Objet multimédia.
Enfants	Texte, a, abbr, acronym, address, b, bdo, big, blockquote, br, button, cite, code, del, dfn, div, dl, em, fieldset, form, h1, h2, h3, h4, h5, h6, hr, i, img, input, ins, kbd, label, map, noscript, object, ol, p, param, pre, q, samp, script, select, small, span, strong, sub, sup, table, textarea, tt, ul, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, head, i, ins, kbd, label, legend, li, object, p, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
archive	URL des fichiers contenant des ressources utiles à l'affichage de l'objet spécifié par classid ou data.
classid	URL d'un fichier utile à l'affiochage de l'objet (plug-in).
codebase	URL de base pour résoudre les adresses relatives des fichiers nécessaires à l'affichage de l'objet.
codetype	Type MIME.
data	URL du fichier de données de l'objet. Pour une image c'est l'adresse du fichier image.
declare(declare)	S'il est utilisé, cet attribut précise que l'élément ne fait que déclarer un objet sans l'instancier. Il faut alors inclure un autre élément object dans le premier.
height	Hauteur de l'objet.
name	Nom de l'objet.
standby	Texte à afficher pendant le temps de chargement du fichier.
tabindex	Entier donnant un ordre de tabulation pour l'élément.
type	Type MIME du fichier cible ("audio/mpeg" ou "image/gif" par exemple).
usemap	URL de la carte des zones sensibles applicable à l'objet.
width	Largeur de l'objet.

Élément …< /ol>

Définition	Liste ordonnée.
Enfants	li.
Parents	blockquote, body, button, dd, del, div, fieldset, form, ins, li, map, noscript, object, td, th.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <optgroup>…< /optgroup>

Définition	Groupe d'options dans une liste de sélection de formulaire.
Enfants	option.
Parents	select.
Définition	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
disabled= (disabled)	Désactive le groupe d'options (reste visible).
label	Libellé du groupe d'options.

Élément <option>…< /option>

Définition	Crée une option dans une liste de sélection.
Enfants	Texte brut.
Parents	select.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
disabled (disabled)	Désactive l'option (qui reste visible).
label	Libellé de l'option (les navigateurs devraient utiliser cette valeur à la place du contenu de l'élément, mais ce n'est pas encore le cas).
selected (selected)	Rend l'option sélectionnée par défaut.
value	Valeur associée à l'option.

Élément <p>…< /p>

Définition	Crée un paragraphe.
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	blockquote, body, button, dd, del, div, fieldset, form, ins, li, map, noscript, object, td, th.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <param />

Définition	Paramètre passé à un objet.
Enfants	Vide.
Parents	object.
Attributs	**Définition et valeurs**
Communs	id.
name	Nom du paramètre.
type	Type MIME du fichier cible quand l'attribut valuetype vaut "ref".
value	Valeur du paramètre.
valuetype=(data | ref | object)	Type de l'attribut value (donnée, URL ou objet).

Élément <pre>…< /pre>

Définition	Crée un texte formaté dont la mise en page est conservée (espaces et sauts de ligne).
Enfants	Texte, a, abbr, acronym, b, bdo, br, button, cite, code, dfn, em, i, input, kbd, label, map, q, samp, script, select, span, strong, textarea, tt, var.
Parents	blockquote, body, button, dd, del, div, fieldset, form, ins, li, map, noscript, object, td, th.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <q>…< /q>

Définition	Citation courte.
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
cite	URL du document donnant des précisions sur le texte cité.

Élément <samp>…< /samp>

Définition	Structure un texte d'exemple
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <script>…< /script>

Définition	Conteneur d'un script, généralement JavaScript.
Enfants	Texte du script et commentaires XHTML.
Parents	a, abbr, acronym, address, area, b, bdo, big, blockquote, body, button, cite, code, dd, del, dfn, div, dt, em, fieldset, form, h1, h2, h3, h4, h5, h6, head, i, ins, kbd, label, legend, li, noscript, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id.
charset	Indique un jeu de caractères (exemple : iso-8859-1).
defer (defer)	Signale au navigateur de ne pas interpréter le script avant d'avoir fini de charger la page.
src	URL du fichier externe contenant le code du script.
type	Précise le langage utilisé ("text/javascript" par exemple).
xml:space	(preserve).

Élément <select>…< /select>

Définition	Liste de sélection d'options.
Enfants	(optgroupoption)+.
Parents	a, abbr, acronym, address, b, bdo, big, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.

Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
disabled (disabled)	Rend la liste inactive.
multiple(multiple)	Permet des choix multiples dans la liste d'option.
name	Nom de la liste.
onblur	Événement qui se produit quand l'élément perd le focus.
onchange	Événement qui se produit quand le choix effectué a changé.
onfocus	Événement qui se produit quand l'élément reçoit le focus.
size	Nombre d'options visibles à l'affichage de la liste.
tabindex	Entier donnant un ordre de tabulation pour l'élément.

Élément <small>…< /small>

Définition	Zone de texte de petite taille.
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, q, samp, small, span, strong, sub, sup, td, th, tt, var.

Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément …< /span>

Définition	Conteneur en ligne. Utilisé pour appliquer des styles localement.
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément …< /strong>

Définition	Texte en gras.
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <style>…< /style>

Définition	Conteneur des styles CSS dans la page.
Enfants	Texte des styles CSS, commentaires XHTML.
Parents	head.
Attributs	**Définition et valeurs**
Communs	xml:lang, dir, title.
media	Type du media concerné par les styles.
type	Précise le type des styles définis ("text/css" en général).

Éléments <sub>…< /sub> et <sup>…< /sup>

Définition	Mise en indice (sub) ou en exposant (sup) d'un texte.
Enfants	Texte, a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <table>…< /table>

Définition	Conteneur des éléments de tableau.
Enfants	caption?, (col*Icolgroup*), thead?, tfoot?, tbody+.
Parents	blockquote, body, button, dd, del, div, fieldset, form, ins, li, map, noscript, object, td, th.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
border	Épaisseur de la bordure du tableau.
cellpadding	Espacement entre le contenu et la bordure des cellules.
cellspacing	Espacement entre les cellules d'un tableau.
frame=(void I above I below I hsides I lhs I rhs I vsides I box I border)	Type d'affichage des bordures externes d'un tableau.
rules=(none I groups I rows I cols I all)	Type d'affichage des bordures internes d'un tableau.
summary	Texte résumant le contenu du tableau.
width	Largeur du tableau.

Éléments <thead>…< /thead>, <tbody>…< /tbody> et <tfoot>…< /tfoot>

Définition	Créent respectivement le conteneur, de l'en-tête, du corps et du pied d'un tableau.
Enfants	tr.
Parents	table.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
align=(leftlcen-terlrightljustifylchar)	Définit l'alignement horizontal du texte.
valign = (topl middle lbottom lbaseline)	Définit l'alignement vertical du texte.

Éléments <td>…< /td> et <th>…< /th>

Définition	Créent des cellules de tableau (respectivement ordinaires et d'en-tête).
Enfants	Texte, a, abbr, acronym, address, b, bdo, big, blockquote, br, button, cite, code, del, dfn, div, dl, em, fieldset, form, h1, h2, h3, h4, h5, h6, hr, i, img, input, ins, kbd, label, map, noscript, object, ol, p, pre, q, samp, script, select, small, span, strong, sub, sup, table, textarea, tt, ul, var.
Parents	tr.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
align=(left l center l right justify l char)	Fixe l'alignement horizontal du contenu d'une cellule.
colspan	Permet de fusionner deux cellules de colonnes voisines.
rowspan	Permet de fusionner deux cellules de lignes voisines.
valign = (top l middle l bottom l baseline)	Fixe l'alignement vertical du contenu d'une cellule.

Élément <textarea>…< /textarea>

Définition	Crée une zone de saisie de texte multiligne dans un formulaire.
Enfants	Texte.
Parents	a, abbr, acronym, address, b, bdo, big, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.

Élément <textarea>…< /textarea> *(suite)*

Attributs	Définition et valeurs
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
accesskey	Définit la touche de raccourci clavier à l'aide d'une lettre.
cols	Nombre de colonnes de la zone de texte.
disabled (disabled)	Rend la zone inactive (saisie impossible).
name	Nom de la zone de saisie.
onblur	Événement qui se produit quand l'élément perd le focus.
onchange	Événement qui se produit quand le contenu de l'élément a changé.
onfocus	Événement qui se produit quand l'élément reçoit le focus.
onselect	Événement qui se produit lors d'une sélection dans l'élément.
readonly =(readonly)	Place la zone en lecture seule.
rows	Nombre de lignes de texte dans la zone.
tabindex	Entier donnant un ordre de tabulation pour l'élément.

Élément <title>…< /title>

Définition	Titre de la page (obligatoire).
Enfants	Texte.
Parents	head.
Attributs	**Définition et valeurs**
Communs	xml:lang, dir.

Élément <tr>…< /tr>

Définition	Ligne de cellules d'un tableau.
Enfants	(th | td)+.
Parents	table, tbody, tfoot, thead.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.
align=(left | center | right | justify | char)	Fixe l'alignement horizontal des cellules d'une ligne.
valign=(top | middle | bottom | baseline)	Fixe l'alignement vertical des cellules d'une ligne.

Élément <tt>…< /tt>

Définition	Texte en police télétype (à espacement fixe). Préférez un style CSS.
Enfants	Texte a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément …< /ul>

Définition	Liste non ordonnée.
Enfants	(li)+.
Parents	blockquote, body, button, dd, del, div, fieldset, form, ins, li, map, noscript, object, td, th.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Élément <var>…< /var>

Définition	Conteneur pour structurer des variables.
Enfants	Texte a, abbr, acronym, b, bdo, big, br, button, cite, code, del, dfn, em, i, img, input, ins, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, tt, var.
Parents	a, abbr, acronym, address, b, bdo, big, button, caption, cite, code, dd, del, dfn, div, dt, em, fieldset, h1, h2, h3, h4, h5, h6, i, ins, kbd, label, legend, li, object, p, pre, q, samp, small, span, strong, sub, sup, td, th, tt, var.
Attributs	**Définition et valeurs**
Communs	id, class, title, xml:lang, dir, onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup.

Annexe B

Référence CSS 2

Feuille de style par défaut recommandée par le W3C

```
html, address,
blockquote,
body, dd, div,
dl, dt, fieldset, form,
frame, frameset,
h1, h2, h3, h4,
h5, h6, noframes,
ol, p, ul, hr, pre     { display: block }
li                  { display: list-item }
head                  { display: none }
table                 { display: table }
tr                  { display: table-row }
thead                 { display: table-header-group }
tbody                 { display: table-row-group }
tfoot                { display: table-footer-group }
col                  { display: table-column }
colgroup              { display: table-column-group }
td, th               { display: table-cell }
caption               { display: table-caption }
th                { font-weight: bolder; text-align: center }
caption               { text-align: center }
body                 { margin: 8px }
h1                   { font-size: 2em; margin: .67em 0 }
h2                   { font-size: 1.5em; margin: .75em 0 }
```

```
h3                  { font-size: 1.17em; margin: .83em 0 }
h4, p,
blockquote, ul,
fieldset, form,
ol, dl,
h5                  { font-size: .83em; margin: 1.5em 0 }
h6                  { font-size: .75em; margin: 1.67em 0 }
h1, h2, h3, h4,
h5, h6, b,
strong               { font-weight: bolder }
blockquote           { margin-left: 40px; margin-right: 40px }
i, cite, em,
var, address           { font-style: italic }
pre, tt, code,
kbd, samp              { font-family: monospace }
pre                 { white-space: pre }
button, textarea,
input, select          { display: inline-block }
big                 { font-size: 1.17em }
small, sub, sup        { font-size: .83em }
sub                  { vertical-align: sub }
sup                  { vertical-align: super }
table                { border-spacing: 2px; }
thead, tbody,
tfoot                { vertical-align: middle }
td, th               { vertical-align: inherit }
del                  { text-decoration: line-through }
hr                   { border: 1px inset }
ol, ul, dd             { margin-left: 40px }
ol                   { list-style-type: decimal }
ol ul, ul ol,
ul ul, ol ol           { margin-top: 0; margin-bottom: 0 }
ins                  { text-decoration: underline }
br:before              { content: "\A" }
:before, :after        { white-space: pre-line }
:link, :visited        { text-decoration: underline }
:focus                 { outline: thin dotted invert }

@media print {
  h1                 { page-break-before: always }
  h1, h2, h3,
  h4, h5, h6           { page-break-after: avoid }
  ul, ol, dl           { page-break-before: avoid }
}
```

Référence des propriétés

Quand elle existe, la valeur par défaut de chaque propriété est indiquée en **gras souligné**.

Unités de longueur

Tous les paramètres désignés par l'expression `<longueur>` ou `N%` peuvent s'exprimer à l'aide d'un nombre et de l'une des unités suivantes :

- relatives : em, ex, px ;
- absolues : mm, cm, in (inch), pt (point), pc (pica) ;
- en pourcentage : en référence à une dimension de l'élément parent.

Couleurs

Tous les paramètres désignés par l'expression `<couleur>` représentent une couleur et peuvent être donnés par :

- des mots-clés, par exemple `black` (voir la liste des mots-clés figurant en annexe C) ;
- des codes hexadécimaux précédés du caractère dièse (#), par exemple `#FC34A9`
- une fonction `rgb(R, G, B)` dans laquelle les paramètres `R`, `G` et `B` représentent les composantes Red, Green et Blue (rouge, vert, bleu) de la couleur, exprimées par des nombres variant de 0 à 255 ou de 0 à 100 % au choix.

Propriété background-attachment

Définition	Fixe l'image de fond ou permet son défilement.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	background-attachment : scroll | fixed.
	Valeurs
scroll	L'image défile avec le contenu.
fixed	L'image reste fixe dans le fenêtre du navigateur.

Propriété background-color

Définition	Couleur de fond.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	background-color : <couleur> | transparent | inherit.
	Valeurs
<couleur>	Code de couleur, mot-clé ou fonction rgb().
transparent	Transparence qui laisse voir la couleur de l'élément parent.

Propriété background-image

Définition	Image de fond d'un élément.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	background-image : url (adresse image) I none I inherit.
Valeurs	
url(adresse image)	Adresse de l'image de fond.
none	Pas d'image.

Propriété background-position

Définition	Donne la position de l'image de fond par rapport aux bords gauche et haut de l'élément.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	background-position : [[N% I <longueur>] I [left I center I right] [NN% I <longueur>I top I center I bottom] ?] I [left I center I right] [top I center I bottom]] I inherit.
Valeurs	
<longueur>	Nombre et unité de longueur.
N%	Pourcentage de la taille de l'élément lui-même. La valeur par défaut est 0%.
top	Image placée en haut.
center	Image centrée verticalement.
bottom	Image placée en bas.
left	Image placée à gauche.
center	Image centrée horizontalement.
right	Image placée à droite.

Propriété background-repeat

Définition	Répète ou non l'image de fond dans l'élément.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	background-repeat : repeat I repeat-x I repeat-y I no-repeat.
Valeurs	
repeat	Répétition sur les axes x et y.
repeat-x	Répétition sur l'axe des x.
repeat-y	Répétition sur l'axe des y.
no-repeat	Pas de répétition.

Propriété background

Définition	Raccourci pour définir en une fois les couleur et images de fond.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	background : [background-color II background-image II background-repeat II background-attachement II background-position] I inherit.

Valeurs	
background-color	Voir les valeurs de la propriété background-color.
background-image	Voir les valeurs de la propriété background-image.
background-repeat	Voir les valeurs de la propriété background-repeat.
background-attachment	Voir les valeurs de la propriété background-attachment.
background-position	Voir les valeurs de la propriété background-position.

Propriété border-collapse

Définition	Fusionne ou non les bordures adjacentes des cellules d'un tableau.
Appliquée à	Éléments de tableau.
Héritée	Oui.
Syntaxe	border-collapse : collapse I separate I inherit.

Valeurs	
collapse	Les bordures des cellules adjacentes sont fusionnées.
separate	Les bordures ne sont pas fusionnées (chaque cellule peut avoir une bordure différente.

Propriété border-color

Définition	Raccourci pour définir la couleur des quatre bordures d'un élément.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	border-color : [<couleur> I transparent] {1,4} I inherit.

Valeurs	
<couleur>	De un à quatre codes de couleur, mot-clé ou fonction rgb().
transparent	Transparence qui laisse voir la couleur de fond.

Propriétés border-top-color, border-right-color, border-bottom-color, border-left-color

Définition	Définit individuellement la couleur d'une bordure haute, droite, basse ou gauche.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	border-top-color : <couleur> I transparent I inherit.
	Valeurs
Paramètres	Voir la propriété border-color.

Propriété border-spacing

Définition	Espacement horizontal et vertical entre les bordures des cellules de tableau. Il faut que la propriété border-collapse ait la valeur separate.
Appliquée à	Éléments de tableau.
Héritée	Oui.
Syntaxe	border-spacing : <longueur> <longueur> ? I inherit.
	Valeurs
<longueur>	De une à deux longueurs, la seconde étant facultative. La valeur par défaut est 0.

Propriété border-style

Définition	Raccourci définissant le style de une à quatre bordures d'un élément.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	border-style : [dashed I dotted I double I groove I inset I outset I ridge I solid I none] {1,4}.
	Valeurs
dashed	Pointillés ronds.
dotted	Pointillés avec tirets.
double	Trait continu double (si l'épaisseur le permet, sinon le trait est simple).
groove	Bordure en relief, double biseau creux.
inset	Relief simple en creux.
outset	Relief simple en saillie.
ridge	Bordure en relief, double biseau saillant.
solid	Trait simple continu.
none	Pas de bordure.

Propriétés border-top-style, border-right-style, border-bottom-style, border-left-style

Définition	Définit individuellement le style d'une bordure haute, droite, basse ou gauche.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	border-top-style : dashed I dotted I double I groove I inset I outset I ridge I solid I none.
	Valeurs
Paramètres	Voir la propriété border-style.

Propriété border-width

Définition	Largeur des bordures d'un élément. On peut donner de une à quatre valeurs pour les bords haut, droit, bas et gauche dans cet ordre.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	border-width : [thick I medium I thin I <longueur>]{1,4}.
	Valeurs
thick	Bordure épaisse.
medium	Bordure moyenne.
thin	Bordure mince.
<longueur>	Une unité de longueur.

Propriétés border-top-width, border-right-width, border-bottom-width, border-left-width

Définition	Définit individuellement la largeur d'une bordure haute, droite, basse ou gauche.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	border-top-width : thick I medium Ithin I <longueur>.
	Valeurs
Paramètres	Voir la propriété border-width.

Propriété border

Définition	Raccourci pour définir en une fois la largeur, le style et la couleur des quatre bordures.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	border : [border-width ll border-style ll<couleur> l transparent] l inherit.
	Valeurs
border-width	Voir la propriété border-width.
border-style	Voir la propriété border-style.
<couleur>	Voir la propriété border-color.

Propriétés border-top, border-right, border-bottom, border-left

Définition	Raccourcis permettant de définir les caractéristiques (largeur, style et couleur) des bordures haute, droite, basse ou gauche en une seule fois.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	[border-width ll border-style ll <couleur>] l inherit.
	Valeurs
border-width	Largeur de la bordure (voir la propriété border-width).
border-style	Style de la bordure (voir la propriété border-style).
<couleur>	Couleur de la bordure (voir la propriété border-color).

Propriété bottom

Définition	Position par rapport au bord bas du conteneur.
Appliquée à	Les éléments positionnés.
Héritée	Non.
Syntaxe	bottom : <longueur> l N% l auto l inherit.
	Valeurs
<longueur>	Nombre et unité de longueur.
N%	Pourcentage de la hauteur de l'élément parent.
auto	Le navigateur détermine au mieux.

Propriété caption-side

Définition	Position du titre d'un tableau.
Appliquée à	L'élément <caption> et ceux pour lesquels la propriété display vaut table-caption.
Héritée	Oui.
Syntaxe	caption-side : top | bottom | inherit.
	Valeurs
top	Titre au-dessus du tableau.
bottom	Titre en-dessous du tableau.

Propriété clear

Définition	Empêche le flottement des autres éléments sur le côté de celui pour lequel cette propriété est définie.
Appliquée à	Les éléments de bloc.
Héritée	Non.
Syntaxe	clear : none | left | right | both | inherit.
	Valeurs
none	Le flottement est autorisé.
left	Le flottement est interdit sur la gauche.
right	Le flottement est interdit sur la droite.
both	Le flottement est interdit sur les deux cotés.

Propriété clip

Définition	Définit les dimensions du rectangle de visualisation en cas de débordement.
Appliquée à	Les éléments positionnés de manière absolue.
Héritée	Non.
Syntaxe	clip : <rectangle> | auto | inherit.
	Valeurs
<rectangle>	Le rectangle est défini par la fonction rect(Xh, Xb,Yb, Yh). Les paramètres définissent le rectangle par rapport aux bords de la zone totale de l'élément.
auto	Géré par le navigateur.

Propriété color

Définition	Couleur du texte et de l'avant-plan.
Appliquée à	Tous les éléments.
Héritée	Oui.
Syntaxe	color : <couleur> I inherit.
	Valeurs
<couleur>	Code de couleur, mot-clé ou fonction rgb().

Propriété content

Définition	Crée un contenu généré avant ou après un élément.
Appliquée à	Les pseudo-éléments : before et after.
Héritée	Non.
Syntaxe	content : normal I none I [chaine I <url> I <counter> I attr(attribut) I open-quote I close-quote I no-open-quote I no-close-quote]+ I inherit.
	Valeurs
none ou normal	Aucun contenu n'est généré.
<chaine>	Le contenu de la chaine est ajouté.
<url>	Le contenu du fichier cible désigné par son adresse est ajouté.
attr(attribut)	Le contenu de l'attribut désigné est ajouté.
open-quote	Ajout des caractères choisis comme symbole d'ouverture de citation (par exemple “, ", ou <<).
close-quote	Ajout des caractères choisis comme symbole de fermeture de citation (par exemple ”, ", ou >>).

Propriété counter-increment

Définition	Incrémentation d'un compteur identifié.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	counter-increment : [<identifiant> N?]+ I none I inherit.
	Valeurs
none	Aucune incrémentation.
<identifiant>N	Incrémente le compteur identifié par son nom de la valeur N à chaque item.

Propriété counter-reset

Définition	Initialise un compteur.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	counter-reset : [<identifiant>]+ I none I inherit.
	Valeurs
<identifiant>	Initialise le compteur dont le nom est précisé par son identifiant.

Propriété direction

Définition	Sens de lecture du texte.
Appliquée à	Tous les éléments.
Héritée	Oui.
Syntaxe	direction : ltr I rtl I inherit.
	Valeurs
ltr	Lecture de gauche à droite.
rtl	Lecture de droite à gauche.

Propriété display

Définition	Définit le type d'affichage d'un élément (bloc, en ligne, liste, etc.).
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	display : inline I block I list-item I run-in I inline-block I table I inline-table I table-row-group I table-header-group I table-footer-group I table-row I table-column-group I table-column I table-cell I table-caption I none I inherit.
	Valeurs
none	Aucun affichage.
inline	Type en ligne.
block	Type bloc.
list-item	Sous forme de liste.
inline-block	Bloc en ligne.
table	Tableau (comme l'élément <table>).
inline-table	Tableau en ligne.
table-row-group	Groupe de ligne.
table-header-group	En tête de tableau (comme l'élément <thead>).
table-footer-group	Pied de tableau (comme l'élément <tfoot>).

Propriété display *(suite)*

Valeurs	
table-row	Ligne de tableau (comme l'élément <tr>).
table-column-group	Groupe de colonnes (comme l'élément <colgroup>).
table-column	Colonne de tableau (comme l'élément <col />).
table-cell	Cellule de tableau (comme les éléments <td> ou <th>).
table-caption	Titre de tableau (comme l'élément <caption>).

Propriété empty-cells

Définition	Permet de gérer la visualisation des cellules vides d'un tableau.
Appliquée à	Les éléments de tableau <td> et <th>.
Héritée	Oui.
Syntaxe	empty-cells : show I hide I inherit.
Valeurs	
show	Les cellules vides sont visibles (bord et couleur de fond).
hide	Les cellules vides sont cachées.

Propriété float

Définition	Définit le flottement d'un élément.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	float : left I right I none.
Valeurs	
left	L'élément flotte à gauche de la page.
right	L'élément flotte à droite de la page.

Propriété font-family

Définition	Un ou plusieurs noms de police de caractères ou de familles génériques du texte.
Appliquée à	Tous les éléments.
Héritée	Oui.
Syntaxe	font-family. [[nom_de_police I [serif I sans-serif I cursive I fantasy I monospace] [,nom_de_police I [serif Isans-serif I cursive I fantasy I monospace]*] I inherit.

Propriété font-family *(suite)*

	Valeurs
nom_de_police	Un nom de police courant comme Arial ou Times ; les noms composés doivent être écrits entre guillemets comme “Times New Roman”.
serif	Police à empâtement du type Times.
sans-serif	Police sans empâtement du type Arial.
cursive	Police cursive du type Script.
fantasy	Polices diverses fantaisie.
monospace	Police à espacement fixe.

Propriété font-size

Définition	Taille de la police donnée par un mot-clé ou un nombre et une unité (px, em, ex, mm, cm, in, pt, pc).
Appliquée à	Tous les éléments.
Héritée	Oui.
Syntaxe	font-size : xx-small | x-small | small | medium | large | x-large | xx-large | larger | smaller | <longueur> | N% | inherit.

	Valeurs
de xx-small xx-large	Taille progressive de très très petite à très très grande, medium étant la taille par défaut.
small	Petite taille.
medium	Taille moyenne.
larger	Taille relative plus grande que le contexte.
smaller	Taille relative plus petite que le contexte.
N%	Valeur calculée en fonction de la taille de la police de l’élément parent.

Propriété font-style

Définition	Style de la police.
Appliquée à	Tous les éléments.
Héritée	Oui.
Syntaxe	font-style : normal | italic | oblique | inherit.

	Valeurs
normal	Police droite.
italic et oblique	Police italique.

Propriété font-variant

Définition	Variantes apportées au texte dont les minuscules peuvent être écrites en petites majuscules.
Appliquée à	Tous les éléments.
Héritée	Oui.
Syntaxe	font-variant : normal | small-caps | inherit.
	Valeurs
normal	N'applique aucun effet.
small-caps	Transforme les minuscules en petites majuscules.

Propriété font-weight

Définition	Graisse de la police.
Appliquée à	Tous les éléments.
Héritée	Oui.
Syntaxe	font-weight : normal | bold | bolder | lighter | 100 à 900 | inherit.
	Valeurs
normal	Texte préservé tel quel.
bold	Texte en gras.
bolder	Graisse relative plus gras que le contexte.
lighter	Graisse relative plus maigre que le contexte.
100 à 900	De très maigre (100) à très gras (900).

Propriété font

Définition	Raccourci pour définir toutes les caractéristiques d'une police en une seule fois.
Appliquée à	Tous les éléments.
Héritée	Oui.
Syntaxe	font : [[font-style || font-variant || font-weight]? font-size [/ line-height]? font-family] | caption | icon | menu | message-box | small-caption | status-bar | inherit.
	Valeurs
font-style	Paramètres de la propriété font-style.
font-variant	Paramètres de la propriété font-variant.
font-weight	Paramètres de la propriété font-weight.
font-size	Paramètres de la propriété font-size.
line-height	Paramètres de la propriété line-height.
font-family	Paramètres de la propriété font-family.

Propriété font *(suite)*

	Valeurs
caption	Police système des titres.
icon	Police système des légendes des icônes.
menu	Police système des menus.
message-box	Police système des boites de dialogue.
small-caption	Police système des titres.
status-bar	Police système des la barre d'état.

Propriété height

Définition	Hauteur de l'élément.
Appliquée à	Tous sauf les éléments non remplacés en ligne et <col /> et <colgroup>.
Héritée	Non.
Syntaxe	height : <longueur> I N% I auto I inherit.
	Valeurs
auto	Le navigateur détermine la hauteur en fonction du contenu.
N%	Les pourcentages font référence à la hauteur du bloc parent.

Propriété left

Définition	Position par rapport au bord gauche du conteneur.
Appliquée à	Les éléments positionnés.
Héritée	Non.
Syntaxe	left : <longueur> I N% I auto I inherit.
	Valeurs
auto	Le navigateur détermine la dimension en fonction de la largeur de l'élément.
N%	Les pourcentages font référence à la largeur du bloc parent.

Propriété letter-spacing

Définition	Espacement entre les lettres, ajouté ou retranché à la valeur normale.
Appliquée à	Tous les éléments.
Héritée	Oui.
Syntaxe	letter-spacing : normal I <longueur> I inherit.
	Valeurs
normal	Espacement normal de la police utilisée.
<longueur>	Une unité de longueur positive ou négative.

Propriété line-height

Définition	Hauteur totale de ligne y compris les interlignes.
Appliquée à	Tous les éléments.
Héritée	Oui.
Syntaxe	line-height : normal I N I <longueur> I N% I inherit.
	Valeurs
normal	Hauteur normale pour la police utilisée.
N	Un coefficient multiplicateur par rapport à la hauteur de la police.
N%	Calculée par rapport à la valeur de la propriété font-size.

Propriété list-style-image

Définition	Définition de la puce graphique d'une liste.
Appliquée à	Les éléments et et ceux dont la propriété display vaut list-item.
Héritée	Oui.
Syntaxe	list-style-image : url (image) I none I inherit.
	Valeurs
none	Aucune puce.
url (image)	Donne l'adresse relative ou absolue du fichier image.

Propriété list-style-position

Définition	Donne la position du symbole de numérotation d'une liste.
Appliquée à	Les éléments et et ceux dont la propriété display vaut list-item.
Héritée	Oui.
Syntaxe	list-style-position : inside I outside I inherit.
	Valeurs
inside	Puce dans le même alignement vertical que le contenu des items de la liste.
outside	Puce en retrait à gauche par rapport aux items.

Propriété list-style-type

Définition	Type de numérotation des listes.
Appliquée à	Les éléments et et ceux dont la propriété display vaut list-item.
Héritée	Oui.
Syntaxe	list-style-type :circle|decimal|disc|lower-alpha|lower-roman|none|upper-alpha|upper-roman|square.
	Valeurs
Les mots clés sont significatifs, respectivement cercle, nombre décimal de 1 à N, disque plein, lettres minuscules, chiffres romains minuscules, pas de numérotation, lettres majuscules, chiffres romains majuscules, carré plein.	

Propriété list-style

Définition	Raccourci pour définir en une seule fois le type, la position et la puce d'une liste.
Appliquée à	Les éléments et et ceux dont la propriété display vaut list-item.
Héritée	Oui.
Syntaxe	list-style :[list-style-type || list-style-position || list-style-image] | inherit.
	Valeurs
list-style-type	Voir les valeurs de la propriété list-style-type.
list-style-position	Voir les valeurs de la propriété list-style-position.
list-style-image	Voir les valeurs de la propriété list-style-image.

Propriétés margin-bottom, margin-top, margin-left, margin-right

Définition	Définissent respectivement la marge basse, haute, gauche ou droite d'un élément.
Appliquée à	Tous les éléments sauf les éléments de tableau.
Héritée	Non.
Syntaxe	margin-bottom :<longueur> | N% | none | inherit.
	Valeurs
none	Pas de marge.
N%	Calculée par rapport à la hauteur (pour les marges haute et basse) ou à la largeur (pour les marges gauche et droite) de l'élément de bloc parent le plus proche.

Propriété margin

Définition	Raccourci pour définir les dimensions de toutes les marges en une seule propriété. On peut préciser de une à quatre valeurs.
Appliquée à	Tous les éléments sauf les éléments de tableau.
Héritée	Non.
Syntaxe	margin : [<longueur> I N%] {1,4} I inherit.
	Valeurs
N%	Calculée par rapport à la largeur de l'élément parent.

Propriété max-height, max-width

Définition	Définit respectivement une hauteur ou une largeur maximale pour la boîte d'un élément.
Appliquée à	Tous sauf les éléments en lignes non remplacés et les éléments de tableau.
Héritée	Non.
Syntaxe	max-height : <longueur> I N% I none I inherit.
	Valeurs
none	Le navigateur gère la hauteur en fonction du contenu.
N%	Calculée en fonction de la dimension de l'élément parent.

Propriété min-height, min-width

Définition	Définit respectivement une hauteur ou une largeur minimale pour la boîte d'un élément.
Appliquée à	Tous sauf les éléments en lignes non remplacés et les éléments de tableau.
Héritée	Non.
Syntaxe	min-height : <longueur> I N% I none I inherit.
	Valeurs
none	Le navigateur gère la hauteur en fonction du contenu.
N%	Calculée en fonction de la dimension de l'élément parent.

Propriété outline-color

Définition	Couleur des contours.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	outline-color : <couleur> I invert I inherit.
	Valeurs
invert	Inversion vidéo de la couleur de fond de l'élément parent.

Propriété outline-style

Définition	Définit le style des contours.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	outline-style : <border-style> I inherit.
	Valeurs
Prend les mêmes valeurs que celles de la propriété border-style.	

Propriété outline-width

Définition	Définit la largeur des contours.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	outline-width : <border-width> I inherit.
	Valeurs
Prend les mêmes valeurs que celles de la propriété border-width.	

Propriété outline

Définition	Raccourci pour définir en une seule fois toutes les caractéristiques d'un contour.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	outline : [<outline-color> II <outline-style> II <outline-width>] I inherit.
	Valeurs
Prend les mêmes valeurs que celles des propriétés outline-color, outline-style et outline-width.	

Propriété overflow

Définition	Gére les débordements de largeur et de hauteur d'un élément dimensionné.
Appliquée à	Les éléments de bloc, remplacés les cellules de tableau et les blocs en ligne.
Héritée	Non.
Syntaxe	overflow : visible I hidden I scoll I auto I inherit.
	Valeurs
visible	La boîte de l'élément est entièrement visible.
hidden	Le contenu débordant est caché.
scroll	Des barres de défilement horizontales et verticales apparaissent pour visualiser le contenu débordant.
auto	Des barres de défilement apparaissent si nécessaire.

Propriété padding-left, padding-top, padding-right, padding-bottom

Définition	Crée un espacement respectivement sur la gauche, le haut, la droite ou le bas entre le contenu et la bordure d'un élément.
Appliquée à	Tous les éléments sauf les éléments de tableau.
Héritée	Non.
Syntaxe	padding-left : <longueur> I N% I inherit.
	Valeurs
N%	Calculé par rapport à la largeur du bloc parent.

Propriété padding

Définition	Raccourci permettant de définir toutes les caractéristiques des contours. On peut définir de une à quatre valeurs différentes.
Appliquée à	Tous les éléments sauf les éléments de tableau.
Héritée	Non.
Syntaxe	padding : [<longueur> I N%] {1,4} I inherit.
	Valeurs
N%	Valeur calculée par rapport à la largeur du bloc parent.

Propriété page-break-after

Définition	Crée un saut de page après un élément donné.
Appliquée à	Les éléments de bloc.
Héritée	Non.
Syntaxe	page-break-after : auto I always I avoid I left I right I inherit.
	Valeurs
auto	Le navigateur gère les sauts de page.
always	Saut de page après chaque occurrence de l'élément.
avoid	Pas de saut de page forcé.
left	Saut de page après les pages de gauche.
right	Saut de page après les pages de droite.

Propriété page-break-before

Définition	Crée un saut de page avant un élément donné.
Appliquée à	Les éléments de bloc.
Héritée	Non.
Syntaxe	page-break-before : auto | always | avoid | left | right | inherit.
	Valeurs
	Les mêmes que pour la propriété page-break-after.

Propriété page-break-inside

Définition	Autorise un saut de page au milieu du contenu d'un élément.
Appliquée à	Les éléments de bloc.
Héritée	Oui.
Syntaxe	page-break-inside : avoid | auto | inherit.
	Valeurs
avoid	Saut de page interdit au milieu du contenu.
auto	Les sauts de page sont gérés au mieux par le navigateur.

Propriété position

Définition	Position d'un élément dans son conteneur.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	position : absolute | fixed | relative | static | inherit.
	Valeurs
absolute	Positionnement absolu.
fixed	Positionnement fixe dans la fenètre du navigateur.
relative	Positionnement relatif par rapport à la position normale.
static	Positionnement normal.

Propriété quotes

Définition	Détermine le type des cotations (guillemets, apostrophes, …) d'un contenu textuel et particulièrement les citations.
Appliquée à	Tous les éléments.
Héritée	Oui.
Syntaxe	quotes : [<chaine> <chaine>]? | none | inherit.
	Valeurs
<chaine> <chaine>	Indique les chaînes de début et de fin.

Propriété right

Définition	Position d'un élément par rapport au bord droit de son conteneur.
Appliquée à	Les éléments positionnés.
Héritée	Non.
Syntaxe	right : <longueur> I N% I auto I inherit.
	Valeurs
N%	Valeur calculée par rapport à la largeur de l'élément parent.

Propriété table-layout

Définition	Détermine le choix de l'algorithme de calcul de la largeur d'un tableau.
Appliquée à	Les tableaux.
Héritée	Non.
Syntaxe	table-layout : auto I fixed I inherit.
	Valeurs
auto	La largeur dépend du contenu des cellules.
fixed	La largeur totale ne dépend pas du contenu des cellules mais du dimensionnement de celles-ci.

Propriété text-align

Définition	Alignement horizontal du texte dans son conteneur.
Appliquée à	Les éléments de bloc et cellules de tableau.
Héritée	Oui.
Syntaxe	text-align : left I center I right I justify.
	Valeurs
left	Texte aligné à gauche (valeur par défaut pour une lecture de gauche à droite, sinon c'est la valeur right).
center	Texte centré.
right	Texte aligné à droite.
justify	Texte justifié.

Propriété text-decoration

Définition	Crée des effets décoratifs sur un texte.
Appliquée à	Tous les éléments.
Héritée	Non.
Syntaxe	text-decoration : none I [underline II overline II line-through II blink] I inherit.

Propriété text-decoration

	Valeurs
underline	Texte souligné.
overline	Texte surligné.
line-through	Texte barré.
blink	Texte clignotant.

Propriété text-indent

Définition	Détermine l'indentation du texte de l'élément.
Appliquée à	Les éléments de bloc, cellules de tableau et blocs en ligne.
Héritée	Oui.
Syntaxe	text-indent : <longueur> I N% I inherit.
	Valeurs
N%	Calculé par rapport à la largeur du bloc.

Propriété text-transform

Définition	Transforme la casse du texte.
Appliquée à	Tous les éléments.
Héritée	Oui.
Syntaxe	text-transform : capitalize I lowercase I uppercase I none I inherit.
	Valeurs
capitalize	Texte avec une majuscule au début de chaque mot.
lowercase	Texte en minuscules.
uppercase	Texte en majuscules.

Propriété top

Définition	Position par rapport au bord droit du conteneur.
Appliquée à	Les éléments positionnés.
Héritée	Non.
Syntaxe	top : <longueur> I N% I auto I inherit.
	Valeurs
N%	Valeur calculée par rapport à la hauteur du bloc parent.

Propriété vertical-align

Définition	Alignement d'un contenu dans un élément.
Appliquée à	Élément en ligne et cellules de tableau.
Héritée	Non.
Syntaxe	vertical-align : baseline I bottom I middle I sub I super I text-bottom I text-top I top I N%.
	Valeurs
baseline	Alignement sur la ligne de base des caractères.
bottom	Alignement sur le bas de la boîte du texte.
middle	Alignement sur la ligne du milieu du texte minuscule.
sub	Alignement en indice.
super	Alignement en exposant.
text-bottom	Alignement sur le bas du texte.
text-top	Alignement sur le haut du texte.
top	Alignement sur le haut de la boîte du texte.
N%	Calculée par rapport à la valeur de la propriété line-height.

Propriété visibility

Définition	Détermine la visibilité ou non d'un élément.
Appliquée à	Tous les éléments.
Héritée	Oui.
Syntaxe	visibility : collapse I hidden I visible I inherit.
	Valeurs
collapse	Utilisée pour les éléments <col /> et <colgroup> elle empêche la visibilité de toute la colonne.
hidden	L'élément est caché.
visible	L'élément est visible.

Propriété width

Définition	Fixe la largeur d'un élément.
Appliquée à	Tous les éléments.
Héritée	Oui.
Syntaxe	width : <longueur> I N% I auto I inherit.
	Valeurs
N%	Valeur calculée par rapport à la largeur de l'élément parent.

Propriété white-space

Définition	Permet de gérer l'affichage des espaces blancs multiples.
Appliquée à	Tous les éléments.
Héritée	Oui.
Syntaxe	white-space : normal I pre I nowrap I pre-wrap I pre-line I inherit.
	Valeurs
normal	Gestion habituelle : un seul espace est affiché quel qu'en soit le nombre.
pre	Tous les espaces et retours chariot sont conservés.
nowrap	Empêche les sauts de ligne.
pre-wrap	Les espaces du texte sont conservés mais pas les sauts de ligne.
pre-line	Les espaces multiples sont réduits à un seul mais les sauts de ligne sont conservés.

Propriété word-spacing

Définition	Définit l'espacement ajouté ou retiré entre les mots par rapport à la valeur normale issue de la police utilisée.
Appliquée à	Tous les éléments.
Héritée	Oui.
Syntaxe	word-spacing : normal I <longueur> I inherit.
	Valeurs
normal	Espacement normal de la police utilisée.
<longueur>	Augmente ou diminue la valeur par défaut de l'unité précisée (la valeur peut être négative).

Propriété z-index

Définition	Crée l'ordre d'empilement d'un élément par rapport aux autres en cas de positionnement.
Appliquée à	Les éléments positionnés.
Héritée	Non.
Syntaxe	z-index : auto N I inherit.
	Valeurs
N	Entier donnant l'ordre d'empilement, l'élément ayant le plus grand étant placé au dessus des autres.
auto	Le dernier apparu dans le code est placé au dessus.

Pseudo-classes

pseudo-élément	Permet d'appliquer un style
X:active	un élément X qui devient actif.
a:link	à un lien <a> non actif (dans son état initial).
a:visited	un lien <a> déjà visité.
X:hover	un élément X survolé par le curseur.
X:first-child	un élément X qui est le premier enfant (dans l'ordre du code XHTML) de son élément parent.
X:focus	l'élément qui reçoit le focus.
X:lang(code)	un élément dont le code de langue a la valeur code.
@page:first	la première page imprimée d'un document (non prise en compte actuellement sauf par Opera).
@page:left	aux pages de gauche (donc paires) d'un document imprimé (non prise en compte actuellement sauf par Opera).
@page:right	aux pages de droite (donc impaires) d'un document imprimé (non prise en compte actuellement sauf par Opera).

Pseudo-éléments

pseudo-élément	Permet d'appliquer un style
X:after	après le contenu de l'élément X.
X:before	avant le contenu de l'élément X.
X:first-letter	à la première lettre du texte de l'élément X.
X:first-line	à la première ligne d'un texte contenu dans l'élément X.

Annexe C

Codes des couleurs

Pour visualiser le rendu réel des couleurs présentées ci-après, consulter le site :
http://www.funxhtml.com/couleur

Les paramètres de la fonction `rgb()` peuvent être des pourcentages variant entre 0 et 100 % pour chaque couleur indiquée dans l'ordre RGB (Red, Green, Blue).

Nom	Code hexadécimal	Code RGB décimal
aliceblue	F0 F8 FF	rgb(240,248,255)
antiquewhite	FA EB D7	rgb(250,235,215)
aqua	00 FF FF	rgb(0,255,255)
aquamarine	7F FF D4	rgb(127,255,212)
azure	F0 FF FF	rgb(240,255,255)
beige	F5 F5 DC	rgb(245,245,220)
bisque	FF E4 C4	rgb(255,228,196)
black	00 00 00	rgb(0,0,0)
blanchedalmond	FF EB CD	rgb(255,255,205)
blue	00 00 FF	rgb(0,0,255)
blueviolet	8A 2B E2	rgb(138,43,226)
brown	A5 2A 2A	rgb(165,42,42)
burlywood	DE B8 87	rgb(222,184,135)
cadetblue	5F 9E A0	rgb(95,158,160)

Nom	Code hexadécimal	Code RGB décimal
chartreuse	7F FF 00	rgb(127,255,0)
chocolate	7F FF 00	rgb(210,105,30)
coral	FF 7F 50	rgb(255,127,80)
cornflowerblue	64 95 ED	rgb(100,149,237)
cornsilk	FF F8 DC	rgb(255,248,220)
crimson	DC 14 3C	rgb(220,20,60)
cyan	00 FF FF	rgb(0,255,255)
darkblue	00 00 8B	rgb(0,0,139)
darkcyan	00 8B 8B	rgb(0,139,139)
darkgoldenrod	B8 86 0B	rgb(184,134,11)
darkgray	A9 A9 A9	rgb(169,169,169)
darkgreen	00 64 00	rgb(0,100,0)
darkkhaki	BD B7 6B	rgb(189,183,107)
darkmagenta	8B 00 8B	rgb(139,0,139)
darkolivegreen	55 6B 2F	rgb(85,107,47)
darkorange	FF 8C 00	rgb(255,140,0)
darkorchid	99 32 CC	rgb(153,50,204)
darkred	8B 00 00	rgb(139,0,0)
darksalmon	E9 96 7A	rgb(233,150,122)
darkseagreen	8F BC 8F	rgb(143,188,143)
darkslateblue	48 3D 8B	rgb(72,61,139)
darkslategray	2F 4F 4F	rgb(47,79,79)
darkturquoise	00 CE D1	rgb(0,206,209)
darkviolet	94 00 D3	rgb(148,0,211)
deeppink	FF 14 93	rgb(255,20,147)
deepskyblue	00 BF FF	rgb(0,191,255)
dimgray	69 69 69	rgb(105,105,105)
dodgerblue	1E 90 FF	rgb(30,144,255)
firebrick	B2 22 22	rgb(178,34,34)
floralwhite	FF FA F0	rgb(255,250,240)
forestgreen	22 8B 22	rgb(34,139,34)
fuchsia	FF 00 FF	rgb(255,0,255)

Nom	Code hexadécimal	Code RGB décimal
gainsboro	DC DC DC	rgb(220,220,220)
ghostwhite	F8 F8 FF	rgb(248,248,255)
gold	FF D7 00	rgb(255,215,0)
goldenrod	DA A5 20	rgb(218,165,32)
gray	80 80 80	rgb(127,127,127)
green	00 80 00	rgb(0,128,0)
greenyellow	AD FF 2F	rgb(173,255,47)
honeydew	F0 FF F0	rgb(240,255,240)
hotpink	FF 69 B4	rgb(255,105,180)
indianred	CD 5C 5C	rgb(205,92,92)
ivory	FF FF F0	rgb(255,255,240)
khaki	F0 E6 8C	rgb(240,230,140)
lavender	E6 E6 FA	rgb(230,230,250)
lavenderblush	FF F0 F5	rgb(255,240,245)
lawngreen	7C FC 00	rgb(124,252,0)
lemonchiffon	FF FA CD	rgb(255,250,205)
lightblue	AD D8 E6	rgb(173,216,230)
lightcora	F0 80 80	rgb(240,128,128)
lightcyan	E0 FF FF	rgb(224,255,255)
lightgoldenrodyellow	FA FA D2	rgb(250,250,210)
lightgreen	90 EE 90	rgb(144,238,144)
lightgrey	D3 D3 D3	rgb(211,211,211)
lightpink	FF B6 C1	rgb(255,1182,193)
lightsalmon	FF A0 7A	rgb(255,160,122)
lightseagreen	20 B2 AA	rgb(32,178,170)
lightskyblue	87 CE FA	rgb(135,206,250)
lightslategray	77 88 99	rgb(119,136,153)
lightsteelblue	B0 C4 DE	rgb(176,196,222)
lightyellow	FF FF E0	rgb(255,255,224)
lime	00 FF 00	rgb(0,255,0)
limegreen	32 CD 32	rgb(50,205,50)
linen	FA F0 E6	rgb(250,240,230)

Nom	Code hexadécimal	Code RGB décimal
magenta	FF 00 FF	rgb(255,0,255)
maroon	80 00 00	rgb(128,0,0)
mediumaquamarine	66 CD AA	rgb(102,205,170)
mediumblue	00 00 CD	rgb(0,0,205)
mediumorchid	BA 55 D3	rgb(186,85,211)
mediumpurple	93 70 DB	rgb(147,112,219)
mediumseagreen	3C B3 71	rgb(60,179,113)
mediumslateblue	7B 68 EE	rgb(123,104,238)
mediumspringgreen	00 FA 9A	rgb(0,250,154)
mediumturquoise	48 D1 CC	rgb(72,209,204)
mediumvioletred	C7 15 85	rgb(199,21,133)
midnightblue	19 19 70	rgb(25,25,112)
mintcream	F5 FF FA	rgb(245,255,250)
mistyrose	FF E4 E1	rgb(255,228,225)
moccasin	FF E4 B5	rgb(255,228,181)
navajowhite	FF DE AD	rgb(255,222,173)
navy	00 00 80	rgb(0,0,128)
oldlace	FD F5 E6	rgb(253,245,230)
olive	80 80 00	rgb(128,128,0)
olivedrab	6B 8E 23	rgb(107,142,35)
orange	FF A5 00	rgb(255,165,0)
orangered	FF 45 00	rgb(255,69,0)
orchid	DA 70 D6	rgb(218,112,214)
palegoldenrod	EE E8 AA	rgb(238,232,170)
palegreen	98 FB 98	rgb(152,251,152)
paleturquoise	AF EE EE	rgb(175,238,238)
palevioletred	DB 70 93	rgb(219,112,147)
papayawhip	FF EF D5	rgb(255,239,213)
peachpuff	FF DA B9	rgb(255,218,185)
peru	CD 85 3F	rgb(205,133,63)
pink	FF C0 CB	rgb(255,192,203)
plum	DD A0 DD	rgb(221,160,221)

Nom	Code hexadécimal	Code RGB décimal
powderblue	B0 E0 E6	rgb(176,224,230)
purple	80 00 80	rgb(128,0,128)
red	FF 00 00	rgb(255,0,0)
rosybrown	BC 8F 8F	rgb(188,143,143)
royalblue	41 69 E1	rgb(65,105,225)
saddlebrown	8B 45 13	rgb(139,69,19)
salmon	FA 80 72	rgb(250,128,114)
sandybrown	F4 A4 60	rgb(244,164,96)
seagreen	2E 8B 57	rgb(46,139,87)
seashell	FF F5 EE	rgb(255,245,238)
sienna	A0 52 2D	rgb(160,82,45)
silver	C0 C0 C0	rgb(192,192,192)
skyblue	87 CE EB	rgb(135,206,235)
slateblue	6A 5A CD	rgb(106,90,205)
slategray	70 80 90	rgb(112,128,144)
snow	FF FA FA	rgb(255,250,250)
springgreen	00 FF 7F	rgb(0,255,127)
steelblue	46 82 B4	rgb(70,130,180)
tan	D2 B4 8C	rgb(210,180,140)
teal	00 80 80	rgb(0,128,128)
thistle	D8 BF D8	rgb(216,191,216)
tomato	FF 63 47	rgb(255,99,71)
turquoise	40 E0 D0	rgb(64,224,208)
violet	EE 82 EE	rgb(238,130,238)
wheat	F5 DE B3	rgb(245,222,179)
white	FF FF FF	rgb(255,255,255)
whitesmoke	F5 F5 F5	rgb(245,245,245)
yellow	FF FF 00	rgb(255,255,0)
yellowgreen	FF FF 00	rgb(139,205,50)

Annexe D

Les entités de caractères

Caractère	Code	Entité	Caractère	Code	Entité
!	&#33;		"	&#34;	&quot;
#	&#35;		$	&#36;	
%	&#37;		&	&#38;	&amp;
‘	&#39;		(	&#40;	
)	&#41;		*	&#42;	
+	&#43;		,	&#44;	
-	&#45;		.	&#46;	
/	&#47;		0	&#48;	
1	&#49;		2	&#50;	
3	&#51;		4	&#52;	
5	&#53;		6	&#54;	
7	&#55;		8	&#56;	
9	&#57;		:	&#58;	
;	&#59;		<	&#60;	&lt;
=	&#61;		>	&#62;	&gt;
?	&#63;		@	&#64;	
A	&#65;		B	&#66;	
C	&#67;		D	&#68;	

Caractère	Code	Entité	Caractère	Code	Entité
E	&#69;		F	&#70;	
G	&#71;		H	&#72;	
I	&#73;		J	&#74;	
K	&#75;		L	&#76;	
M	&#77;		N	&#78;	
O	&#79;		P	&#80;	
Q	&#81;		R	&#82;	
S	&#83;		T	&#84;	
U	&#85;		V	&#86;	
W	&#87;		X	&#88;	
Y	&#89;		Z	&#90;	
[	&#91;		\	&#92;	
]	&#93;		^	&#94;	
_	&#95;		`	&#96;	
a	&#97;		b	&#98;	
c	&#99;		d	&#100;	
e	&#101;		f	&#102;	
g	&#103;		h	&#104;	
i	&#105;		j	&#106;	
k	&#107;		l	&#108;	
m	&#109;		n	&#110;	
o	&#111;		p	&#112;	
q	&#113;		r	&#114;	
s	&#115;		t	&#116;	
u	&#117;		v	&#118;	
w	&#119;		x	&#120;	
y	&#121;		z	&#122;	
{	&#123;		|	&#124;	
}	&#125;		~	&#126;	
€	&#128;	&euro;	‚	&#130;	
ƒ	&#131;		„	&#132;	
…	&#133;		†	&#134;	

Caractère	Code	Entité	Caractère	Code	Entité
‡	‡		ˆ	ˆ	
‰	‰		Š	Š	
<	‹		Œ	Œ	
			Ž	Ž	
‘	‘		’	’	
“	“		”	”	
•	•		–	–	
—	—		˜	˜	
™	™		š	š	
>	›		œ	œ	
			ž	ž	
Ÿ	Ÿ		espace		
¡	¡	¡	¢	¢	¢
£	£	£	¤	¤	¤
¥	¥	¥	¦	¦	¦
§	§	§	¨	¨	¨
©	©	©	ª	ª	ª
«	«	«	¬	¬	¬
			®	®	®
¯	¯	¯	°	°	°
±	±	±	²	²	²
³	³	³	´	´	´
µ	µ	µ	¶	¶	¶
·	·	·	¸	¸	¸
¹	¹	¹	º	º	º
»	»	»	¼	¼	¼
½	½	½	¾	¾	¾
¿	¿	¿	À	À	À
Á	Á	Á	Â	Â	Â
Ã	Ã	Ã	Ä	Ä	Ä
Å	Å	Å	Æ	Æ	&Aelig;
Ç	Ç	Ç	È	È	È

Caractère	Code	Entité	Caractère	Code	Entité
É	&#201;	&Eacute;	Ê	&#202;	&Ecirc;
Ë	&#203;	&Euml;	Ì	&#204;	&Igrave;
Í	&#205;	&Iacute;	Î	&#206;	&Icirc;
Ï	&#207;	&Iuml;	Ð	&#208;	&ETH;
Ñ	&#209;	&Ntilde;	Ò	&#210;	&Ograve;
Ó	&#211;	&Oacute;	Ô	&#212;	&Ocirc;
Õ	&#213;	&Otilde;	Ö	&#214;	&Ouml;
×	&#215;	&times;	Ø	&#216;	&Oslash;
Ù	&#217;	&Ugrave;	Ú	&#218;	&Uacute;
Û	&#219;	&Ucirc;	Ü	&#220;	&Uuml;
Ý	&#221;	&Yacute;	Þ	&#222;	&THORN;
ß	&#223;	&szlig;	à	&#224;	&agrave;
á	&#225;	&aacute;	â	&#226;	&acirc;
ã	&#227;	&atilde;	ä	&#228;	&auml;
å	&#229;	&aring;	æ	&#230;	&aelig;
ç	&#231;	&ccedil;	è	&#232;	&egrave;
é	&#233;	&eacute;	ê	&#234;	&ecirc;
ë	&#235;	&euml;	ì	&#236;	&igrave;
í	&#237;	&iacute;	î	&#238;	&icirc;
ï	&#239;	&iuml;	ð	&#240;	&eth;
ñ	&#241;	&ntilde;	ò	&#242;	&ograve;
ó	&#243;	&oacute;	ô	&#244;	&ocirc;
õ	&#245;	&otilde;	ö	&#246;	&ouml;
÷	&#247;	&divide;	ø	&#248;	&oslash;
ù	&#249;	&ugrave;	ú	&#250;	&uacute;
û	&#251;	&ucirc;	ü	&#252;	&uuml;
ý	&#253;	&yacute;	þ	&#254;	&thorn;
ÿ	&#255;	&yuml;	€	&#8364;	&euro;

Annexe E

Bibliographie et adresses utiles

Bibliographie

- *Référencement sur le net* de Sandrine Saporta aux Éditions d'Organisation.
- *Créer du trafic sur son site web* de Olivier Andrieu téléchargeable sur le site : http://www.boutique-abondance.com
- *JavaScript - Les références du programmeur* de Jean Engels aux Éditions OEM
- *PHP 5 - Cours et exercices* de Jean Engels aux Éditions Eyrolles
- *CSS - La référence* de Éric Meyer aux Éditions O'Reilly

Adresses utiles

- Les fichiers du livre : vous y trouverez les fichiers du livre à télécharger et à afficher dans votre navigateur :

  http://www.funhtml.com/xhtml
- Spécifications XHTML du World Wide Web Consortium (W3C) :

  http://www.w3.org/TR/xhtml1/
- Spécifications CSS 2.1 du World Wide Web Consortium (W3C) :

  http://www.w3.org/TR/CSS21/

- Téléchargement des différents navigateurs :
  - –Mozilla : http://www.mozilla.org/releases/#1.7.11
  - –FireFox : http://www.mozilla.org/products/firefox/all.html
  - –Opera : http://www.opera.com/download/
  - –Internet Explorer : http://www.microsoft.com/downloads
- Des applets Java :
  http://www.gamelan.com
- Des scripts JavaScript :
  http://javascriptsource.com
  http://www.javascript.com
- Utilitaires pour le choix des couleurs et CSS2 en français :
  http://www.yoyodesign.org
  http://www.lighthouse.org/color_contrast.htm
- Des scripts PHP :
  http://www.funhtml.com/php5
  http://www.nexen.net

Index

C

D