Francis **Draillard**

accès libre

Premiers pas en CSS et XHTML

2e édition

EYROLLES

Premiers pas en CSS et XHTML

→ 2e édition

Collection « Accès libre »

Pour que l'informatique soit un outil, pas un ennemi !

Réussir son site web avec XHTML et CSS.
M. Nebra.
N°12307, 2e édition, 2008, 306 pages.

Ergonomie web. *Pour des sites web efficaces.*
A. Boucher.
N°12158, 2007, 426 pages.

Réussir un site web d'association… avec des outils libres !
A.-L. Quatravaux et D. Quatravaux.
N°12000, 2e édition, 2007, 372 pages.

Réussir un projet de site web.
N. Chu.
N°11974, 4e édition, 2006, 230 pages.

Réussir son site e-commerce avec osCommerce.
D. Mercer.
N°11932, 2007, 446 pages.

Scenari – La chaîne éditoriale libre.
S. Crozat.
N°12150, 2007, 200 pages.

Tiny ERP/Open ERP. *Pour une gestion d'entreprise efficace et intégrée.*
F. Pinckaers et G. Gardiner.
N°12261, 2008, 287 pages.

Gimp 2.4 efficace. *Dessin et retouche photo.*
C. Gémy.
N°12152, 2008, 402 pages avec CD-Rom.

La 3D libre avec Blender.
O. Saraja.
N°12196, 2e édition, 2007, 420 pages avec CD-Rom.

Mise en page avec OpenOffice.org Writer.
I. Barzilai.
N°12149, 2007, 338 pages.

OpenOffice.org 2.2 efficace.
S. Gautier, C. Hardy, F. Labbe, M. Pinquier.
N°12166, 2007, 420 pages avec CD-Rom.

Ubuntu efficace.
L. Dricot, avec la contribution de R. Mas.
N°12003, 2e édition, 2006, 360 pages avec CD-Rom.

PGP/GPG – *Assurer la confidentialité de ses mails et fichiers.*
M. Lucas, ad. par D. Garance , contrib. J.-M. Thomas.
N°12001, 2006, 248 pages.

Monter son serveur de mails sous Linux
M. Bäck *et al*., adapté par P. Tonnerre.
N°11931, 2006, 360 pages.

Collection « Poche Accès libre »

Gimp 2.4. *Débuter en retouche photo et graphisme libre*.
D. Robert.
N°12295, 2e édition, 2008, 300 pages environ.

SPIP 1.9. *Créer son site avec des outils libres.*
M.-M. Maudet. A.-L. Quatravaux, D. Quatravaux., avec la contribution de Perline.
N°12002, 2007, 376 pages.

OpenOffice.org 2 Writer.
S. Gautier, avec la contribution de G. Veyssière.
N°11668, 2005, 248 pages.

Mozilla Thunderbird. *Le mail sûr et sans spam.*
D. Garance, A.-L. et D. Quatravaux.
N°11609, 2005, 300 pages avec CD-Rom.

Firefox. *Retrouvez votre efficacité sur le Web !*
T. Trubacz, préface de T. Nitot.
N°11604, 2005, 250 pages.

OpenOffice.org 2 Calc.
S. Gautier, avec la contribution de J.-M. Thomas.
N°11667, 2006, 220 pages.

Chez le même éditeur

É. Sloïm. – **Sites web.** Les bonnes pratiques*. N°12101, 2007, 14 pages.*
C. Porteneuve, préface de T. Nitot. – **Bien développer pour le Web 2.0** – Bonnes pratiques Ajax. *N°12028, 2007, 580 pages.*
R. Goetter. – **CSS 2 : pratique du design web** . *N°11976, 2e édition, 2007, 324 pages.*
R. Goetter. – **Mémento CSS.** *N°11726, 2006, 14 pages.*
R. Goetter. – **Mémento XHTML**. *N°11955, 2006, 14 pages.*
A. Andrieu. – **Réussir son référencement web**. *N°12264, 2008, 302 pages.*
J.-M. Defrance. – **Premières applications Web 2.0 avec Ajax et PHP.** *N°12090, 2008, 450 pages.*
A. Clarke. **– Transcender CSS.** Sublimez le design web ! *N°12107, 2007, 370 pages.*
G. Davis. – **500 grilles et feuilles de styles pour l'imprimé et le Web.** *N°12267, 2008, 160 pages avec CD-Rom.*
D. Tardiveau. – **150 scripts pour Flash CS3**. N°12112, 2007, 520 pages.
Laboratoire SUPINFO des technologies Apple. – **Mac OS X Leopard**. *N°12272, à paraître, 2008.*
G. Gete. – **Mémento Mac OS X.** *N°11935, 2006, 14 pages.*
I. Hurbain, E. Dreyfus. – **Mémento Unix/Linux.** *N°11954, 2006, 14 pages.*
R. Hertzog, R. Mas. – **Debian Etch. Gnu/Linux.** *N°12062, 2007, 428 pages avec DVD.*
É. Daspet et C. Pierre de Geyer. – **PHP5 avancé**. *N°12167, 4e édition, 2007, 792 pages.*
C. Pierre de Geyer et G. Ponçon. – **Mémento PHP et SQL.** *N°11785, 2006, 14 pages.*
R. Rimelé. – **Mémento MySQL.** N°12012, 2007, 14 pages.

Francis **Draillard**

Premiers pas en CSS et XHTML

→ 2e édition

EYROLLES

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

Remerciements à Jean-Baka Domelevo-Entfellner

Avant-propos

Des pages web ? Oui, mais avec du style ! Si les pages que nous concevons ont belle allure, c'est bien ; mais qu'y a-t-il derrière, comment sont codées ces magnifiques pages ?

Sera-t-il facile de changer la charte graphique du site ? Ou de modifier la structure d'une page ? Sera-t-il possible de créer de nouvelles pages en réutilisant le travail de mise en forme déjà effectué ?

Bref, à quoi sert d'avoir une voiture rutilante, avec toit ouvrant, jantes alliage, rétroviseurs électriques et tout le tralala, s'il faut démonter le moteur pour faire la vidange ? C'est une comparaison exagérée, certes, mais qui a le mérite de poser clairement le problème.

Vous avez donc compris qu'au-delà du résultat affiché d'un site web, il faut penser à sa maintenance : rectifications, mises à jour, changements de mise en page doivent pouvoir s'effectuer facilement. Alors, existe-t-il une technique pour améliorer les pages HTML classiques, souvent pleines à craquer d'attributs de mise en forme et de tableaux pour obtenir du côte à côte ? Bien sûr ! Et vous avez de la chance, c'est justement l'objet de l'ouvrage qui se trouve entre vos mains !

Une méthode moderne, pour créer des sites web de bonne qualité et faciles à maintenir, consiste à écrire les pages en XHTML, langage clair et efficace, en association avec des feuilles de style ou CSS, pour *Cascading Style Sheets*.

Les feuilles de style sont utilisées depuis longtemps dans les traitements de texte. Elles facilitent la mise en forme, tout en rendant plus homogènes les différentes pages d'un long document. En ce qui concerne les pages web,

les premières normes pour les feuilles de style, CSS 1, ont été publiées fin 1996, suivies des normes CSS 2 en 1998.

Il a fallu que les logiciels de navigation web les prennent en compte pour que les concepteurs puissent enfin les utiliser, d'où un certain « retard à l'allumage » que nous retrouverons de la même façon avec les normes CSS 3 à venir.

Les CSS 2 sont disponibles, profitons-en ! Et découvrons ensemble dans cet ouvrage tout le bénéfice qu'elles nous apportent : une façon différente d'appréhender la conception des pages web. Il suffit de s'y aventurer pour être conquis... Bonne lecture !

Structure de l'ouvrage

Le **premier chapitre** est une introduction qui nous présente le principe général du XHTML, des feuilles de style et d'une bonne écriture XHTML/CSS.

Le **deuxième chapitre** présente de façon concrète les bases du XHTML. À partir d'exemples, il détaille l'utilisation des principales balises XHTML, présente leur classement par types de balise et leur hiérarchie dans une page web. Il fournit les quelques mots de vocabulaire qui seront utilisés par la suite pour expliquer la conception d'une feuille de style.

Au **troisième chapitre** apparaissent les feuilles de style. À partir d'exemples, nous verrons de quelle façon et à quel endroit les écrire, quelles unités de mesure utiliser, etc.

Les propriétés CSS sont détaillées dans les **quatrième** et **cinquième chapitres**, qui expliquent respectivement les propriétés de mise en forme et celles liées au positionnement des éléments dans la page. Des exemples illustrent chaque propriété, dont toutes les valeurs possibles sont détaillées.

Le **sixième chapitre** nous parle des autres médias pour lesquels des propriétés de style existent et le **septième chapitre** présente des astuces très pratiques, pour adapter les feuilles de style aux différents navigateurs, en particulier pour remplacer quelques propriétés mal interprétées par la version 6 de Internet Explorer.

En **annexes** se trouvent les noms et codes des couleurs de base, ainsi qu'un tableau de synthèse sur le comportement des principaux navigateurs. Suivent un index des propriétés, en guise de formulaire, puis une liste de références bibliographiques et de sites web utiles.

Les fichiers qui servent d'exemples dans le livre peuvent être téléchargés à l'adresse http://www.antevox.fr/livre.

Crédits des photographies et illustrations

Les crédits qui ne figurent pas dans les légendes des illustrations sont mentionnés ci-après.

Pages de garde des chapitres 1 et 5 : extraits du site ZenGarden http://www.csszengarden.com/tr/francais/, respectivement les versions « Tranquille » par Dave Shea (http://www.mezzoblue.com) et « Like the Sea » par Lars Daum (http://www.redrotate.de/).

Figures 1-3, 3-4 à 3-11, 6-2, B-1, pages de garde des chapitres 3, 4 et 6, ainsi que des annexes A, C et D : copyright 2006 Francis Draillard, Micro Application et ses concédants.

Figures 1-1, 1-2, 2-1 à 2-8, 2-1, 2-12 à 2-16, 3-1, 3-2, 3-12 à 3-14, 4-1 à 4-15, 5-1 à 5-16, 6-1, 7-7 à 7-9, A-1, B-2, B-3, pages de garde du chapitre 7 et de l'annexe B : Francis Draillard.

Page de garde du chapitre 2 : dessin d'Alice Draillard.

Figure 3-3 : extraite du site http://www.wikipedia.fr, photo d'Éric Pouhier (décembre 2005).

Figures 5-9, 5-12 à 5-14, 7-7 et 7-9 : utilisation d'illustrations provenant du site http://www.wikipedia.fr

Remerciements

Je tiens à remercier Muriel Shan Sei Fan, éditrice informatique des Éditions Eyrolles. C'est grâce à elle que la publication de ce livre a été possible et ses conseils m'ont été précieux pour la rédaction finale. Merci aussi à Dimitri Robert : auteur d'un excellent livre sur le logiciel Gimp, dans cette même collection, il m'a aiguillé vers Muriel pour lui proposer mon manuscrit.

Merci encore à Eliza Gapenne et Anne Bougnoux pour la relecture de ce livre, à Gaël Thomas et Jean-Marie Thomas pour sa mise en page.

Pour cette deuxième édition, je tiens à remercier Jean-Baka Domelevo Entfellner pour la relecture, Muriel Shan Sei Fan et Karine Joly pour leurs conseils et la coordination, Gaël Thomas pour la mise en page.

Je remercie beaucoup pour leur contribution :

- Alain Beyrand (http://www.pressibus.org) : son classement des couleurs est très intéressant. Il est publié en annexe (mais en moins bien, car sans les couleurs !).
- David Hammond (http://nanobox.chipx86.com) : il est l'auteur d'un excellent travail sur le comportement des différents navigateurs web, dont la synthèse est donnée en annexe.
- Les auteurs du site http://www.w3.org, source extrêmement riche de renseignements qui explique dans tous leurs détails les normes du World Wide Web Consortium (W3C), ainsi que Jean-Jacques Solari, qui a traduit en français bon nombre de ces documents sur le site http://www.yoyodesign.org. Trois figures sont extraites de ce site, ainsi que les tableaux des propriétés CSS, qui se trouvent en annexe.

Je suis reconnaissant également à mes étudiants de l'EIGSI et de Sup de Co La Rochelle. Qu'ils me pardonnent : je me suis servi de leurs erreurs et de leurs difficultés pour rendre ce livre plus clair et plus pédagogique.

Enfin, c'est de tout mon cœur que je remercie mon épouse et ma fille, pour leur soutien et leur participation.

Francis Draillard

contact@antevox.fr

http://www.antevox.fr

Table des matières

chapitre

1


Jardin Zen CSS
La beauté de la conception CSS
Le chemin vers l'édification
Les reliques passées des sélecteurs spécifique aux navigateurs, des DOMs incompatibles, et du manque de support des CSS encombrent un long chemin sombre et morne.
Aujourd'hui, nous devons nous clarifier l'esprit et nous débarasser des pratiques passées. La révélation de la véritable nature du Web est maintenant possible, grâce aux efforts infatigables des gens du W3C, du WaSP et des créateurs des principaux navigateurs.
Le Jardin Zen css vous invite à vous relaxer et à méditer sur les leçons importantes des maitres. Commencez à voir clairement. Apprenez à utiliser ces techniques (bientôt consacrées par l'usage) de manière neuve et revigorante. Ne faites qu'Un avec le Web.
Alors, de quoi s'agit-il?
Il y a clairement un besoin pour les graphistes de prendre les CSS au sérieux. Le Jardin Zen vise à exciter, inspirer, et encourager la participation. Pour commencer, voyez quelques concepts choisis dans la liste. Cliquez sur n'importe lequel pour le charger sur cette page. Le code HTML demeure le même, et seule la feuille de style extérieure change. Oui, vraiment.
Une demonstration de ce qu'on peut accomplir lorsqu'on utilise les CSS pour la conception web. Sélectionez n'importe quelle feuille de style listée pour charger le résultat sur cette page.
Téléchargez les fichiers d'exemple html et css
Les CSS permettent un contrôle complet et total du style d'un document hypertexte. La seule manière de vraiment démontrer cela d'une manière qui excite les gens est de démontrer ce qui peut vraiment être, une fois que les rennes ont été placées dans les mains de ceux capables de créer la beauté basée sur la forme. Jusqu'à maintenant, les exemples de trouvailles et montages intéressants ont étés fournis par des programmeurs et des structuralistes. Les concepteurs ont encore à faire leurs preuves. Cela doit changer.
Choisissez une conception:
Dazzling Beauty by Deny Sri Supriyono
Dark Rose by Rose Fu
Legge My Ego by Jon Tan
LuGoZee by Viallon Pierre-Antoine
The Diary by Alexander Shabunewicz
Lonely Flower by Mitja Ribic
Mozart by Andrew Brundle
Organica Creativa by Eduardo Cesario
Archives:
Conceptions suivantes »
Voir toutes les conceptions

Introduction au XHTML et aux feuilles de style CSS

Que signifient XHTML et CSS ? Quels avantages nous apportent les feuilles de style ? Comment se partagent-elles le « travail » de mise en page avec le code XHTML ?

Sommaire

Cette introduction nous emmène à la découverte de quelques notions fondamentales à propos du XHTML et des feuilles de style CSS. Ce sera aussi l'occasion, à partir d'exemples, de poser les principes d'une bonne écriture des pages web.

Signification de XHTML et CSS

Le XHTML est une évolution du HTML, initiales de « Hypertext Markup Language », c'est-à-dire *langage de marquage hypertexte*.

Cela signifie que la mise en place d'une page web (titres, paragraphes, images...) utilisera des caractères pour *marquer* d'une certaine façon les différentes parties du texte. Parmi ces caractères de marquage, certains correspondront à des liens vers d'autres pages web : ce sont des liens *hypertexte*.

Le « X » de XHTML vient de XML, soit « eXtensible Markup Language », langage plus complexe et plus strict que le HTML. C'est lui qui a inspiré la transition du HTML vers la forme plus rigoureuse qu'est le XHTML.

Quant à CSS, cela signifie « *Cascading Style Sheets* », ce qui se traduit en français par *feuilles de style en cascade*.

La feuille de style fournit la mise en forme des éléments de la page, qui auront été écrits en XHTML. Elle s'applique à une ou plusieurs pages du site.

Le terme « en cascade » indique que la mise en forme d'une page peut faire appel à plusieurs feuilles de style. Les différentes propriétés affectées à un même élément s'ajoutent alors pour lui donner sa mise en forme finale. Lorsque deux propriétés se contredisent, des règles de priorité s'appliquent et c'est généralement le dernier style défini qui est pris en compte.

Principes de base pour une page web

Voici les principales qualités demandées à une page web : qu'elle soit claire dans sa conception, accessible à tous et que son esthétique s'accorde bien avec son contenu.

Choix sensé des balises HTML

En HTML, chaque élément doit être porteur de sens. Par exemple :

- Pour un titre de page, utiliser un titre de niveau 1 `<h1>` plutôt qu'un paragraphe quelconque `<p>`.
- Pour un menu (liste de liens), choisir une liste sans numérotation `<ul>`.

L'utilisation de balises qui donnent du sens présente plusieurs intérêts :

- le code sera plus clair pour le développeur et la maintenance future du site en sera facilitée ;
- les moteurs de recherche indexeront mieux les pages, car ils y retrouveront plus facilement les mots-clés essentiels ;
- l'accessibilité sera améliorée pour les personnes en situation de handicap.

Adaptation aux navigateurs

Il s'agit de couvrir, autant que possible, une large gamme de navigateurs :

- différents logiciels du marché ;
- divers systèmes d'exploitation ;
- d'autres médias que le PC : assistant personnel ou PDA, téléphone mobile...

De plus, les pages web doivent rester lisibles lorsque la feuille de style n'est pas prise en compte :

- lecture en mode texte ;
- lecture vocale ou en braille ;
- anciens navigateurs qui ne reconnaissent pas les styles.

Figure 1–1 *Nos pages doivent pouvoir s'afficher dans différents navigateurs.*

Accessibilité

L'accès aux personnes handicapées (que le handicap soit visuel, auditif, moteur) doit être facilité :

- Proposez une navigation alternative lorsque sont utilisés des menus graphiques ou reposant sur des scripts (des modules complémentaires, appelés *plug-ins* en anglais, sont nécessaires pour Java, Flash...).
- Évitez les structures de page reposant sur des cadres (*frames*) ou des tableaux (réservez les tableaux à la présentation de données en lignes ou en colonnes).
- Ne vous basez pas uniquement sur les couleurs, permettez l'augmentation de la taille du texte.
- Proposez une alternative aux contenus purement visuels (images) ou auditifs, facilitez la lecture des liens hypertexte...

L'apparence, fonction du thème et du public concerné

Le choix des couleurs et des polices de caractères est fonction du style à donner aux pages web, donc de leur thème et du public visé.

Polices de caractères

- Pour le web, utilisez plutôt des polices sans sérif (Arial, Helvetica, Trebuchet, Verdana...).
- Réservez aux titres les autres polices et les polices fantaisie.
- N'abusez pas de l'italique, à réserver pour quelques mots ou remarques.
- Évitez les caractères trop petits pour des paragraphes entiers.
- Limitez à deux ou trois le nombre de polices différentes dans une même page.

Figure 1–2 *Choisissez des polices lisibles et harmonieuses : ne suivez pas ce mauvais exemple !*

En résumé, quelques sentiments liés aux couleurs

Les différentes couleurs correspondent à des sentiments, des impressions, des atmosphères. Cela peut nous aider à choisir le graphisme du site à créer, en fonction de son sujet et de la catégorie d'internautes auquel il est destiné. Voici les valeurs communément associées aux couleurs les plus courantes :

- Les couleurs chaudes, telles que le jaune, l'orange et le rouge, représentent la chaleur et le dynamisme, ainsi que les impulsions.
- Les couleurs froides, comme le gris, le bleu, le vert et le violet, indiquent la fraîcheur, le calme et aussi le raisonnement (sciences).
- Les couleurs vives sont associées à l'action.
- Les couleurs pastel font penser à la poésie et donnent une impression de sensibilité.
- Enfin, le gris et le blanc sont des couleurs passe-partout.

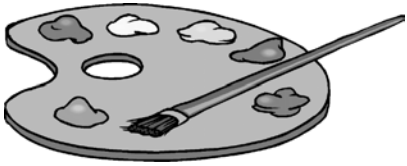


FIGURE 1–3 *Bien choisir les couleurs d'une page*

Homogénéité du site

Les différentes pages d'un site doivent présenter un minimum d'homogénéité entre elles. Elles proposeront par exemple des variations autour d'un graphisme commun.

Il est donc important de définir une « charte graphique » (polices, couleurs, logos...) à partir de laquelle les pages seront construites.

Principes d'une bonne écriture XHTML/CSS : donner du sens au codage

L'essentiel est de séparer le contenu (codé en XHTML) de la mise en forme (feuilles de style CSS). Cette méthode présente plusieurs avantages, notamment la clarté du code et la possibilité de définir des styles communs à plusieurs pages.

Voici quelques exemples de mise en forme à l'aide de balises qui donnent du sens au texte. Ils utilisent le principe des balises, que nous n'avons pas encore détaillé (c'est l'objet du chapitre qui suit). Cependant c'est une première approche intéressante, avant de se jeter dans le grand bain du XHTML et des CSS !

Titre de page

Au lieu d'écrire le titre dans un paragraphe normal <p> et de l'affubler d'une tonne de mises en forme (grande taille, gras, espacement au-dessus et en

dessous), codez-le plutôt comme « titre de niveau 1 » avec la balise <h1> et, si besoin est, complétez sa mise en forme à l'aide d'une règle de style CSS :

- dans le code XHTML : `<h1>Ici un titre</h1>`
- et dans la feuille de style : `h1 { ...mise en forme...}`

Mise en gras ou en italique

Pour mettre des mots en gras ou en italique, il existait en HTML les balises <b> (gras) et <i> (italique) qui sont obsolètes en XHTML : remplacez-les respectivement par et .

L'affichage sera identique avec les anciennes et les nouvelles balises, alors pourquoi ce changement ? Il correspond au raisonnement suivant :

- Les nouvelles balises et indiquent une mise en relief plus ou moins prononcée sans dire par quel moyen elle s'effectuera, notamment sans imposer le gras ou l'italique comme le font les anciennes <b> (« gras » se dit *bold* en anglais) et <i>.
- Ainsi par exemple, le concepteur de la page peut effectuer cette mise en forme par un changement de couleur du texte, sans recourir au gras ni à l'italique. Ce choix serait en contradiction avec le nom des anciennes balises, il ne l'est pas avec celui des nouvelles.

Liste de liens hypertexte (menu)

Pour écrire un menu, évitez d'avoir recours à une succession de balises <p>. Préférez-leur une structure de liste en délimitant l'ensemble par la balise <ul> (liste non numérotée), et chaque ligne par une balise <li>. Nous reviendrons bientôt sur l'utilisation de ces balises.

Ainsi, cette partie se différenciera du texte et constituera un ensemble homogène, avec une fonction bien précise : celle d'un menu. Par contre, si la suite de la page contient des paragraphes <p> incluant aussi des liens hypertexte, ceux-ci seront bien repérés comme représentant le texte de la page.

> À NOTER **Menu en début de page**
>
> Puisque nous parlons de menu, il faut signaler que sa place de choix dans le code se trouve au début de la page. Celle-ci sera alors mieux comprise par les navigateurs qui lisent la page en mode texte, d'où une accessibilité améliorée pour les personnes handicapées et un meilleur référencement par les moteurs de recherche.

FIGURE 1–4 *Cet extrait de page contient en haut un titre de niveau 1 (nom du site), à gauche une liste de liens (menu général) et sur la droite un paragraphe de texte. Il provient de la page d'accueil du site http://www.framasoft.net, portail de la communauté francophone des logiciels libres.*

Intérêt des feuilles de style

L'utilisation des feuilles de style n'a pas pour seul but de répondre aux normes et de faire plaisir au W3C (consortium qui définit les règles de codage des pages web : http://www.w3.org). Un bénéfice réel et concret découle de cette façon de travailler.

La dissociation du contenu (XHTML) et de la mise en forme (feuille de style) permet :

- de retrouver et corriger plus facilement le texte des pages ;
- d'utiliser une feuille de style externe, commune aux différentes pages d'un site. Il en résulte une meilleure unité graphique entre ces pages et aussi des mises à jour plus simples par la suite. Une modification dans la feuille de style externe se répercute d'un seul coup sur toutes les pages du site.

La mise en page est beaucoup plus légère, car elle ne nécessite plus l'utilisation de tableaux. Les CSS permettent en effet de positionner les différentes partie d'une page web :

- soit de façon rigoureuse : blocs fixes dont les coordonnées sont choisies ;
- soit d'une manière souple : blocs flottants qui s'alignent les uns par rapport aux autres.

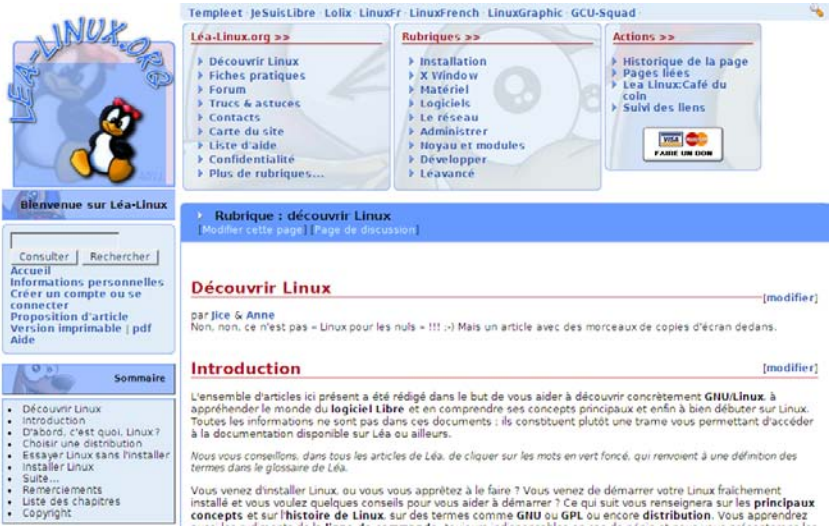


Figure 1–5 *Une page web est constituée de blocs contenant du texte et des images (extrait de la page http://lea-linux.org/cached/index/Intro-index.html, site francophone d'entraide pour les utilisateurs de Linux).*

Cette présentation était l'occasion de connaître quelques principes de base auxquels nous ferons référence plus tard. Le chapitre qui suit va nous donner des outils concrets pour écrire une page XHTML et nous préparer à la mise en place d'une feuille de style.

chapitre

2

L'essentiel du XHTML

Qu'est-ce qu'une page web et quelle en est la structure ? Quelles sont les règles d'imbrication des balises XHTML ?

Sommaire

Dans le contenu d'une page web, chaque balise XHTML représente un élément. Après avoir étudié l'utilisation de ces balises, nous nous intéresserons à leur imbrication : un élément est toujours inclus à l'intérieur d'un autre, ne serait-ce que celui qui délimite et englobe le contenu de la page. La hiérarchie entre ces éléments est importante à comprendre pour bien utiliser les feuilles de style.

Dans ce chapitre nous préciserons quelques termes, essentiels pour comprendre les feuilles de style.

Rappel sur le principe des balises

HISTOIRE **Au départ était le HTML**

Le langage HTML, qui a maintenant évolué vers le XHTML, fut inventé en 1990 par Tim Berners-Lee lorsqu'il travaillait au CERN de Genève (Centre Européen de Recherche Nucléaire). L'objectif était alors d'afficher des pages d'information ayant les propriétés suivantes :

- les pages étaient reliées entre elles par des liens hypertexte (ces liens sur lesquels, aujourd'hui, nous n'arrêtons pas de cliquer !) ;
- ces documents devaient être lisibles sur tous les ordinateurs, quel que soit leur système d'exploitation : Windows, Unix, Mac OS, etc.

C'est pour constituer un ensemble de pages reliées entre elles, accessibles de n'importe quelle machine sur le réseau, que le HTML fut créé, sur le principe du codage de portions de texte à l'aide de balises. Cette méthode reste bien sûr d'actualité, les pages du Web étant exclusivement codées avec des caractères.

Les différentes parties de texte sont donc délimitées par des « balises », qui donnent des indications de mise en forme ou de structuration, l'une d'elles (<a> comme *ancre*) servant à relier les pages entre elles par les fameux liens hypertexte.

Par exemple, la mise en italique du mot « bonjour » s'écrit :

```
<em>bonjour</em>
```

> Normes **La balise <i> est obsolète, vive la balise !**
> Difficile de résister à l'usage de cette balise `<i>` qui est si simple et intuitive : **i** comme **i**talique ! Elle reste utilisée en HTML, mais en XHTML c'est la balise `<em>` (emphase), qu'il faut employer. Elle est plus générale et indique une mise en relief du texte.
> Ainsi, si cette balise `<em>` produit par défaut de l'italique, libre au concepteur d'indiquer dans sa feuille de style que cette mise en relief ne se fera pas en italique mais, pourquoi pas, en jaune sur fond rose... C'est très gai, non ?

Autre exemple d'élément (ici un titre de niveau1) :

```
<h1>Premier chapitre</h1>
```

Les mots ainsi encadrés par la balise `<h1>` seront graissés, écrits en gros caractères et plus clairement séparés du texte qui l'entoure.

Du HTML au XHTML

Le XHTML est une évolution du HTML dans laquelle la syntaxe est plus rigide, mais plus rigoureuse : cela simplifie beaucoup la maintenance des pages web.

Aussi, rien de compliqué dans le passage du HTML 4 au XHTML 1. Il suffit d'adopter quelques règles de notation et surtout d'utiliser les feuilles de style CSS 2 (*Cascading Style Sheets*) qui complètent cette norme, pour ne pas mélanger le contenu du texte et sa mise en forme. D'ailleurs, ce livre est là pour vous y aider !

NORMES **Méli-mélo de versions : HTML 5, XHTML 2, ou XHTML 5 ?**

Le HTML en était à sa version 4.0 lorsqu'il a été remplacé par le XHTML version 1.0 puis 1.1. Mais c'était sans doute trop simple, car une nouvelle norme HTML 5 est en préparation, avec une déclinaison en XHTML 5. Est-ce un retour en arrière vers le HTML ? Et le futur XHTML 2 ne serait-il plus d'actualité ? Si, si, pensez-donc ! Deux versions de XHTML pour le prix d'une ! Pour s'y retrouver dans toutes ces versions :

- Le XHTML 2 sera utilisé pour des applications particulières, son absence de compatibilité avec les versions antérieures le rendant délicat à employer pour des pages web : celles-ci doivent rester lisibles sur le plus grand nombre de navigateurs, y compris anciens.
- Si le HTML revient sous sa version 5, c'est notamment pour les besoins des éditeurs *Wysiwyg* (*what you see is what you get*). La mise en forme s'y fait au jugé avec la souris, sans écrire de code, et ces logiciels mélangent donc le contenu et la mise en forme.
- C'est finalement le XHTML 5 qui sera le successeur naturel du XHMTL 1 : comme lui, il nous permettra d'obtenir un code propre et des pages faciles à modifier. Il reprend l'essentiel de son prédécesseur et y ajoute quelques balises supplémentaires.

Pour en savoir plus sur leurs avantages respectifs :

▸ **http://xhtml.com/fr/future/x-html-5-versus-xhtml-2/**

Premières règles d'écriture XHTML

Familiarisons-nous avec les principes de base qui valent pour l'écriture d'un fichier XHTML : son nom d'abord, puis la façon d'écrire son contenu.

Règles pour les noms des fichiers

Chaque page web est un fichier dont le nom peut comprendre des lettres, des chiffres et des tirets. À éviter : les espaces, les caractères accentués et le « *ç* ». Son extension est généralement « `.html` ».

La première page du site et de chacun de ses sous-dossiers doit être nommée `index.html` (ou `index.htm`). En effet, c'est cette page de nom *index* qui s'affichera par défaut si l'internaute tape l'adresse de votre site sans préciser le nom du fichier, comme dans www.votresite.com. S'il n'existe pas de page nommée *index*, l'internaute ne verra alors qu'une liste de fichiers et de sous-dossiers, façon explorateur de fichiers...

> Remarque **Arborescence du site**
>
> Lorsque le site comprend des sous-niveaux, notez qu'il faut saisir dans la barre d'adresses le séparateur « / », habituel également dans le monde Unix/Linux, tandis que l'explorateur de fichiers sous Windows utilise la barre oblique inverse ou *antislash* « \ »...

Règles d'écriture des balises XHTML

La norme XHTML impose les règles suivantes :

- Les balises s'écrivent toujours en minuscules.
- Chaque balise doit être refermée.

> Normes **Fermeture des balises simples**
>
> En XHTML, toutes les balises doivent être fermées, y compris celles qui, n'entourant pas de contenu, ne s'écrivent pas par paires. Leur barre de fermeture est alors intégrée dans la balise elle-même :
> - Saut de ligne : `<br />` au lieu de `<br>`
> - Tracé d'une ligne horizontale : `<hr />` au lieu de `<hr>`
> - Image : `<img src="logo.gif" />` au lieu de `<img src="logo.gif">`

- Les attributs des balises sont à écrire entre guillemets simples ou doubles. Par exemple la balise qui affiche l'image fournie par le fichier `logo.gif` s'écrit :

  ```
  <img src="logo.gif" />
  ```

- S'il y a imbrication de balises, leur ordre de fermeture est inverse à celui d'ouverture :

  ```
  <h1>...<em>...</em></h1>
  ```

Structure d'une page XHTML

Une page HTML ou XHTML s'écrit de la façon suivante :

- Sur la première ligne, la balise **<!DOCTYPE ...>** indique la version de (X)HTML utilisée.
- Le reste de la page est encadré par des balises **<html>** et **</html>** qui signifient *début* et *fin* de HTML.
- Entre ces deux balises se trouvent deux parties : l'en-tête de la page entre **<head>** et **</head>** et le contenu (le corps) de la page entre **<body>** et **</body>**.

Structure générale d'une page XHTML

```
<!DOCTYPE ... >
<html>
  <head>
    <title>Titre affiché dans la barre du navigateur
    </title>
              ...
  </head>
  <body>
    ...Contenu de la page...
  </body>
</html>
```

La balise `<!DOCTYPE ...>` étant longue et un peu barbare, elle n'est pas écrite ici en entier. Ne vous inquiétez pas pour autant : un simple copier-coller nous fournira le bon `DOCTYPE` et nous y reviendrons plus loin. Notez que cette balise est la seule à s'écrire en majuscules, toutes les autres sont en minuscules.

Délimité par les balises `<head>` et `</head>`, l'*en-tête* donne des informations qui ne seront pas visibles dans la page web, sauf la balise `<title>` qui fournit le titre de la page affiché dans la barre de titre, tout en haut de la fenêtre du navigateur. Les autres balises de l'en-tête indiquent la langue et le codage utilisés, les styles (feuilles de style CSS), etc. Nous les détaillerons plus loin également.

Tout le contenu visible dans le navigateur, le texte comme les liens vers les images, se trouve dans le *corps de la page* entre les balises `<body>` et `</body>`.

Espaces, sauts de ligne et commentaires invisibles

Dans votre fichier d'édition, vous pouvez sauter des lignes et aérer le code XHTML à votre guise : quel que soit le nombre d'espaces se succédant, le navigateur saura n'en afficher qu'un seul. Quant aux sauts de ligne, il les ignorera.

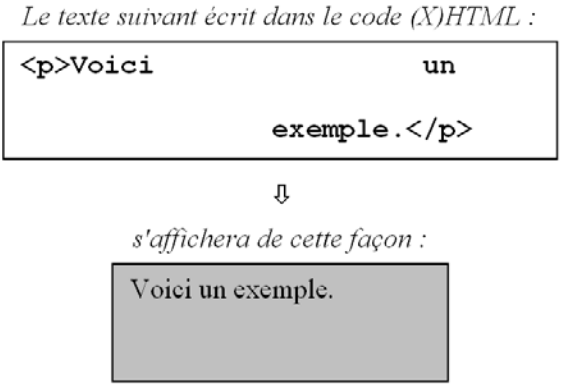


Figure 2–1 *Les suites d'espaces et les sauts de ligne tapés dans le code XHTML sont ignorés lors de l'affichage.*

Pour créer un saut de ligne qui sera effectivement affiché, il faut utiliser la balise `<br />` (comme *break*).

Pour forcer l'affichage de plusieurs espaces successifs sur une ligne, il faut utiliser le caractère spécial ` ` pour *non breakable space*, ou *espace insécable* en français .

> **Syntaxe** **Caractères spéciaux**
>
> Attention à ne pas omettre le point-virgule à la fin de vos caractères spéciaux ainsi que l'esperluette (&) au début.

D'autre part, il est toujours utile de placer des *commentaires* dans le code XHTML, pour s'y retrouver plus tard. Ils ne seront pas affichés par le navigateur, mais constituent une aide pour celui qui écrit et lit le code source de la page. Ils peuvent être écrits sur une ou plusieurs lignes et sont délimités par les balises `<!--` et `-->`.

Exemples de commentaires

```
<!-- Ici un commentaire pour le créateur de la page -->

<!--
Et là un autre exemple de commentaire,
sur plusieurs lignes cette fois,
toujours à l'attention du programmeur.
-->
```

Principales balises XHTML

L'expérience montre que dans l'utilisation d'un logiciel de traitement de texte, par exemple, ce sont toujours les mêmes fonctions de base qui sont utilisées et qui permettent de satisfaire la majorité des besoins.

De même en XHTML, la connaissance de quelques balises simples nous suffira pour une première approche et dans la plupart des utilisations courantes.

Un exemple pour commencer

Avant de détailler ces balises essentielles utilisées en XHTML, voici un exemple concret et illustré qui nous permettra d'en découvrir quelques-unes en avant-première.

Exemple de page codée en XHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" (1)
          "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr"> (2)

<head>
    <meta http-equiv="content-type"
          content="text/html; charset=utf-8" /> (3)
    <title>Blog de Vincent THYME</title> (4)
</head>
```

```
<body>
    <h1>Bienvenue chez Vincent THYME</h1> 5

    <h2>Mon blog à quatre sous</h2> 6
    <p> 7
       Voici quelques petites lignes à l'attention de
       mes visiteurs. Je voudrais partager avec vous
       <em>mes passions, mes idées, mes projets</em>...
    </p>

    <h2>Mes activités préférées</h2> 8
    <ul> 9
        <li>Le surf</li>
        <li>La plongée sous-marine</li>
        <li>L'informatique</li>
    </ul>
</body>

</html>
```

La figure 2-2 montre le résultat de cette page intitulée *Blog de Vincent Thyme*. Nous allons étudier et commenter les balises qui la composent.

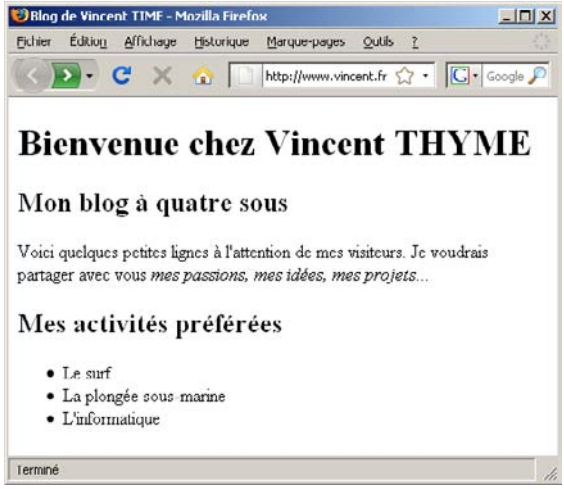


FIGURE 2–2 *Le blog de Vincent : une première page toute simple en XHTML*

Les deux premières balises

❶ `<!DOCTYPE ...>` nous indique que cette page est codée en XHTML Strict, version 1.0. Cette balise est trop compliquée à retenir : elle ne s'écrit pas, elle se colle à partir d'un modèle !

> NORME **Les « DOCTYPE »**
> La liste des `<!DOCTYPE ...>` officiels se trouve sur le site du *W3C*. Ces prologues sont aussi expliqués sur le site *pompage.net*.
> - **http://www.w3.org/QA/2002/04/valid-dtd-list.html**
> - **http://www.pompage.net/pompe/doctype/**

❷ La deuxième balise pourrait s'écrire simplement `<html>`, mais il est conseillé d'utiliser les deux attributs mentionnés ici : le premier donne un lien vers la liste des balises XHTML et le deuxième précise la langue utilisée, ici `fr` pour le français.

Là encore, un copier-coller s'impose pour cette deuxième et dernière balise un peu complexe. Notez que la balise de fermeture, tout à la fin du fichier, s'écrit simplement `</html>`.

En-tête

❸ La première ligne de l'en-tête est nécessairement la balise `<meta ...>` qui indique le *type de codage utilisé*, ici `utf-8`. Si le fichier XHTML est bien enregistré suivant ce codage (voir les options du logiciel utilisé pour écrire ce fichier), il sera possible de taper tels quels les accents, le ç et le symbole €, sans avoir à les coder avec des *entités HTML*, comme `&euro;` pour le caractère €.

❹ La balise `<title>` permet d'écrire un *titre* qui apparaîtra dans la barre de titre du navigateur.

Corps de la page

❺ La balise `<h1>` met en forme le *titre de niveau 1* : grande taille, gras, espaces au-dessus et en dessous.

❻ / ❽ Les balises `<h2>` correspondent à des *sous-titres* : taille un peu moins grande, un peu moins d'espacement autour.

❼ Le *paragraphe* ** ** permet d'écrire une phrase. Quelques mots sont en *italique* à la fin de celle-ci, grâce à la balise **** imbriquée à l'intérieur des balises ... .

❾ Enfin, la page se termine par une *liste à puces*, indiquée par la balise **<ul>**, chaque *ligne* étant repérée par une balise **<li>**, ce qui signifie retour à la ligne et nouvelle puce.

Quittons maintenant ce premier exemple pour présenter en détail les balises XHTML qui sont le plus utiles.

Paragraphes et titres

Chaque *paragraphe* est encadré par les balises ** ... **. Un paragraphe ne contient jamais d'autres paragraphes, mais peut inclure des balises de mise en forme, comme le gras ou l'italique, des liens hypertexte et des images.

Les *titres* sont des balises commençant par **h** comme *header*, c'est-à-dire *en-tête* en anglais. **<h1>**, **<h2>** et **<h3>**, ... correspondent à différents niveaux de titre : **<h1>** pour un titre de niveau 1, **<h2>** pour un sous-titre de niveau 2, etc.

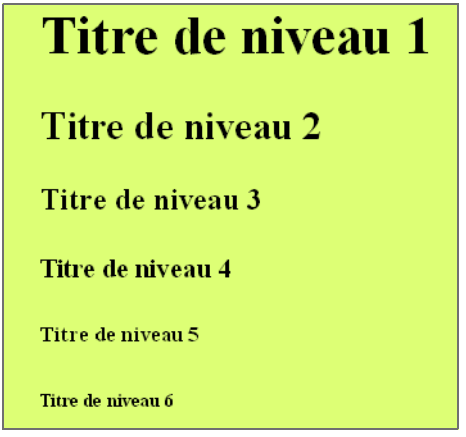


FIGURE 2–3 *Différences de taille pour les niveaux de titre 1 à 6, encadrés par les balises <h1> ... </h1> jusqu'à <h6> ... </h6>.*

> À NOTER **Niveaux de titre courants**
>
> Il est rare d'utiliser des niveaux de titre au-delà de 3 ou 4. Si ces balises existent jusqu'à <h6>, le titre <h4> correspond à un texte de taille normale mis en gras, <h5> et <h6> donnant des caractères plus petits. La figure 2-3 donne un aperçu des tailles associées aux différents niveaux de titre.

Mise en forme commune à une partie du texte

Pour *regrouper en un seul bloc* un ensemble de paragraphes, de titres, etc., il suffit de les encadrer avec les balises **`<div> .... </div>`** (*div* comme une *division* ou partie d'une page). Le bloc ainsi constitué peut être encadré, mis en forme ou positionné dans la page, à l'aide des styles CSS qui seront abordés dans les chapitres suivants.

Les balises **`<span> ... </span>`** permettent de *regrouper plusieurs mots* d'un paragraphe et de leur donner une mise en forme commune, par exemple la taille, la couleur ou la police de caractère. Le mot anglais *span* signifiant *portée*, cela revient à sélectionner une *partie du texte* à mettre en forme, quelques mots ou quelques lettres. Il ne s'agit pas ici d'un bloc de texte dont la position est définie dans la page, mais d'un groupe de mots ou de lettres qui conserve sa place à l'intérieur d'un paragraphe.

> À NOTER **Balises génériques**
>
> Les balises **`<div>`** et **`<span>`** sont appelées *balises génériques*, parce qu'elles n'ont pas de sens en elles-mêmes. Pour respecter l'esprit du XHTML, il est préférable de ne pas en abuser et d'utiliser autant que possible des balises qui ont une signification, donnant une indication sur l'importance ou le rôle du contenu :
>
> - **`<strong>`** ou **`<em>`** sera plus approprié que **`<span>`** pour une mise en relief du texte, même si ce n'est pas du gras ou de l'italique
> - **`<h1>`** ou **`<h2>`**, ... pour un titre, **`<p>`** pour un paragraphe unique, remplaceront avantageusement un **`<div>`** s'il peut être évité.

Italique et gras

L'*italique* s'obtient avec la balise ** ... ** qui signifie en anglais *emphasis*, c'est-à-dire *accent* ou *insistance*. Elle n'entraîne pas de retour à la ligne, ce n'est pas un « bloc » de texte.

Pour mettre un terme en **gras**, il suffit de l'encadrer avec la balise ** ... **, comme *stronger emphasis*, soit encore plus en relief qu'avec la balise .

> À NOTER **Modifier la mise en forme par défaut**
>
> Les balises et correspondent par défaut à l'italique et au gras. Mais grâce aux styles CSS, vous pouvez décider de remplacer ces mises en forme. Par exemple :
> - l'italique fera place à un texte en vert, police Arial ;
> - le gras sera remplacé par du blanc sur fond bleu, encadré de rouge.
>
> Dans les deux cas, l'idée de mettre le texte en relief est respectée, mais avec d'autres apparences que le gras et l'italique. Cette modification de la mise en forme par défaut sera d'ailleurs possible pour toutes les balises XHTML.

Liens hypertextes

La balise **<a....>...** permet d'écrire un *lien hypertexte* : le curseur prend la forme d'une main au passage de la souris sur ce lien, et un clic permet d'afficher la page qui est référencée dans cette balise. Cette balise est dite *en ligne*, car elle ne produit pas de retour à la ligne.

Exemples simples de liens

```
<a href="http://www.antevox.fr/livre"> 1
         Exemples du livre à télécharger
</a>

<a href="index.html">Retour à la page d'accueil</a> 2
<a href="documents/liste.html">Documents à télécharger</a> 3
```

L'attribut **href** (*hypertext reference*) est obligatoire, puisqu'il indique l'adresse de la page à afficher lors d'un clic sur ce lien.

Cela peut être *une adresse internet* (1) ou *un nom de fichier seul* (2), lorsque la page à atteindre se trouve située dans le même dossier que la page en cours d'affichage.

Si le fichier est dans un sous-dossier, il faut taper le nom de ce dossier suivi d'un séparateur : / comme sous Unix ou Linux, et non \ comme dans Windows. Le dernier exemple (3) affiche le fichier `liste.html` qui se trouve dans le sous-dossier `documents` du dossier courant, c'est-à-dire du dossier qui contient la page affichée.

Lien avec attributs `accesskey` et `title`

```
<a href="http://www.antevox.fr"
   accesskey="a" title="Site de l'auteur">
      Visitez le site de l'auteur
</a>
```

Il est très utile de faire figurer l'attribut **`accesskey`** : les personnes handicapées pourront activer le lien par un appui simultané sur la touche *Alt* et la lettre indiquée entre guillemets.

L'attribut **`title`** permet d'afficher un texte automatiquement dans une bulle au passage de la souris sur le lien, comme le montre la figure 2-4.

FIGURE 2–4 *Lien hypertexte : au passage de la souris, le curseur prend la forme d'une main et le contenu de l'attribut title s'affiche dans une bulle.*

Lien hypertexte vers un endroit de la page

```
<a href="#poissons"
   accesskey="p" title="Les habitants de la mer">
      Accès au paragraphe "Les habitants de la mer"
</a>
...
...
<h2 id="poissons">Les habitants de la mer</h2>
```

Pour mettre en place un lien vers un endroit précis de la page courante, il suffit d'ajouter un identifiant en attribut de la balise *destination*, par exemple **`id="toto"`**, ce qui permet de l'atteindre directement grâce au lien `<a href="#toto" ...>...</a>`.

Si le texte à relier au lien en question n'est pas encadré par une balise, il est possible d'utiliser la balise **`<a>`** comme ancre simple, uniquement pour attribuer un identifiant à cette partie du texte :

```
<a id="toto">Texte à relier au lien interne</a>
```

Cette méthode permet également de créer un lien vers un endroit précis d'une autre page que celle affichée. Par exemple, le lien `<a href="oiseaux.html#rossignol">` affichera la page `oiseaux.html` et placera le curseur de la fenêtre sur la balise d'identifiant `id="rossignol"`.

> À NOTER **Dièse # - pas dièse**
>
> Le lien `<a href="#toto" ...> ... </a>` contient un *dièse* # avant le nom de l'identifiant, alors que la balise destination contient un attribut `id="toto"` sans dièse. Bien que cette méthode d'adressage soit très simple, au sujet du dièse ce bémol s'imposait !

Par ailleurs, la balise <a> permet d'autres types de liens que ceux vers une page web, en particulier des *liens de contact* vers une adresse de messagerie, comme le montre l'exemple ci-dessous.

Lien vers une adresse de messagerie

```
<a href="mailto:contact@antevox.fr"
   accesskey="m"
   title="Envoi d'un courriel à l'auteur">
    Pour contacter l'auteur...
</a>
```

Dans cet exemple, l'attribut **`href`** contient **`mailto:`** suivi de l'adresse électronique vers laquelle sera envoyé le courriel.

Un clic sur ce lien ouvre le *logiciel de messagerie par défaut* sur l'ordinateur du visiteur, et remplit la rubrique *Destinataire* avec l'adresse fournie.

Il est possible d'ajouter un sujet et d'autres destinataires, avec le séparateur **?** après l'adresse courriel et le séparateur **&** par la suite, avant chaque nouveau paramètre :

```
<a href="mailto:toto@laposte.net?subject=Essai de
                message&cc=titi@laposte.net">
    Envoyer un message
</a>
```

Dans l'exemple précédent, le *destinataire* du message est `toto@laposte.net`, le *titre* du message est « Essai de message », et il y a un destinataire en *copie* (cc comme copie carbone) : `titi@laposte.net`.

ATTENTION **Protégez du spam vos adresses de courriel !**

Les adresses ainsi inscrites dans une page web peuvent être lues par des logiciels automatiques, qui balaient les pages pour remplir des bases de données d'adresses électroniques à l'usage des *spammeurs* : ceux qui envoient du *spam*, ces messages publicitaires indésirables également appelés *pourriels*.

Il faut donc protéger les adresses :

- soit en codant la balise en *javascript*, programme que ces logiciels n'interprètent pas mais qui est normalement pris en compte par les navigateurs ; une recherche sur Internet vous fournira rapidement différentes solutions, comme celle proposée par ce site autour de la messagerie électronique :
  - **http://www.arobase.org**
- soit *au minimum* en modifiant l'adresse, que l'internaute rectifiera à la main dans la fenêtre de sa messagerie, par exemple sous cette forme :
  `<a href="mailto:titi@pasdespam-laposte.net">`
  ou encore celle-ci :
  `<a href="mailto:titi-chez-laposte.net">`.

Listes à puces ou numérotées

Une énumération gagne en clarté lorsque chacun des points est repéré par une puce ou un numéro. De telles listes sont délimitées par les balises :

- `<ul>...</ul>` comme *unordered list* pour une *liste à puces* ;
- `<ol>...</ol>` comme *ordered list* pour une *liste numérotée.*

Dans les deux cas, *chaque ligne* est repérée par `<li>...</li>` à l'intérieur de ces balises.

Exemples de listes

```
<p>Liste à puces :</p>
<ul>
   <li>Premièrement</li>
   <li>Deuxièmement</li>
   <li>Troisièmement</li>
</ul>

<p>Liste numérotée :</p>
<ol>
   <li>Premièrement</li>
   <li>Deuxièmement</li>
   <li>Troisièmement</li>
</ol>
```

L'affichage correspondant à ce code XHTML est donné par la figure 2-5.

Figure 2–5 *Les deux types de listes : liste à puces et liste numérotée.*

VARIANTES **Personnalisation des listes**

Comme nous le verrons plus loin, ces listes pourront être personnalisées grâce à l'emploi des feuilles de style CSS :

- choix du type de puce, celle-ci pouvant éventuellement être une image ;
- modification de la numérotation, variantes avec différents types de chiffres ou avec des lettres.

Tableaux

En XHTML, un tableau est délimité par les balises **`<table> ... </table>`**.

Initialement, les bordures du tableau sont invisibles, donc il faut ajouter dans la balise `<table>` l'attribut **`border`** pour indiquer l'épaisseur de la bordure en pixels.

Également inclus dans la balise `<table>`, les attributs **`cellspacing`** et **`cellpadding`** permettent de préciser en pixels, respectivement *l'espacement entre cellules* (traits d'encadrement simples si 0, doubles sinon) et *les marges intérieures* des cellules.

Le tableau se construit ligne par ligne, avec les balises **`<tr> ... <tr>`** comme *table row* en anglais, c'est-à-dire *rangée du tableau*.

Les cellules du tableau sont définies à l'intérieur de chaque ligne, grâce aux balises **`<td> ... </td>`** comme *table data*, soit *donnée du tableau*. Lorsqu'il s'agit d'une cellule de titre, il est possible d'utiliser à la place les balises **`<th> ...</th>`** comme *table header* ou *en-tête du tableau* : le texte est alors mis en gras et centré.

Les balises `<caption> ... </caption>` sont facultatives : placées sous la balise de début de tableau `<table>`, elles permettent de définir un titre associé au tableau, qui s'écrira par défaut au-dessus de celui-ci.

FIGURE 2–6 *Structure d'un tableau en XHTML.*

Exemple de tableau

```
<table border="1" cellspacing="0">

  <tr> ❶
    <th>Pays</th>
    <th>Capitale</th>
  </tr>

  <tr> ❷
    <td>France</td>
    <td>Paris</td>
  </tr>

  <tr> ❸
    <td>Espagne</td>
    <td>Madrid</td>
  </tr>

</table>
```

Cet exemple affiche un tableau à trois lignes (trois balises `<tr>`) et deux colonnes (deux balises `<td>` ou `<th>` par ligne). La première ligne ❶ contient deux cellules d'en-tête, les deux autres lignes ❷ et ❸ contiennent les données du tableau. La figure 2-7 montre le résultat obtenu avec ce code.

Pays	Capitale
France	Paris
Espagne	Madrid

FIGURE 2–7 *Premier exemple de tableau : une ligne de titre (balises <th>) et deux lignes de données (balises <td>).*

Fusionner des cellules

L'attribut **`colspan`** permet de fusionner horizontalement les cellules de plusieurs colonnes. Par exemple, la balise `<td colspan="3">...</td>` sera équivalente à trois cellules `<td>...</td>` à l'intérieur d'une ligne.

De même, l'attribut **rowspan** sert à fusionner verticalement les cellules de plusieurs lignes. Une balise `<td rowspan="4">...</td>` sera équivalente à quatre cellules dans le sens vertical : cela signifie que les trois lignes qui suivent auront une cellule `<td>...</td>` en moins.

L'exemple suivant montre le codage d'un tableau qui comprend des cellules fusionnées, horizontalement et verticalement :

```
<table border="1">
  <tr> 1
    <th colspan="2">Infos pays</th>
  </tr>

  <tr>
    <th>Pays</th>
    <th>Langue</th>
  </tr>

  <tr>
    <td>Andorre</td>
    <td>catalan</td>
  </tr>

  <tr> 2
    <td rowspan="2">Canada</td>
    <td>anglais</td>
  </tr>

  <tr> 3
    <td>français</td>
  </tr>
</table>
```

Comme le montre la figure 2-8, la première ligne 1 de ce tableau contient une *fusion horizontale* de cellules, *sur deux colonnes* : une seule balise `<th>` au lieu de deux sur cette ligne.

L'avant-dernière ligne 2 contient une *fusion verticale* de cellules, *sur deux lignes* : la ligne suivante 3 ne contient donc qu'une seule balise `<td>` au lieu de deux.

Figure 2–8 *Deuxième exemple de tableau : il comprend des cellules fusionnées, horizontalement et verticalement.*

> Normes **Éviter les tableaux pour la mise en page**
>
> Rappelons qu'en XHTML, les tableaux servent uniquement à présenter des données. En effet, ils sont tout à fait déconseillés lorsqu'il s'agit de placer côte à côte des blocs de texte : nous étudierons plus loin comment positionner des blocs de texte dans une page en utilisant les propriétés CSS adéquates.

Insertion d'images

En XHTML, insérer une image revient à placer un *lien vers le fichier qui la contient*, avec la balise `<img ... />`. Elle contient obligatoirement l'attribut `src` qui indique le nom du fichier image à afficher.

Exemple de balise image

```
<img src="tux.png" alt="Le pingouin Tux"
                   title="Tux est la mascotte de Linux" />
```

Si l'attribut `src` ne contient qu'un nom de fichier, cela signifie que ce fichier image se trouve dans le même dossier que la page web qui l'utilise.

Si cette image se trouvait dans le sous-dossier *images*, la balise `img` s'écrirait :

```
<img src="images/tux.png" alt="..." title="..." />
```

NORMES **Fermeture de la balise `img`**

Vous souvenez-vous qu'en XHTML, toute balise doit être fermée ? Or, la balise **`img`** est seule ! Alors, sa barre de fermeture est intégrée : elle se trouve juste avant le chevron fermant, précédée par un espace. Cette règle vaut pour toutes les balises qui s'utilisent seules.

Il est important de renseigner l'attribut **`alt`** d'une balise image : c'est un *texte de remplacement* pour ceux qui naviguent en mode texte, notamment les handicapés visuels. C'est aussi un moyen de renseigner les moteurs de recherche.

NORMES **Contenu de l'attribut `alt`**

La balise **`alt`** doit indiquer, de façon concise, le *contenu* de l'image et sa *fonction* si elle en a une (par exemple, si l'image est un bouton d'action). Il est superflu d'y écrire « image de... », « graphisme de... » ou « photo de... ».
Pour une image décorative qui n'a pas de sens en elle-même, ou si une légende est associée à l'image, l'attribut `alt` peut être vide mais reste obligatoire : `alt=""`.

Quant à l'attribut **`title`**, il permet d'afficher un *texte dans une bulle* au passage de la souris sur l'image : c'est un renseignement complémentaire, qui n'a pas le but de décrire l'image comme la balise `alt`.

ATTENTION **Lorsque `alt` peut devenir une bulle...**

Si l'attribut `title` n'est pas présent, aucune bulle ne doit s'afficher au passage de la souris sur l'image... en principe !
C'est sans compter avec l'interprétation personnelle des normes par Internet Explorer jusqu'à sa version 7 : en l'absence de l'attribut `title`, il affiche en tant que bulle le contenu de l'attribut `alt` de l'image. Un détail à connaître pour maîtriser ce qui va s'afficher sous les yeux de nos visiteurs !

Dimensionner une image

Il est possible de préciser les dimensions que doit prendre l'image, en utilisant les attributs **`width`** et **`height`**, qui donnent respectivement sa *largeur*

et sa *hauteur*. Le plus simple consiste à ne définir qu'un seul de ces attributs, car l'autre sera calculé automatiquement pour que les proportions soient respectées.

Cependant, mieux vaut ne pas utiliser ces attributs et retailler préalablement l'image aux dimensions souhaitées : l'image affichée sera de meilleure qualité et cela n'obligera pas les internautes à télécharger un gros fichier pour afficher seulement une petite image ! Ce redimensionnement du fichier image peut s'effectuer très simplement, en utilisant par exemple un logiciel gratuit comme *Gimp*, *Photofiltre*, *Irfanview*, ...

Objets multimédias

La balise **`<object ...> ... </object>`**, accompagnée de balises **`<param ... />`**, permet d'insérer des objets multimédias tels que du *son*, une *vidéo*, une *animation*...

Animation Flash

Voici un exemple qui montre l'insertion d'une animation Flash au format swf :

```
<object type="application/x-shockwave-flash"
        data="nom-fichier-flash.swf" (1)
        width="150" height="100">
   <param name="movie" value="nom-fichier-flash.swf" /> (2)
   <param name="wmode" value="transparent" />
   <p>Texte alternatif</p> (3)
</object>
```

La balise `<object ...>` possède plusieurs attributs, dont **`data`** (1) pour indiquer le *nom du fichier Shockwave Flash* (`.swf`), **`width`** et **`height`** - à ne pas oublier - pour donner la *largeur* et la *hauteur* de cette animation sur l'écran.

Le *nom du fichier* est répété dans une balise paramètre `<param ... />` (2) Attention à la *barre de fermeture intégrée*, pour chacune de ces balises `<param ... />`.

D'autre part, il ne faut pas oublier d'écrire un paragraphe `<p>` (3) contenant un *texte alternatif*, pour les personnes handicapées ainsi que pour

tous les navigateurs qui ne pourront pas afficher cette animation, dont les moteurs de recherche.

Vidéo

Il existe plusieurs techniques et plusieurs formats pour insérer une vidéo sur une page web. Dans chaque cas, le visiteur doit posséder sur son ordinateur le logiciel capable de lire le format vidéo choisi.

Une méthode courante et assez universelle consiste à insérer la vidéo en tant qu'*objet Flash*, comme sur les sites de partage de vidéos Youtube et Dailymotion.

Il suffira alors que FlashPlayer soit installé, ce qui est généralement le cas : ce complément logiciel gratuit et intégré au navigateur, qui permet l'affichage des animations Flash, est présent sur au moins 95 % des ordinateurs.

Le code XHTML utilise donc la syntaxe précédente et nécessite la présence de deux fichiers :

- Le fichier vidéo converti au format *flv* comme *FLash Vidéo*. Cette conversion peut s'effectuer avec un logiciel gratuit comme *ffmpeg* pour Linux, *ffmpegX* pour Mac ou *Super* pour Windows.
- Un *lecteur vidéo Flash*, en quelque sorte une télévision qui s'affichera sur la page et qui diffusera notre fichier vidéo (voir la figure 2-10). Il s'agit d'un fichier Flash d'extension *swf*, dont il existe moult versions sur Internet, comme par exemple le lecteur *JW FLV Player* qui est disponible à l'adresse :

  ▸ **http://www.jeroenwijering.com.**

FIGURE 2–9 *Extrait du logiciel de conversion vidéo Super (http://www.erightsoft.com) : choix du codage de sortie, ici le format flv comme Flash Video.*

Code à écrire pour insérer un fichier vidéo

```
<object type="application/x-shockwave-flash"
        data="flvplayer.swf" width="300" height="300"> 1
    <param name="movie" value="flvplayer.swf" /> 2
    <param name="flashvars" value="file=mavideo.flv" /> 3
    <p>Texte alternatif</p> 4
</object>
```

Le nom du lecteur vidéo Flash *flvplayer.swf* est mentionné deux fois, sur les lignes 1 et 2 : c'est lui qui représente l'animation Flash à afficher. Le nom du fichier vidéo, ici `mavideo.flv`, est indiqué par un paramètre 3 attribué à l'animation donc au lecteur vidéo.

D'autres paramètres peuvent être ajoutés, comme par exemple celui qui autorise la fonction plein écran :

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

Il est également possible de programmer un démarrage automatique de la vidéo, en modifiant le paramètre qui indique le nom de la vidéo à diffuser :

```
    <param name="flashvars"
           value="autostart=true&file=mavideo.flv" />
```

Pour connaître la liste des paramètres possibles et leur syntaxe, il faut consulter l'aide associée au lecteur vidéo utilisé, en général disponible sur le site de son concepteur.

Enfin, il ne faudra pas oublier le *texte alternatif* 4, pour faciliter la vie des internautes handicapés ; il servira en même temps à renseigner les moteurs de recherche.

> Attention **Dimensions indispensables**
>
> Comme lorsqu'il s'agit d'insérer une animation Flash, la balise `<object ...>` doit contenir les attributs de dimension `width` et `height`, qui donnent respectivement la largeur et la hauteur de la vidéo en pixels.

FIGURE 2–10 *Exemple de vidéo diffusée par le lecteur JW FLV Player, qui propose une barre de commande située sous la vidéo : lecture/pause, plein écran, réglage du son (source : site de l'EID Atlantique http://www.eidatlantique.eu).*

Sauts, lignes et caractères spéciaux

Nous avons déjà vu le *saut de ligne* : balise `<br />` comme *break* en anglais. Il ne faut pas oublier sa barre de fermeture.

Déjà rencontré également, l'*espace insécable* : ` ` comme *non breakable space*, est un espace qu'il ne faudra pas couper. Il est utile pour éviter la séparation de deux mots par un retour à la ligne... sous réserve que le navigateur veuille bien l'interpréter correctement !

La balise `<hr />` comme *horizontal rule* (traduit mot à mot : *règle horizontale*) permet d'afficher un trait de séparation horizontal.

Autres balises de texte

Une liste de quelques autres balises, moins courantes d'utilisation dans une première approche du XHTML, complétera notre énumération.

> Norme **Liste complète des balises XHTML sur le site du W3C**
> La liste détaillée des balises XHTML est disponible sur de nombreux sites, comme celui-ci (en anglais) qui détaille de façon claire et précise les normes du W3C :
> ▸ **http://www.w3schools.com/tags**

`q`	quote = pour une courte citation (mis entre guillemets, sauf pour Internet Explorer 6 et 7)
`address`	paragraphe constitué par une adresse (nouveau paragraphe, mis en italique)
`cite`	citation au cours d'une phrase (en italique)
`dfn`	définition d'un mot, en cours de phrase (en italique)
`var`	nom d'une variable (en italique)
`code`	extrait de code informatique (police Courier)
`samp`	exemple de code informatique (police Courier)
`kbd`	saisie au clavier (police Courier)
`pre`	préformaté (police Courier, espaces et sauts de lignes affichés tels qu'ils sont notés)
`abbr`	abréviation (pas de mise en forme)
`acronym`	acronyme (pas de mise en forme)

Autres balises de listes

`dl`	*definition list* = liste de définitions
`dt`	*definition list term* = une terme de la liste de définitions
`dd`	*definition list definition* = une définition de la liste de définitions (associée à un terme)

Formulaires

`form`	formulaire
`textarea`	zone de texte (pour la saisie) à plusieurs lignes
`input`	entrée (zone de texte à une seule ligne, case à cocher, case d'option, bouton d'action)
`select`	liste de choix
`option`	élément de liste de choix
`optgroup`	regroupement d'éléments de liste de choix
`label`	étiquette pour liste de choix

FIGURE 2–11 *Exemple de formulaire.*

Deux catégories d'éléments : blocs et en ligne

Dans le premier exemple XHTML que nous avons étudié, vous avez remarqué que certaines balises comme <p> provoquaient un retour à la

ligne, alors que d'autres comme laissaient le texte concerné à sa place, à la suite des mots précédents.

Certaines propriétés de mise en forme par les feuilles de style s'appliqueront dans un cas et pas dans l'autre. C'est pourquoi il est important de bien les distinguer.

Le XHTML définit donc pour les balises deux types d'éléments :

- Certains se suivent sur une même ligne de texte : ce sont des *éléments en ligne*.
- Les autres se succèdent verticalement, séparés par un retour à la ligne automatique : ils sont de type *bloc*.

Éléments en ligne

Ils s'écrivent les uns à la suite des autres, dans le texte de la page.

Exemples d'éléments en ligne

```
<strong>...</strong> <!-- mise en relief -->
<em>....</em> <!-- emphase -->
```

Les éléments en ligne se répartissent eux-même en deux catégories :

- les « éléments **remplacés** » dont les dimensions (largeur et hauteur) peuvent être définies : images, zones de saisie d'un formulaire...
- les « éléments **non remplacés** » dont la taille est fonction de leur contenu : éléments , , , ancre <a>, ...

Certaines propriétés liées aux blocs peuvent être appliquées aux éléments en ligne de type *remplacés*.

Les principaux éléments XHTML de type « en ligne » sont les suivants :

- élément (qui sert à délimiter une partie de texte ayant une mise en forme commune) ;
- ancre <a> ;
- image <img> et objet multimédia <object> ;
- texte mis en relief avec (italique) ou encore plus en évidence avec (en gras) ;
- extraits de citation <q> (apparaît entre guillemets) et <cite> (italique) ;

- extrait de programme <code> ou de texte à entrer au clavier (police de type Courier) ;
- exemple <samp> (police Courier), variable <var> (italique) ;
- abréviation <abbr> et acronyme <acronym> ;
- texte inséré <ins> (apparaît souligné) et texte supprimé <del> (apparaît barré).

Ces éléments en ligne peuvent être imbriqués, mais ils ne peuvent pas contenir d'élément de type bloc.

Éléments de type bloc

Ils se placent automatiquement les uns sous les autres. L'utilisation des styles permet de les positionner de façon précise.

Exemples d'éléments de type bloc

```
<h1>....</h1> <!-- titre de niveau 1 -->
<p>...</p> <!-- paragraphe -->
```

Les blocs peuvent contenir d'autres blocs et bien sûr des éléments en ligne, à la notable exception des <p> et <h1>, <h2>,…<h6> qui ne peuvent inclure d'autres blocs.

> **ATTENTION**
>
> Les paragraphes et les titres ne peuvent inclure de blocs.

> **À NOTER** **Marges par défaut**
>
> Tous les blocs (sauf) possèdent des marges intérieures (padding) et extérieures (margin) par défaut, qu'il faut préciser ou mettre à zéro dans la feuille de style.

Voici les principaux éléments HTML de type « bloc » :

- élément , qui sert de boîte « conteneur » et dans lequel seront placés d'autres blocs ;
- titres <h1> à <h6> ;

- paragraphe <p>
- liste et élément de liste <ul>, <ol>, et <li>, liste de définition <dl>, <dd> ;
- citation <blockquote> (apparaît en retrait) ;
- texte préformaté <pre> (affichage fidèle de tous les espaces et retours à la ligne) ;
- adresse <address> (s'affiche en italique, avec espacement vertical).

Hiérarchie des éléments : l'héritage

Les éléments qui composent une page XHTML, c'est-à-dire les balises avec leur contenu, sont *juxtaposés* ou *imbriqués*. Il en découle une *hiérarchie*, qui sera à prendre en compte dans le choix des propriétés de style : *seuls les styles qui ont la faculté d'être hérités se transmettront aux blocs imbriqués.*

Hiérarchie des blocs imbriqués et juxtaposés

Les blocs qui constituent une page forment en quelque sorte une famille, ils sont désignés les uns par rapport aux autres en conséquence :

- Lorsque des blocs sont contenus dans un autre bloc, ils sont les *enfants* de ce dernier.
- Entre eux, ces blocs imbriqués sont des *frères*.
- Le bloc conteneur est leur *père*.
- Le *premier fils* d'un bloc conteneur est le premier des blocs imbriqués qu'il contient.

Voici un exemple qui nous transporte dans l'atmosphère calme et sereine de la campagne, et qui produit l'affichage montré par la figure 2.12 :

Exemple de code contenant des blocs imbriqués et juxtaposés

```
<body id="ferme"> (1)
  <div id="basse-cour">À l'ombre du noisetier... (2)
    <p id="poule">Cot ! Cot ! Cot!</p>
    <p id="canard">Coin ! Coin ! Coin !</p>
```

```
    <p id="chien">Ouah ! Ouah !
        <em id="puce">une puce pique le chien</em>
    </p>
  </div>
  <div id="enclos">Dans une prairie verte... (3)
    <p id="vache">Meuh ! Meuh !</p>
    <p id="cochon">Groin ! Groin !</p>
  </div>
</body>
```

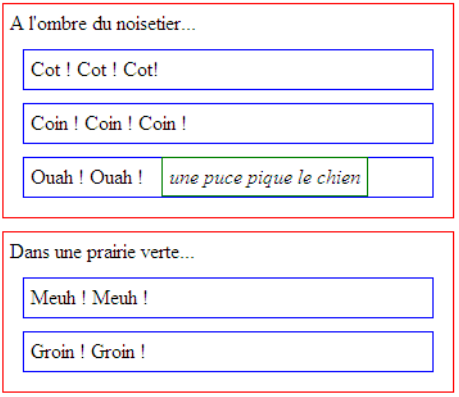


FIGURE 2–12 *Exemple de blocs imbriqués et juxtaposés.*

En français, l'histoire se raconte ainsi :

- La ferme contient la basse-cour et l'enclos.
- La basse-cour contient la poule, le canard et le chien.
- Le chien « contient » la puce.
- L'enclos contient la vache et le cochon.

Les éléments XHTML de cet exemple sont imbriqués de la même façon, en suivant la logique de cette histoire.

Figure 2–13 *À la ferme...*

Termes hiérarchiques utilisés en XHTML/CSS

Le bloc `<body id="ferme">` ❶ est l'*ancêtre* commun à tous les autres blocs.

Il est le *père* de `<div id="basse-cour">` ❷ et de `<div id="enclos">` ❸ qui sont ses deux descendants directs, appelés ses *fils* ou ses *enfants*. Ces deux blocs sont donc *frères*. Le *premier fils* de l'élément `<body id="ferme">` ❶ est `<div id="basse-cour">` ❷.

Le bloc `<div id="basse-cour">` ❷ a trois fils (paragraphes *poule*, *canard*, *chien*, qui sont frères entre eux) et un petit-fils (le texte *puce*, qui est le fils du paragraphe *chien*).

Le bloc `<div id="enclos">` ❸ a deux fils (paragraphes *vache* et *cochon*).

Héritage des propriétés de style

Certaines des propriétés définies dans les feuilles de style sont *héritées*, c'est-à-dire qu'elles sont transmises aux éléments imbriqués ; d'autres ne peuvent pas l'être.

Si une propriété qui peut être héritée est appliquée à un élément, elle s'appliquera en même temps à tous les éléments que celui-ci contient.

Si en revanche elle n'est pas héritée, elle ne se retrouvera pas sur les éléments imbriqués.

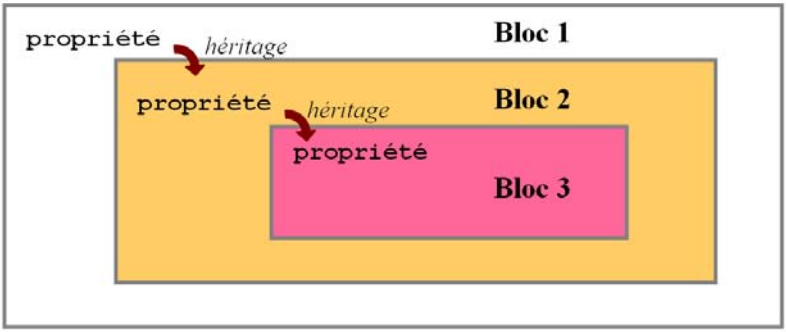


FIGURE 2–14 *Principe de l'héritage d'une propriété, illustré ici pour des blocs imbriqués.*

Reprenons l'exemple de la ferme pour illustrer le comportement de deux propriétés dans les blocs imbriqués.

La propriété `font-family` (police d'écriture) est *héritée.* Supposons, par exemple, que la propriété `font-family: Arial;` soit appliquée aux blocs `<div>` ❷ et ❸ seulement (basse-cour et enclos). Grâce à l'héritage de cette propriété et comme le montre la figure 2-15, tous leurs descendants directs ou indirects seront également écrits en Arial (ces descendants sont les blocs `<p>` qui sont leurs enfants et l'élément `<em>` qui est un petit-enfant).

La propriété `border` (type de bordure) ne *peut pas* être héritée. La figure 2-16 donne le résultat affiché lorsqu'une bordure est attribuée aux seuls blocs `<div>` ❷ et ❸ (basse-cour et enclos). Cette propriété n'étant pas héritée, leurs descendants ne seront pas encadrés, car ils conserveront la valeur par défaut « aucune bordure ».

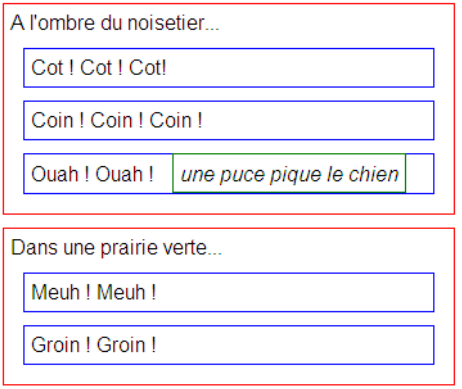


Figure 2–15 *La propriété font-family est héritée.*

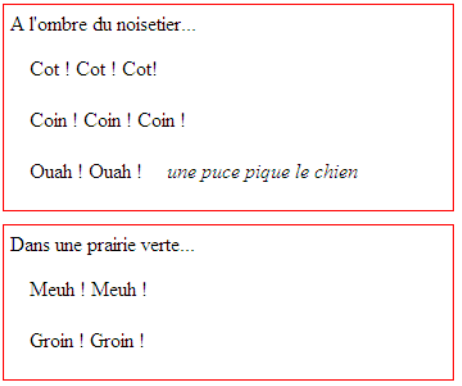


Figure 2–16 *La propriété border (type de bordure) n'est pas héritée.*

Lorsque nous utiliserons des feuilles de style CSS pour mettre en forme les différents éléments de nos pages, nous nous inquiéterons des facultés d'héritage de chacune des propriétés utilisées :

- lorsqu'il s'agit d'une propriété héritée, il sera inutile de la réécrire pour chaque bloc imbriqué, il suffira de l'appliquer une fois au bloc conteneur ;

- par contre, si la propriété de style concernée n'est pas héritée, elle devra être répétée pour chacun des blocs imbriqués auxquels il faudra affecter ce style.

Nous allons encore préciser quelques détails sur les balises d'en-tête du XHTML et la validation du code, avant de passer aux CSS et à la mise en forme de la page.

Compléments sur les balises d'en-tête

Balise DOCTYPE

Rappelons que cette balise, située tout au début de notre fichier, indique au navigateur la version de HTML ou XHTML utilisée.

Cette déclaration est importante pour une bonne interprétation du code. Voici les DOCTYPE les plus courants :

HTML 4.01

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
```

XHTML 1.0 Transitional

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

XHTML 1.0 strict (sans utilisation de balises obsolètes)

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

> NORMES **Strict ou Transitional ?**
>
> Le XHTML *Strict* respecte exactement les normes XHTML.
> En XHTML « de transition », nommé *Transitional*, il est possible d'utiliser des balises ou attributs HTML « obsolètes » parce qu'abandonnés en XHTML.

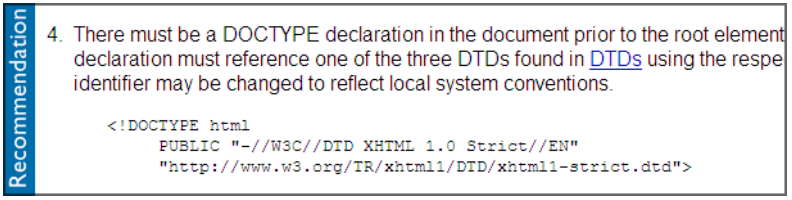


FIGURE 2–17 *Le site http://www.w3.org fournit les DOCTYPE officiels.*

Balise meta et codage en utf-8

Rappelons que la première ligne qui suit `<html ...>` doit être une balise `<meta ... />` qui indique le *type de codage* utilisé pour enregistrer le fichier texte.

Si ce codage est correctement déclaré, il sera possible de taper directement lettres accentuées, c cédille ç et symbole euro €, sans avoir à utiliser des entités HTML comme `&agrave;` pour « à » ou `&ccedil;` pour « ç » par exemple.

Voici la balise **`meta`** qui indique que la page a été enregistrée selon la norme de codage **`utf-8`** :

```
<meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
```

MÉTHODE

Éditer son fichier XHTML avec le bon type de codage

Le fichier XHTML doit être édité et enregistré avec le même type de codage que celui déclaré dans cette ligne (en général, le choix peut se faire dans la boîte de dialogue d'enregistrement de l'éditeur, souvent à l'aide d'une liste déroulante *Codage*).

Les guillemets n'entourent pas seulement le type de codage (ici `utf-8`), mais l'expression entière qui suit le mot `content` (ici, `"text/html; charset=utf-8"`)

NORMES **Principaux types d'encodage**

- utf-8 (`charset=utf-8`) : codage universel Unicode (sur un ou deux octets), à privilégier car c'est la solution d'avenir.
- iso-8859-1 ou Latin-1 (`charset=iso-8859-1`) : codage occidental classique, souvent utilisé.
- iso-8859-15 ou Latin-9 (`charset=iso-8859-15`) : codage occidental iso-8859-1 plus quelques caractères, dont €, Œ, œ, et Ÿ.
- Windows-1252 ou ANSI (`charset=windows-1252`) : codage provenant du iso-8859-1, comprenant également €, Œ, œ, et Ÿ.

Autres balises d'en-tête

La balise **`<base ... />`** permet de définir le dossier de référence pour les liens hypertextes, les images ou autres fichiers fournis dans le code.

EXEMPLE **Utilisation de la balise `<base ...>`**

Si l'en-tête contient la balise :

```
<base href="http://monsite.com/fichiers/" />
```

alors la balise

```
<img src="images/logo.png" alt="logo">
```

(placée dans le corps de la page) affichera l'image `logo.png` située dans le dossier :

`http://monsite.com/fichiers/images/`.

Comme nous l'avons vu, la balise **`<title>`**`Titre de la page`**`</title>`** indique le texte qui s'affichera dans la barre de titre du navigateur.

ATTENTION **Ne pas confondre titre et nom du fichier**

Le contenu de cette balise `title` n'a rien à voir avec le nom du fichier, ce dernier étant défini lors de l'enregistrement de la page. Par contre, ce sera le nom proposé par défaut à l'internaute qui voudra mémoriser l'adresse de cette page parmi ses *Marque-pages* ou *Favoris*.

D'autres balises **`<meta ... />`** permettent de fournir des informations complémentaires : mots-clés, résumé de la page, nom de son auteur...

Validation du code XHTML

Pour vérifier si le code d'une page est valide, suivant la norme indiquée dans la balise `<!DOCTYPE ...>`, il suffit de mettre cette page en ligne et d'indiquer son url au validateur du W3C, à l'adresse : http://validator.w3.org.

> À NOTER **La validation n'est qu'une indication**
>
> Le validateur vérifie uniquement la *syntaxe* de la page (en quelque sorte, sa « grammaire »), mais pas la logique dans l'emploi des balises, à savoir si le sens associé à telle ou telle balise correspond bien au texte qu'elle encadre.

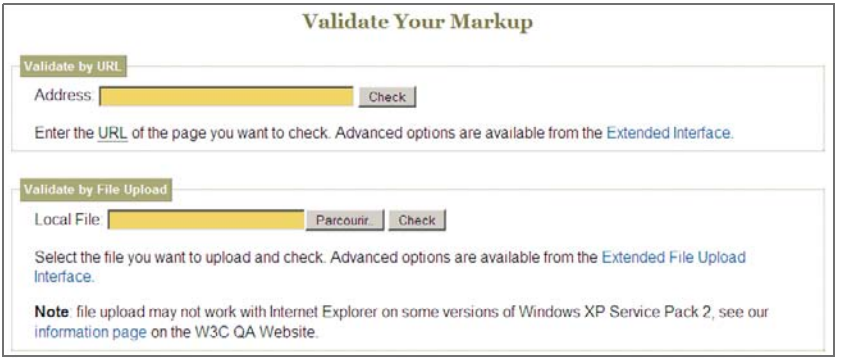


Figure 2–18 *À l'adresse http://validator.w3.org, le W3C propose la validation du code XHTML ou HTML.*

Nous voilà familiarisés avec le XHTML : à présent, la construction d'une page élémentaire avec quelques balises simples est tout à fait dans nos cordes. De plus, nous sommes sensibilisés à des notions telles que l'imbrication des blocs et l'héritage des propriétés de style, qui nous seront utiles pour mettre en forme notre page.

Il nous reste à apprendre de quelle façon effectuer cette mise en forme. Il s'agira de savoir, dans un premier temps, où et comment écrire des règles de style CSS, puis de quelle façon celles-ci pourront être appliquées aux différents éléments de la page. Le chapitre qui suit va nous permettre de comprendre ces notions.

chapitre

3

Écriture des feuilles de style

Comment écrire une règle de style ?
Où et dans quel ordre écrire ces règles ?
De quelle façon attribuer une propriété
à un élément donné de la page web ?

Sommaire

Nous voilà prêts à aborder les feuilles de style proprement dites. Organisons notre démarche, en commençant par nous poser les questions de base : comment, où et dans quel ordre écrire ces définitions qui, une fois réunies, formeront une feuille de style ?

Pour chaque règle de mise en forme, il faudra d'abord définir les éléments de la page web, puis les propriétés à leur attribuer.

Définition d'une règle de style

Voici comment sélectionner un élément de la page et lui attribuer une propriété de mise en forme.

Principe

Une règle de style comprend :

- un **sélecteur** : il s'agit des balises concernées par cette règle ;
- un **bloc de déclarations** : il indique les propriétés à attribuer à ces balises.

Chaque déclaration est du type : `propriété : valeur;`

Exemple de règle de style

La règle de la figure 3-1 indique que les titres de niveau 3 (encadrés par `<h3>...</h3>`) s'afficheront en *italique* et en *Arial* (ou dans une police générique *sans-serif* si la police Arial est absente).

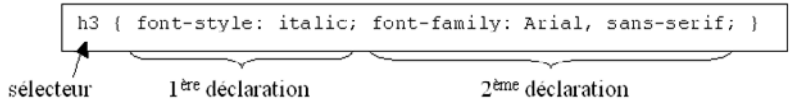


FIGURE 3–1 *Exemple de règle de style*

Cette règle comprend :

- le sélecteur (`h3`) ;
- deux déclarations, donc deux propriétés à attribuer aux titres de niveau 3 de la page.

> À NOTER **Écriture d'une règle de style**
> - Chaque déclaration se termine par un point-virgule.
> - Une règle peut s'écrire sur plusieurs lignes :

```
h3 {
   font-style: italic;
   font-family: Arial, sans-serif;
}
```

Commentaires

Il est utile de commenter abondamment les feuilles de style, pour s'y retrouver plus tard lorsqu'il s'agira d'apporter des modifications. Il suffit de placer les commentaires entre les signes `/*` et `*/` :

```
/* Voici un commentaire */
/* Et en voilà un autre,
   mais sur plusieurs lignes */
```

Emplacement des styles

Les règles de style peuvent se trouver :

- dans le code HTML, comme attributs de balises : ce sont des styles en ligne (utilisation déconseillée - voir plus loin) ;
- dans l'en-tête de la page web : feuille de style interne ;
- ou dans un fichier distinct : feuille de style externe, à appeler dans l'en-tête de la page web.

Feuille de style interne

Lorsque les règles de styles sont regroupées dans l'en-tête de la page web, elles constituent une feuille de style interne.

Les styles sont écrits entre les balises <head> et </head>, à l'intérieur d'une balise <style> :

```
<head>
...
<style type="text/css">
<!--
...règles de styles ici...
-->
</style>
...
</head>
```

À NOTER Déclaration d'une feuille de style interne

- Les styles ainsi déclarés sont de type texte, d'où l'attribut `type="text/css"` dans la balise `<style>`.
- Les règles de styles sont placées à l'intérieur de balises de commentaires HTML `<!--` et `-->` de façon à être ignorées par les navigateurs qui ne connaissent pas les CSS.

Feuille de style externe

Lorsque des règles de styles sont applicables à plusieurs pages web, il est intéressant de les écrire dans un fichier à part. Cette feuille de style externe est appelée par chacune des pages concernées. Elle garantit l'unité graphique du site et facilite les modifications.

Une feuille de style externe est un fichier d'extension `.css` :

- C'est un fichier texte qui contient l'ensemble des règles définies.
- Il ne contient pas les balises `<style> ... </style>`.
- Il n'y a pas non plus les symboles de commentaires `<!-- ... -->`.

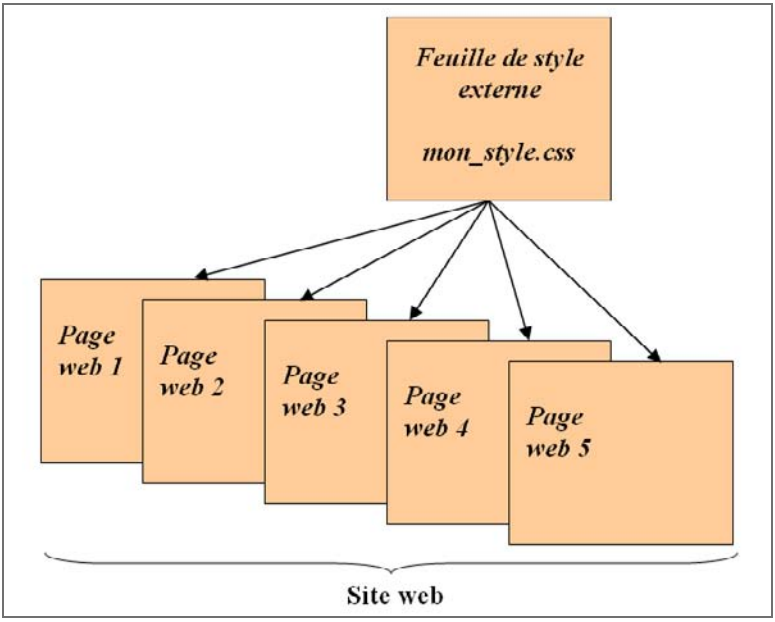


Figure 3–2 *Une feuille de style externe garantit une mise en forme homogène pour l'ensemble du site web.*

Pour que cette feuille de style soit prise en compte dans une page web, il suffit de l'appeler dans l'en-tête, en utilisant une des deux méthodes suivantes (dans l'exemple qui suit, la feuille de style s'appelle `mon_style.css` et se trouve dans le même dossier que la page web) :

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

ou

```
<head>
  ...
  <style type="text/css">
    @import url(mes_styles.css);
  </style>
  ...
</head>
```

> À NOTER **Méthode @import**
>
> Elle peut être utilisée dans une feuille de style externe, ce qui permet d'importer une première feuille de style à l'intérieur d'une deuxième. Elle n'est pas reconnue, donc ignorée par l'ancienne version Netscape 4, qui n'est pas aux normes CSS et interprète mal les styles : mieux vaut lui laisser ignorer ces styles et afficher le texte brut, qui au moins restera lisible.

Styles en ligne

Dans le corps de la page HTML (entre `<body>` et `</body>`), il est possible de préciser des styles qui prévaudront sur ceux précédemment déclarés dans la feuille de style.

Exemple d'un titre de niveau 2 qui doit être centré et écrit en rouge :

```
<h2 style= "text-align: center; color: red;">
  ...Titre...
</h2>
```

Cette méthode est à éviter autant que possible, car elle revient à mélanger à nouveau le contenu et la mise en forme, comme en « ancien » HTML :

- Cela enlève la clarté et l'homogénéité apportées par les feuilles de style.
- La maintenance des pages redevient plus délicate.

Il est préférable d'identifier la balise concernée à l'aide d'un nom, qui permettra de lui attribuer une règle de style spécifique.

Sélecteurs de style

Dans une règle de style, le choix du sélecteur est extrêmement important : il indique les balises concernées par la mise en forme qui suit.

Si des balises de même type doivent avoir différentes mises en forme, elles seront identifiées par des noms, lesquels seront repris dans les sélecteurs.

Comme au théâtre

Figure 3–3 *L'Opéra Garnier*

Avant d'aller plus loin, transportons-nous au théâtre pour un petit instant. Tiens ! On y monte une pièce... Le metteur en scène distribue les rôles, les costumes et il place les acteurs qui joueront la pièce.

De la même façon, le concepteur web va placer les balises XHTML de la page. Le costume qu'il leur attribuera sera fait de couleurs, de polices de caractères, de bordures...

FIGURE 3–4 *Le metteur en scène monte une pièce, comme le concepteur web met en place son site : il aura un certain nombre d'éléments à mettre en forme.*

Sélecteur simple

Au théâtre, le metteur en scène dit : « Vous les hommes, vous serez habillés en bleu. Et vous, les femmes, en rouge. »

Les acteurs représentant nos balises de la page web, les hommes et les femmes sont des types de balises, par exemple, `<div>` et `<p>`, et la règle qui vient d'être énoncée :

```
homme { couleur: bleu; }
femme { couleur: rouge; }
```

pourrait devenir, dans notre feuille de style :

```
div { color: blue; }
p { color: red; }
```

La première règle s'applique à toutes les balises `<div>` de la page web, la deuxième à toutes les balises `<p>`.

Figure 3–5 *« Les hommes seront habillés en bleu, les femmes en rouge. » Hommes et femmes représentent ici des « balises » distinctes, auxquelles sont attribuées des valeurs différentes pour la propriété « couleur ».*

Classe

Une catégorie de balises

Au théâtre, le metteur en scène demande aux femmes qui ont un chapeau de venir au centre de la scène. Là, il vient de définir une sous-catégorie parmi celle des femmes. La consigne ne s'adresse plus à toutes les femmes, mais seulement aux femmes qui ont un chapeau.

Dans nos pages web, supposons que parmi les paragraphes (délimités par les balises . . .), il faille centrer uniquement ceux qui constituent un chapeau dans le texte (un chapeau est une phrase d'introduction qui surplombe plusieurs colonnes).

Il suffit de donner un *nom de classe* à ces paragraphes, comme attribut de la balise dans le code XHTML :

```
<p class="chapeau">...</p>
```

et d'écrire la règle suivante dans la feuille de style :

```
p.chapeau { text-align: center; }
```

Les balises qui ne sont pas de la classe « chapeau » ne seront pas concernées par cette règle, ni les balises autres que <p>, même si elles appartiennent à la classe « chapeau ». En particulier, la règle précédente ne s'applique pas à l'élément `<div class="chapeau"><p> ... </p></div>`.

FIGURE 3–6 *« Les femmes qui ont un chapeau, venez au centre de la scène ! » Parmi les femmes est désignée la « classe » de celles qui ont un chapeau.*

Une même classe pour plusieurs types de balises

Notre metteur en scène prie alors toutes les personnes qui ont un chapeau de se pencher pour saluer le public. Une catégorie est définie, là encore, mais elle n'est pas associée à un type d'acteur : les hommes comme les femmes sont concernés.

De la même façon, il est possible d'appliquer des propriétés à plusieurs éléments de la page web, quelle que soit la balise qui les entoure. Il suffit de leur attribuer une classe commune, par exemple `<p class="menu">`, `<h1 class="menu">`, `<a class="menu">` et d'écrire une règle du type :

```
.menu { font-style: italic; }
```

Le type de balise n'est pas précisé, seule la classe est indiquée. Cette règle fera donc apparaître en italique le contenu de n'importe quelle balise de classe « menu ».

À noter **Plusieurs classes pour une même balise**

Une balise peut être associée à plusieurs classes. Par exemple, l'élément `<p class="intro menu">...</p>` sera mis en forme par les règles du type `p.intro { ... }` et `p.menu { ... }`. Notez cependant qu'il n'est pas possible d'écrire comme sélecteur `p.intro.menu`, donc de préciser l'appartenance simultanée d'une balise à deux classes.

Identifiant

Retour au théâtre : après les directives générales, le metteur en scène donne des instructions plus précises, à chaque acteur en particulier. « Toi, Thomas, passe à droite ! Toi, Marion, assieds-toi dans le fauteuil ! » Chacune de ces consignes ne concerne qu'une personne. Il n'y a qu'un seul Thomas et une seule Marion dans la troupe des Joyeux Cabotins, et au début de la saison, lors des présentations, chacun a donné son prénom.

C'est ainsi qu'en XHTML, une alternative aux classes est utilisée pour repérer un élément *unique* dans une page web. Il s'agit de donner à la balise un *identifiant* :

```
<div id="Thomas">....</div>
<p id="Marion">...</p>
```

ce qui permet de préciser cet identifiant dans la feuille de style, à l'aide du caractère dièse **#** :

```
div#Thomas { text-align: right; }
p#Marion { vertical-align: -50%; }
```

Comme vous le voyez, le sélecteur précise l'identifiant comme une classe, simplement avec un dièse (#) à la place du point. Évidemment, les identifiants seront rarement des prénoms, mais plutôt des repères liés à la fonction de l'élément dans la page. Par exemple, à ce paragraphe :

```
<p id="auteur">...</p>.
```

sera attribué une propriété de style comme :

```
p#auteur { color: gray; }
```

À NOTER **Utilisation des identifiants**

- Un même identifiant ne peut pas être utilisé pour deux éléments distincts de la même page, car en XHTML il remplace l'attribut `name="..."` pour repérer un élément unique (l'attribut `name` doit néanmoins être conservé pour les balises `map` et les éléments de formulaire).
- Dans une règle, il est possible d'utiliser une classe et un identifiant en même temps : `p.menu#auteur {...}`.

FIGURE 3–7 *« Marion, assieds-toi dans le fauteuil ! » Le nom de cette actrice, c'est son « identifiant ».*

Identifiant sans nom de balise

De la même manière que pour les classes, le nom de la balise peut être omis dans le sélecteur :

```
#auteur { color: gray; }
```

Cette règle s'applique à *la* balise d'identifiant « auteur » `<... id="auteur">` (il ne peut y en avoir qu'une seule dans la page).

Précision **Avec ou sans nom de balise ?**

Les sélecteurs `p#auteur` et `#auteur` sont équivalents, puisqu'il ne doit y avoir qu'une seule balise d'identifiant « auteur » dans la page.
En revanche, en ce qui concerne les classes, les sélecteurs `p.chapeau` et `.chapeau` n'ont pas le même sens : la balise `<div class="chapeau">` est concernée par le deuxième sélecteur, mais pas par le premier.

Différence entre classe et identifiant

Tableau 3–1 **Comparaison entre classe et identifiant**

	Classe	Identifiant
Balise	`<p class="toto">`	`<p id="toto">`
Règle de style	`p.toto { ... }` `.toto { ... }`	`p#toto { ... }` `#toto { ... }`
Éléments concernés	plusieurs balises, identiques ou différentes	une seule balise dans toute la page

Pseudo-classes

Notre metteur en scène précise et coordonne les actions des acteurs : « Toi, lorsque le maître de maison arrive, tu te retournes. Toi, après le passage du docteur, tu es guéri ! »

En CSS, il existe des *pseudo-classes* qui, accolées à une balise, apportent des précisions aux sélecteurs. Cette méthode permet d'écrire des propriétés à utiliser uniquement dans certains cas de figure. Les principales pseudo-classes sont les suivantes :

```
:link, :visited, :hover, :active,
:first-child, :focus, :lang
```

Leur utilisation sera détaillée un tout petit peu plus loin. La pseudo-classe la plus utilisée est `:hover` ; elle indique que la règle de style n'est à appliquer qu'au passage de la souris.

Par exemple, pour mettre en rouge le texte d'une balise <a> au moment de son survol par le curseur de la souris, il suffit d'écrire dans la feuille de style :

```
a:hover { color: red; }
```

La règle suivante, elle, met le texte en rouge au passage de la souris, mais uniquement sur les balises `<a class="menu" ...>`, les autres balises <a> n'étant pas concernées :

```
a.menu:hover { color: red; }
```

FIGURE 3–8 *« Après le passage du docteur, tu es guéri ! » L'action (rapide !) du docteur a eu un effet sur l'acteur, comme un clic de souris sur un lien le transforme en lien « visité ».*

Pseudo-classes pour les liens hypertexte

- **`:link`** = lien hypertexte qui n'a pas été visité ;
- **`:visited`** = lien visité (et encore présent dans l'historique du navigateur) ;
- **`:hover`** = élément survolé par la souris ;
- **`:active`** = élément activé (la souris pointant ce lien, son bouton est enfoncé).

Si plusieurs de ces quatre pseudo-classes sont utilisées, il faut respecter un ordre précis pour qu'elles soient bien prises en compte : **`:link`**, puis **`:visited`**, puis **`:hover`**, puis **`:active`**. Il existe un moyen mnémotechnique pour mémoriser leur ordre à partir de leurs initiales : LoVe HAte.

Autres pseudo-classes

Ces trois pseudo-classes sont assez peu utilisées, en particulier parce qu'elles ne sont pas comprises par tous les navigateurs :

- `:first-child` = premier enfant d'une balise quelconque (qui suit immédiatement cette balise) ;
- `:focus` = qui possède le focus (par exemple pour une zone de saisie d'un formulaire, lorsque le curseur clignote dedans) ;
- `:lang` = balise qui possède un attribut « `lang` ».

> Attention **Prise en compte des pseudo-classes**
>
> - Les pseudo-classes `:focus` et `:lang` ne sont pas prises en compte par Internet Explorer, version 6 ou 7.
> - La pseudo-classe `:first-child` n'est pas prise en compte par Internet Explorer 6, mais elle est reconnue à partir de la version 7 ainsi que par Firefox.

Pseudo-éléments

Ressemblant aux pseudo-classes, les pseudo-éléments apportent d'autres types de précision :

- `:first-letter` = première lettre du bloc ;
- `:first-line` = première ligne du bloc ;
- `:before` = avant la balise spécifiée ;
- `:after` = après la balise spécifiée.

L'exemple suivant agrandit la taille de la première lettre pour chaque paragraphe <p> :

```
p:first-letter { font-size: 150%; }
```

> Utilisation **Pseudo-éléments `:before` et `:after`**
>
> Ces deux pseudo-éléments servent à insérer du texte ou une image avant ou après une balise donnée.
> Exemple : `p.note:before { content: "Note : "; }`
> Ils ne sont pas reconnus par les versions 6 et 7 d'Internet Explorer.

Règle associée à plusieurs sélecteurs

Si le metteur en scène dit : « Tous les hommes avec un chapeau et toutes les femmes, venez au centre de la scène ! », il donne une seule consigne qui s'adresse à plusieurs catégories d'acteurs. Sont concernés ici tous ceux des hommes qui ont un chapeau et toutes les femmes sans exception.

En CSS, cela donnerait quelque chose comme :

```
div.chapeau, p { text-align: center; }
```

Cette règle s'applique à toutes les balises <p> et aux balises .

Voici un autre exemple de règle qui s'applique au contenu des balises <h1>, <h2> et <h3 class="sommaire">:

```
h1, h2, h3.sommaire { text-align: center; }
```

FIGURE 3–9 *En donnant une consigne unique pour les hommes avec chapeau et pour les femmes, le metteur en scène attribue, en une seule fois, une « propriété » à une balise avec la classe « chapeau », ainsi qu'à une autre balise, sans précision de classe.*

Regroupement de propriétés à l'aide « raccourcis »

La règle suivante définit, pour les titres de niveau 1, leur type de bordure (épaisseur, style et couleur de l'encadrement) :

```
h1 {
  border-width: 2px;
  border-style: solid;
  border-color: blue;
}
```

Ces trois propriétés peuvent être remplacées par une seule propriété **`border`** qui prend en compte les trois valeurs associées :

```
h1 { border: 2px solid blue; }
```

Hiérarchie des sélecteurs

Pour appliquer un style à un élément à condition qu'il soit inclus dans un autre, il suffit d'écrire un sélecteur avec les deux noms de balises (ou de classe, d'identifiant...) séparés par un espace, comme le montrent les exemples ci-après.

Pour justifier seulement le texte des paragraphes <p> qui sont inclus dans un bloc , il faut écrire la règle suivante :

```
div p { text-align: justify; }
```

Pour mettre en gris uniquement les liens contenus dans l'élément d'identifiant « sommaire », la règle à utiliser est :

```
#sommaire a { color: gray; }
```

> À NOTER **Plusieurs niveaux d'imbrication**
>
> Ces règles s'appliquent aussi aux balises qui sont séparées par d'autres niveaux d'imbrication. Il suffit que la deuxième balise soit un « descendant », direct ou éloigné, de la première.
> La règle du dernier exemple s'appliquera dans le cas suivant :
> `<div #sommaire>...<p>...<a>...</a>...</p>...</div>`

Hiérarchie précise des sélecteurs

Les caractères > et + expriment une hiérarchie plus précise entre les balises : imbrication directe pour >, juxtaposition des balises avec +.

Imbrication directe

```
div > h1 { font-style: italic; }
```

Cette règle s'applique aux balises <h1> qui sont dans le premier niveau d'imbrication à l'intérieur d'une balise (enfant direct), mais elle n'influera pas sur celles imbriquées plus « profondément » à partir du (descendants au-delà de la première génération).

Juxtaposition

```
h1 + h2 { margin-top: 10px; }
```

Cette règle s'applique à chaque balise <h2> qui suit une balise de fermeture </h1> (c'est le frère suivant de <h1>). Entre ces deux balises peut se trouver du texte, mais pas une autre balise.

Sélecteur d'attribut [...]

C'est une méthode d'avenir, qui sélectionne les balises ayant un attribut donné. Elle est prise en compte par Firefox 1.5, 2 et 3, par Internet Explorer 7 et 8, mais pas par Internet Explorer 6.

Voici un exemple de règle qui applique une couleur de fond jaune aux boutons de formulaire. Elle concerne les balises <input> qui ont un attribut type="button", donc les balises <input type="button"...> :

```
input[type="button"] { background-color: yellow; }
```

Voici un autre exemple, qui met en vert le texte des liens pour lesquels un raccourci d'accessibilité est défini par l'attribut accesskey (<a accesskey="..."...>), quelle que soit ici la valeur de cet attribut :

```
a[accesskey] { color: green; }
```

Sélecteur universel *

Retournons une dernière fois au théâtre, si vous le voulez bien. Le metteur en scène, très pointilleux, explique : « À la fin de la pièce, tous les acteurs viennent au centre pour saluer. » (Et les acteurs haussent les épaules, comme pour dire : « Mais il nous prend pour des débutants, ce rigolo ? »). Le terme « tous les acteurs » englobe tous les éléments de notre joyeuse troupe, sans exception.

Figure 3–10 *« À la fin de la pièce, tous les acteurs viennent au centre pour saluer. » Toutes les balises sont concernées par la propriété.*

Dans une règle de style, c'est l'étoile (`*`) qui est utilisée comme sélecteur universel. Elle signifie : « n'importe quelle balise ». La consigne précédente devient alors :

```
* { text-align: center; }
```

Voici en particulier une propriété qui est souvent utilisée pour toutes les balises de la page :

```
* { margin: 0; }
```

À NOTER **Règle pour toutes les balises d'un bloc**

Si une règle concerne toutes les balises qui sont incluses dans un bloc donné, il faut écrire par exemple :

```
div * { color: blue; }
```

Dans ce cas précis, toutes les balises qui sont incluses dans un bloc `<div>` quelconque verront leur texte écrit en bleu.

Ordre de priorité des styles

Si une règle de style vient contredire une règle précédente, c'est *en général* **le dernier style défini qui s'applique**.

Règle de style prioritaire

Pour qu'un style ne soit pas modifié par un autre, il faut écrire `!important` avant le point-virgule qui termine la propriété. Exemple :

```
body { background-color: white !important; }
```

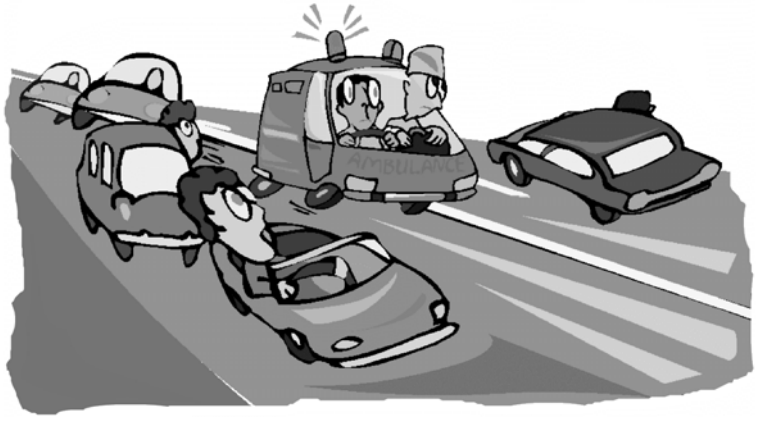


FIGURE 3–11 *Comme les véhicules de secours sur la route, les propriétés marquées par* `!important` *auront la priorité.*

Précision **Comportement d'Internet Explorer 6 avec !important**

Internet Explorer 6 ne prend en compte cette règle contenant `!important` qu'après avoir lu son accolade de fin. Ce défaut peut être utilisé pour indiquer deux règles en une seule, destinées à différents navigateurs.

Dans l'exemple qui suit, une image de fond PNG est affichée, sauf pour IE 6 (Internet Explorer 6) qui ne comprend pas la transparence de ces images et qui affichera la place une image GIF. Ceci est possible grâce à l'utilisation d'une règle de style qui contient deux propriétés :

- une propriété pour les navigateurs aux normes CSS, avec `!important` ;
- puis une autre, sans `!important`, pour IE 6

```
div.ma_classe {
   background-image:url(image1.png) !important;
   background-image:url(image2.gif);
}
```

Pour les navigateurs aux normes CSS, seule la première image de fond sera utilisée, la deuxième propriété étant ignorée. Par contre, Internet Explorer 6 redéfinira le style et utilisera la deuxième image, sans s'occuper du caractère prioritaire donné à la première des deux propriétés `background-image`.

Degré de priorité d'une règle de style

Je propose aux débutants en CSS d'ignorer ce paragraphe dans un premier temps et d'y revenir plus tard.

La règle exacte de priorité, pour les styles en cascade, est la suivante :

Si deux règles de style sont contradictoires, la deuxième remplace la première, sauf si cette première règle a un degré de priorité (c'est-à-dire de spécificité) supérieur à la deuxième.

Le degré de priorité d'une règle de style est fonction d'un nombre de quatre chiffres $x_4\,x_3\,x_2\,x_1$, calculé à partir du sélecteur de cette règle :

- chiffre des *milliers* (x_4) :
  1 si style **prioritaire** (style en ligne, ou `!important`), 0 sinon ;
- chiffre des *centaines* (x_3) :
  nombre d'identifiants (#xxx) dans le sélecteur ;

- chiffre des *dizaines* (x_2) :
  nombre de classes (.xxx) qui interviennent dans le sélecteur ;
- chiffre des *unités* (x_1) :
  nombre d'éléments séparés par des espaces dans le sélecteur.

Le tableau suivant permet de comprendre ce calcul de spécificités, donc de priorités, classées ici par ordre croissant. Ce document provient de la page http://www.openweb.eu.org/articles/cascade_css/, à consulter pour plus d'informations.

TABLEAU 3–2 Exemples de calculs de priorité

Style	Style local ou !important	Nombre d'identifiants	Nombre de classes	Nombre d'éléments	Priorité
`* {...}`	0	0	0	0	0000
`p {...}`	0	0	0	1	0001
`div p {...}`	0	0	0	2	0002
`.class {...}`	0	0	1	0	0010
`p.class {...}`	0	0	1	1	0011
`div p.class {...}`	0	0	1	2	0012
`#id {...}`	0	1	0	0	0100
`p#id {...}`	0	1	0	1	0101
`div p#id {...}`	0	1	0	2	0102
`.class #id {..}`	0	1	1	0	0110
`.class p#id {...}`	0	1	1	1	0111
`div.class p#id {...}`	0	1	1	2	0112
`<p style="...">`	1	0	0	0	1000
`..{..!important;}`	1	0	0	0	1000

Application

Voici un exemple où une propriété est redéfinie à l'aide d'une deuxième règle qui a une priorité plus faible que la première :

```
div p { color: blue; }
p { color: green; }
```

Dans ce cas, les paragraphes <p> seront écrits en vert, sauf ceux inclus dans un bloc , qui resteront en bleu.

Cet exemple n'est toutefois pas un modèle d'écriture à suivre : la logique et la facilité de compréhension du code voudraient que l'ordre de ces deux règles soit inversé, pour aller du plus général vers le plus spécifique.

> À noter **Priorités et combinateurs d'éléments**
>
> Les combinateurs d'éléments tels que **>** (enfants directs) ou **+** (éléments adjacents) n'ont pas d'influence sur les priorités.

Valeurs, tailles et couleurs

Avant d'aborder le détail des propriétés, il est important de définir les codes et unités à adopter pour les valeurs qui leur seront attribuées.

Héritage de propriété

Toutes les propriétés peuvent prendre la valeur `inherit` : cela crée un héritage pour des propriétés qui normalement ne sont pas héritées.

Unités de taille

Les unités de taille sont souvent utilisées pour les propriétés de polices de caractères : bordures, dimensions, marges extérieures et intérieures...

Elles peuvent être fixes, définies par une longueur, ou relatives à l'affichage utilisé.

> À noter **Valeurs décimales**
>
> Si rien n'oblige les valeurs de taille à être entières, il faut cependant penser à utiliser le point et non la virgule comme séparateur décimal.

Unités de taille fixe

Les valeurs utilisables pour des tailles fixes sont :

- **pt** (1 point = 0,35 mm) ;

- **pc** (1 pica = 12 pt = 4,22 mm) ;
- **cm**, **mm**, **in** (1 inch ou 1 pouce = 2,54 cm).

Il vaut mieux éviter ces tailles fixes, qui ne tiennent pas compte de la taille de l'écran et empêchent aussi la personnalisation de l'affichage dans le navigateur.

Unités de taille relatives (conseillées)

Les unités de taille relatives sont fonction de la taille du texte environnant ou du nombre de points sur l'écran.

Voici les mesures disponibles :

- **em** (largeur d'une majuscule comme M) ;
- **ex** (hauteur d'une minuscule comme x, souvent arrondie à 0.5em) ;
- `%` (100 % = 1em pour `font-size`) ;
- **px** (pixel = un point de l'écran ; en théorie il y a 72 ou 96 pixels par pouce).

> ATTENTION **Balises imbriquées**
>
> Dans les balises imbriquées, la taille d'un élément définie avec ces unités (sauf les pixels) est relative à celle du bloc parent. Par exemple, si la feuille de style contient :
> `p, span { font-size: 2em; }`
> alors la ligne HTML suivante :
> `<p> texte 1 <span> texte 2 </span></p>`
> affichera texte 1 en taille 2em et texte 2 en taille 4em.

Tailles définies par mots-clés

Les tailles peuvent également être indiquées à l'aide de mots-clés, qui ressemblent aux dimensions d'un vêtement. Ces définitions sont moins précises, car interprétées de façons diverses par les différents navigateurs.

De la plus petite à la plus grande, les tailles disponibles sont :

```
xx-small, x-small, small, medium (taille standard),
large, x-large, xx-large.
```

Codage des couleurs

Les couleurs sont définies à l'aide de noms ou de codes numériques.

Noms de couleurs

À certaines couleurs « standard » ont été attribués des mots réservés : `blue`, `white`, `red`... La liste de ces mots-clés est donnée en annexe.

Code RVB

Le codage rouge-vert-bleu (RVB en français, RGB en anglais) consiste à préciser la quantité de chacune de ces couleurs, exprimée :

- en **décimal** : `rgb(255,0,0)` = rouge
- en **pourcentage** : `rgb(0,100%,0)` = vert
- en **hexadécimal** : `#0000ff` = bleu (deux chiffres hexadécimaux pour chaque couleur RVB)

> Remarque **Notation hexadécimale raccourcie**
>
> Il est possible d'utiliser une notation hexadécimale raccourcie, dans laquelle chaque chiffre hexadécimal doit être doublé pour obtenir le code réel de la couleur. Par exemple, `#00f` = bleu (équivalent à `#0000ff`)

> À noter **Noir et blanc**
>
> Il existe donc plusieurs façons d'écrire le code du noir et celui du blanc :
> - pour le noir,
>   `rgb(0,0,0)` = `#000` = `#000000` = `black` ;
> - pour le blanc,
>   `rgb(255,255,255)` = `rgb(100%,100%,100%)` = `#fff` = `#ffffff` = `white`.

Couleurs « sûres »

Il existe une liste de 216 couleurs RVB appelées « couleurs sûres », dont l'affichage sur l'écran est garanti sur toutes les configurations matérielles et logicielles.

En hexadécimal, chacune des composantes RVB de ces couleurs sûres vaut 00, 33, 66, 99, cc ou ff.

La plupart des systèmes pouvant aujourd'hui afficher plusieurs millions de couleurs en RVB (en général, 16 277 216 couleurs possibles), il n'est plus nécessaire de se restreindre à ces couleurs sûres.

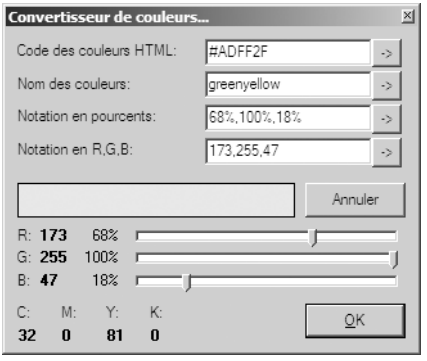


FIGURE 3–12 *Le logiciel PsPad propose un convertisseur de couleurs, entre code hexadécimal, nom de couleur et notation RGB en pourcentage ou en décimal.*

Exemple de page avec feuille de style interne

Voici une page XHTML simplifiée, avec sa feuille de style interne.

La lecture du code et de ses commentaires nous montre une mise en pratique des notions vues jusqu'ici.

Certes, les propriétés et leur utilisation n'ont pas encore été détaillées, mais celles qui figurent ici sont simples à comprendre.

```
<html>
<head>

<meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
<title>Garage des Tacots - Page d'accueil</title>
```

```
<style type="text/css">
<!--
/* Pour toute la page :
   texte centré, fond gris clair */
body { text-align: center; background-color: silver;}
/* Tous les titres h1 sont en marron, en taille 250%
   et sur fond blanc */
h1 { color: brown; font-size: 250%;
     background-color: white; }
/* Tous les titres h2 sont en bleu,
   avec une marge de 30 pixels autour */
h2 { color: blue; margin: 30px; }
/* Toutes les balises de classe "titre"
   sont en vert */
.titre { color: green; }
/* Les titres h1 de classe "titre" sont en Arial,
   en taille 280% et encadrés d'un trait plein */
h1.titre { font-family: Arial, sans-serif;
           font-size: 280%; border: solid; }
/* Les titres h2 de classe "titre" sont en italique
   et en taille 150% */
h2.titre { font-style: italic; font-size: 150%; }
-->
</style>

</head>

<body>

<h1 class="titre">Garage des Tacots</h1>
<h2 class="titre">Voitures anciennes</h2>
<br />

<h1>Nos services</h1>
<h2>Peinture et retouches</h2>
<h2>Pièces sur mesure</h2>
<h2>Pneumatiques toutes dimensions</h2>

</body>

</html>
```

FIGURE 3–13 *Affichage de la page « Garage des tacots » dans un navigateur web*

Vous pouvez recopier ce code dans un fichier texte, l'enregistrer avec l'extension `.html` et l'afficher dans votre navigateur web en double-cliquant sur ce fichier.

À partir de cet exemple, n'hésitez pas à modifier les propriétés, les niveaux de titre et les classes, pour mieux comprendre leur fonctionnement.

Une présentation très différente de cette page peut être obtenue en modifiant seulement *les valeurs* de certaines propriétés. La figure 3-14 en donne une illustration, à partir de la feuille de style suivante :

```
<style type="text/css">
<!--
/* Pour toute la page :
   texte aligné à gauche, fond bleu clair "lavande" */
body {text-align: left; background-color: lavender;}
/* Titres h1 : en blanc, taille 200%, fond gris */
h1 { color: white; font-size: 200%;
     background-color: silver; }
```

```
/* Titres h2 : en noir, marges de 10 pixels */
h2 { color: black; margin: 10px; }
/* Balises de classe "titre" : en bleu */
.titre { color: blue; }
/* Titres h1 de classe "titre" : en Courier New,
   taille 250% et encadré avec des tirets */
h1.titre { font-family: "Courier New", monospace;
           font-size: 250%; border: dashed; }
/* Titres h2 de classe "titre" : en italique,
   taille 190% */
h2.titre { font-style: italic; font-size: 190%; }
-->
</style>
```

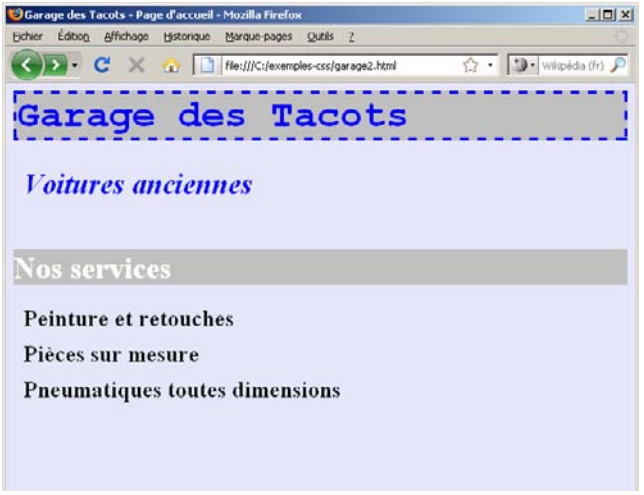


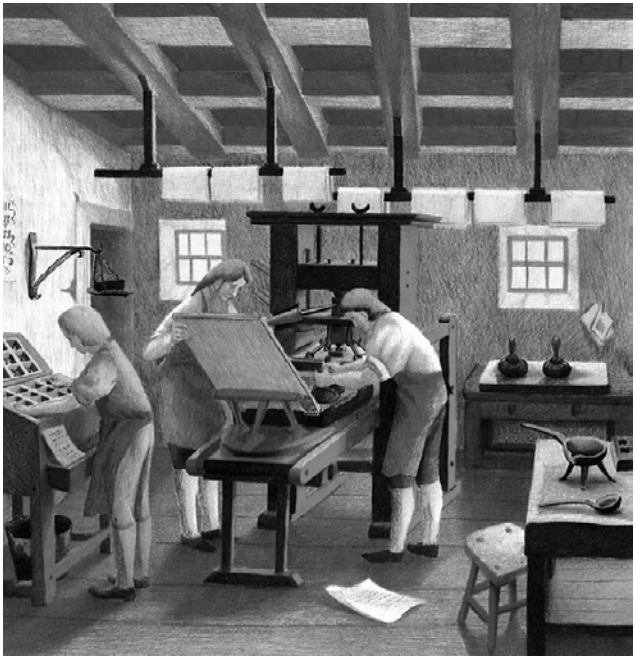
Figure 3–14 *Une autre version de la page précédente, où seules ont été modifiées les valeurs de certaines propriétés dans la feuille de style.*

Nous avons maintenant compris la structure d'une feuille de style CSS et son utilisation en liaison avec le code XHTML. Les exemples précédents nous ont d'ailleurs permis de connaître quelques-unes des propriétés et de leurs valeurs possibles.

Le chapitre qui suit nous emmène vers la découverte méthodique des propriétés CSS qui serviront à mettre en forme le texte de nos pages.

chapitre

4



Propriétés de mise en forme

Détaillons les différentes propriétés de mise en forme : nom, syntaxe, valeurs possibles, héritage. Des exemples illustrent chacune de ces propriétés.

Sommaire

Après avoir découvert le principe des feuilles de style et de leur écriture, nous voici dans le vif du sujet : les propriétés disponibles en CSS 2. Sans avoir à les apprendre par cœur, il est quand même utile d'en connaître l'existence, pour penser à les utiliser.

Mise en forme des caractères

Donnez du style à vos textes ! Vous allez pouvoir leur conférer tantôt une allure qui détonne, tantôt un aspect discret, bref tout ce qu'il faut pour enjoliver votre prose (ou vos poèmes !) à la manière d'un traitement de texte.

Sont regroupées ici toutes les propriétés qui peuvent s'appliquer à un seul ou plusieurs caractère(s). Toutefois, elles sont généralement utilisées pour mettre en forme des mots ou des paragraphes entiers.

Choix des polices

Le choix d'une police de caractères s'effectue à l'aide de la propriété `font-family`.

TABLEAU 4–1 **Propriété font-family**

Propriété	`font-family`

Exemples	`p {font-family: Arial, Verdana, sans-serif;}`
	`p {font-family: "Times New Roman", serif;}`
	`p {font-family: "Courier New" , monospace;}`

Valeurs possibles	**Noms de polices de caractères**, séparés par des virgules (les noms en plusieurs mots étant à mettre entre guillemets), ou type de police générique : `serif`, `sans-serif`, `monospace`.
Héritage	Propriété *héritée* : elle se transmet dans les balises imbriquées.

Important **Choix des polices**

Une police de caractères ne s'affichera que si elle est installée sur l'ordinateur utilisé pour consulter la page web. Il vaut donc mieux :

- Éviter les polices « personnelles », qui ne sont pas installées en standard par Windows, Linux et Mac OS, mais qui ont été ajoutées sur votre ordinateur (par exemple, l'installation de certains logiciels ajoute des polices automatiquement).
- Proposer plusieurs polices de caractères : si le navigateur ne trouve pas la première police sur l'ordinateur utilisé, il prendra la deuxième ; s'il ne la trouve pas non plus, il prendra la troisième, et ainsi de suite...
- Terminer la liste par une police « générique » : **`serif`**, **`sans-serif`**, **`monospace`** (police à chasse fixe, du type Courier) ; il existe aussi les types **`cursive`** et **`fantasy`**, très peu utilisés.

À noter **Polices courantes**

Les polices « standard », qu'on a toutes les chances de trouver sur un PC ou un Mac, sont les suivantes : **`Arial`**, **`Comic`**, **`Courier`**, **`Georgia`**, **`Helvetica`**, **`Times`**, **`Trebuchet`**, **`Verdana`**.
D'autres polices sont assez souvent présentes, tant sur les PC que sur les Mac, pour être utilisables à condition de ne pas oublier de mentionner une police de remplacement dans la propriété `font-family` : *Arial Black*, *Arial Narrow*, *Century Gothic*, *Impact*, *Palatino* et *Tahoma*.

Taille de police

La propriété **`font-size`** permet de préciser la taille des caractères.

Tableau 4–2 **Propriété font-size**

Propriété	**`font-size`**
Exemples	`h1 { font-size: 150%; }` `p { font-size: 15px; }`
Valeurs possibles	*taille relative* (conseillée) en **`em`**, **`ex`**, **`%`** ou **`px`** ou *taille fixe* en **`pt`**, **`pc`**, **`cm`**, **`mm`**, **`in`** ou *mot-clé* **`xx-small`**, **`x-small`**, **`small`**, **`medium`** (= standard), **`large`**, **`x-large`**, **`xx-large`**

TABLEAU 4–2 **Propriété font-size (suite)**

Pourcentages	% de la taille de la police dans l'élément parent.
Héritage	Propriété *héritée.*

> ATTENTION **Tailles relatives**
>
> Mises à part celles exprimées en pixels, les tailles relatives sont exprimées par rapport à la taille de police de la balise qui contient l'élément. Dans l'exemple suivant :
> `<p>Bonjour <span>tout le monde</span></p>`
> associé à la règle de style :
> `p, span { font-size: 80%; }`
> la taille du texte de la balise `<span>` est 64 % de celle de la taille initiale. En effet, puisque la règle s'applique pour chacune des balises, les caractères de la balise imbriquée `<span>` ont une taille égale à 80 % de 80 % de la taille de base.

Couleur du texte

Les valeurs attribuées à la propriété **`color`**, pour la couleur du texte, peuvent s'exprimer soit à l'aide d'un mot-clé, soit avec un code numérique hexadécimal, un nombre entier ou un pourcentage.

TABLEAU 4–3 **Propriété color**

Propriété	**`color`**

Exemples	`body { color: #0000ff; }`
	`h1.menu { color: #6e05c3; }`
	`.utile { color: rgb(255,0,0); }`
	`a { color: rgb(30%,80%,30%) }`
	`em { color: green; }`

Valeurs possibles	**nom de couleur** prédéfini ou **code RVB**
Héritage	Propriété *héritée.*

> **Attention** **Nom de la propriété color**
>
> La propriété qui change la couleur du texte est bien **`color`**. Une erreur courante consiste à écrire *font-color*, alors que ce nom de propriété n'existe pas, même s'il eût été logique pour indiquer la couleur de police.

Figure 4–1 *Utilisation des propriétés* `font-family` *(type de police),* `font-size` *(taille des caractères) et* `color` *(couleur d'écriture)*

Texte en gras

Avec la propriété **`font-weight`**, il s'agit de préciser l'épaisseur d'écriture, c'est-à-dire le degré de graissage de la police.

Tableau 4–4 **Propriété font-weight**

Propriété	**`font-weight`**
Exemple	`.principal { font-weight: bold; }`
Valeurs possibles	**`normal`** (valeur par défaut), **`bold`** : gras, **`lighter`** : moins gras que le style en cours, **`bolder`** : plus gras que le style en cours, ou un nombre (**`100`**, **`200`**… **`900`**) qui définit le niveau de gras
Héritage	Propriété *héritée*. Utiliser la valeur *normal* pour annuler l'héritage.

> À NOTER **Valeurs numériques de gras**
>
> Dans l'échelle des valeurs numériques qui définissent le niveau de gras, `400` correspond à `normal` et `700` à `bold`.
> Toutefois, l'utilisation de nombres est déconseillée avec `font-weight`, car ceux-ci ne sont pas bien pris en compte par les différents navigateurs. Dans la pratique, il vaut mieux s'en tenir aux valeurs de base `normal` et `bold`.

Italique

Pour mettre des mots en italique, c'est la propriété **`font-style`** qui nous sera utile.

TABLEAU 4–5 **Propriété font-style**

Propriété	**`font-style`**
Exemple	`.remarque { font-style: italic; }`
Valeurs possibles	**`normal`** : écriture droite (valeur par défaut), **`italic`** : italique, **`oblique`** : police de type « oblique » (rare).
Héritage	Propriété *héritée*. Utiliser la valeur *normal* pour annuler l'héritage.

Soulignement et autres « décorations »

La propriété **`text-decoration`** permet d'obtenir différents effets : trait au-dessus ou en dessous des mots, texte barré ou clignotant.

TABLEAU 4–6 **Propriété text-decoration**

Propriété	**`text-decoration`**

Exemples	`a:hover { text-decoration: none; }`
	(suppression du soulignement des liens au passage de la souris)
	`h1 { text-decoration: underline overline; }`
	(titres de niveau 1 soulignés et surlignés)

Tableau 4–6 Propriété text-decoration (suite)

Valeurs possibles	**none** : supprime tout soulignement et toute autre valeur attribuée à `text-decoration` (valeur par défaut), **underline** : souligné, **overline** : surligné (trait au-dessus), **line-through** : barré, **blink** : clignotement du texte.
Héritage	Cette propriété n'est *pas héritée*, contrairement aux autres propriétés liées aux caractères.

À noter **Utilisation de `text-decoration`**

- La prise en compte par les navigateurs de **blink** (clignotement du texte) est facultative.
- Il est possible de spécifier plusieurs valeurs pour `text-decoration`. Par exemple, la règle suivante permet d'obtenir des titres de niveau 1 qui sont à la fois soulignés et surmontés d'un trait horizontal :

```
h1 { text-decoration: underline overline; }
```

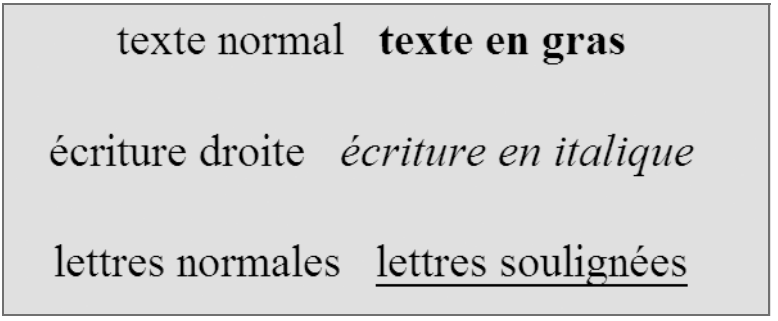


Figure 4–2 *Mise en œuvre des propriétés* `font-weight` *(gras),* `font-style` *(italique) et* `text-decoration` *(soulignement)*

Majuscules et minuscules

Il est parfois utile de demander au navigateur d'afficher une partie du texte en minuscules ou en majuscules, quelle que soit la façon dont il aura été écrit initialement. C'est la propriété **`text-transform`** qui se charge de cette opération.

TABLEAU 4–7 **Propriété text-transform**

Propriété	`text-transform`
Exemple	`.pays { text-transform: uppercase; }`
Valeurs possibles	**`capitalize`** : première lettre de chaque mot en majuscule, **`lowercase`** : tout en minuscules, **`uppercase`** : tout en majuscules, **`none`** : écriture standard (valeur par défaut).
Héritage	Propriété *héritée*. Utiliser la valeur *none* pour annuler l'héritage.

Petites majuscules

Pour mettre un texte en majuscules sans qu'il ne soit trop voyant, la propriété **`font-variant`** nous propose de l'écrire en petites majuscules : ces majuscules ont à peu près la taille des minuscules.

TABLEAU 4–8 **Propriété font-variant**

Propriété	`font-variant`
Exemple	`.ville { font-variant: small-caps; }`
Valeurs possibles	**`normal`** : texte normal (valeur par défaut), **`small-caps`** : tout en petites majuscules.
Héritage	Propriété *héritée*. Utiliser la valeur *normal* pour annuler l'héritage.

Surlignage de lettres

Grâce à la propriété **`background-color`**, il est possible d'attribuer un fond de couleur à certaines lettres ou à certains mots, à la manière du surligneur sur papier. Cette propriété servira aussi à choisir la couleur de fond d'un paragraphe ou d'un bloc, comme nous le verrons plus loin.

TABLEAU 4–9 **Propriété background-color pour le texte**

Propriété	`background-color`
Exemple	`strong { background-color: red; }` `p.remarque { background-color: #0000ff; }`

Tableau 4–9 Propriété background-color pour le texte (suite)

Valeurs possibles	**nom de couleur** prédéfini ou **code RVB** ; `transparent` (valeur par défaut)
Héritage	Cette propriété n'est *pas héritée*, mais la valeur par défaut `transparent` laisse voir la couleur de l'élément conteneur ou qui se trouve en dessous.

Décalage vers le haut ou le bas

Le décalage vertical de lettres ou de mots avec la propriété `vertical-align` sert dans deux cas de figure :

- pour l'affichage d'éléments en *indice* ou en *exposant* (attention : il s'agit d'un simple décalage en hauteur, il faudra y ajouter une réduction de la taille des lettres) ;
- pour le *centrage vertical d'une image* sur une ligne.

Cette propriété `vertical-align` est également applicable aux *cellules d'un tableau*, avec les valeurs suivantes : `baseline`, `top` (en haut), `middle` (au milieu) et `bottom` (en bas).

Tableau 4–10 Propriété vertical-align

Propriété	`vertical-align`
Exemples	`.exposant { vertical-align: super; }` `.indice { vertical-align: -50%; }`
Valeurs possibles	**`baseline`** : sur la base de la ligne (valeur par défaut), **`sub`** : indice, **`super`** : exposant, **`middle`** : au milieu de la ligne (centrage vertical sur la ligne), **`text-top`** ou **`text-bottom`** : alignement avec le haut ou le bas de la boîte parente, **valeur** ou **pourcentage** : valeur *positive* pour un décalage vers le *haut*, *négative* pour un décalage vers le *bas*.
Héritage	Cette propriété n'est *pas héritée*.

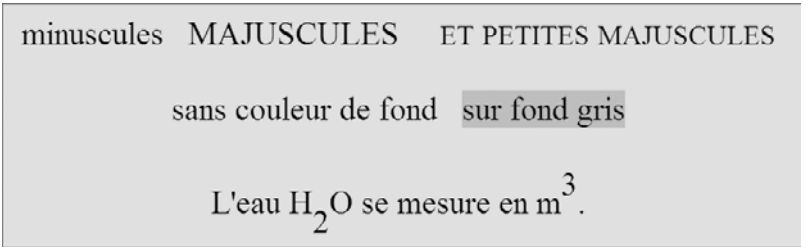


FIGURE 4–3 *Application des propriétés* `text-transform` *(majuscules),* `font-variant` *(petites majuscules),* `background-color` *(couleur de fond = surlignage) et* `vertical-align` *(indice/exposant)*

Raccourci pour la mise en forme de caractères

Le raccourci **`font`** permet d'indiquer, en une seule propriété, les mises en forme qui concernent l'italique, les petites majuscules, le graissage, la taille et la police de caractères.

TABLEAU 4–11 **Propriété raccourcie font**

Propriété	**`font`**

Exemples	`h3 { font: bold 1.2em Verdana, sans-serif; }`
	`#note1 { font: italic 80% Garamond, serif; }`
	`.ville {font: bold small-caps 2em Times, serif;}`

Valeurs possibles	Valeurs des propriétés `font-style`, `font-variant`, `font-weight`, `font-size`, `line-height` (hauteur de ligne, voir plus loin) et `font-family`.
Héritage	Propriétés *héritées*.

IMPORTANT **Utilisation du raccourci `font`**

- La propriété **`font-family`** est **obligatoire**, les autres sont facultatives.
- Les propriétés qui ne sont pas fournies sont réinitialisées à `normal`.
- Ce raccourci n'inclut pas les propriétés `color`, `text-decoration`, `text-transform`, `background-color` et `vertical-align`.

Paragraphes et blocs de texte

Nos mots étant mis en forme, penchons-nous à présent sur les propriétés qui s'appliquent à des paragraphes ou à des blocs de texte tout entiers.

Alignement horizontal du texte

La propriété `text-align` modifie l'alignement horizontal comme le ferait un traitement de texte : paragraphe aligné à gauche, centré, aligné à droite ou justifié.

> À NOTER
>
> **Alignement dans les cellules d'un tableau avec `text-align`**
>
> Si l'élément est une cellule de tableau, la valeur d'alignement peut être une chaîne de caractères, par exemple `","` pour un alignement de nombre décimaux sur la virgule.

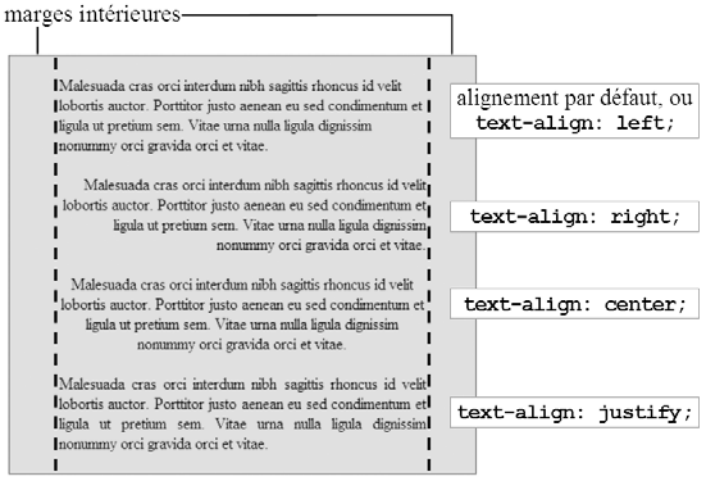


Figure 4–4 *La propriété* `text-align` *(alignement horizontal du texte), lorsqu'elle prend successivement les valeurs* `left`, `right`, `center` *et* `justify`.

TABLEAU 4–12 **Propriété text-align**

Propriété	`text-align`
Exemples	`p { text-align: justify; }` `.auteur { text-align: right; }`
Valeurs possibles	**`left`** : aligné à gauche (par défaut), **`right`** : aligné à droite, **`center`** : centré, **`justify`** : justifié.
Héritage	Propriété *héritée*. Pour retrouver la valeur initiale, utiliser `left`.

Retrait de première ligne

Il s'agit de créer, avec la propriété **`text-indent`**, un retrait à gauche qui ne s'applique qu'à la première ligne de chacun des paragraphes concernés. Les autres lignes débutent sur la marge de gauche.

> ASTUCE **Retrait négatif de première ligne**
>
> Pour obtenir un « retrait négatif de première ligne », c'est-à-dire tout le paragraphe en retrait sauf la première ligne, il suffit d'écrire par exemple :
> `p { text-indent: -5em; padding-left: 5em; }`
> La première ligne reste alors à sa place habituelle et le reste du paragraphe est en retrait de 5em (voir la figure 4-5).

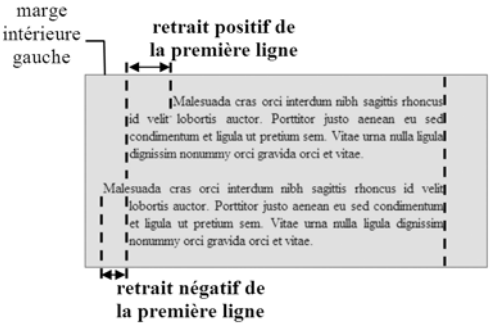


FIGURE 4–5 *Retraits positif et négatif de la première ligne d'un paragraphe, avec la propriété* `text-indent`

Tableau 4–13 Propriété text-indent

Propriété	`text-indent`
Exemples	`p { text-indent: 5em; }`
Valeurs possibles	**valeur positive ou négative**, pour un retrait respectivement vers la droite ou vers la gauche de la première ligne ; valeur par défaut : 0. Mêmes unités que les tailles de polices de caractères, % inclus.
Pourcentages	% de la largeur du bloc conteneur.
Héritage	Propriété *héritée*. Utiliser la valeur 0 pour annuler cet héritage.

Interligne minimum

La hauteur d'une ligne de texte peut être modifiée à l'aide de la propriété `line-height`.

Tableau 4–14 Propriété line-height

Propriété	`line-height`
Exemple	`a.menu { line-height: 2em; }`
Valeurs possibles	`normal` (valeur standard) ou **valeur positive** pour régler l'espacemententre les lignes Mêmes unités que les tailles de polices, ou % de cette taille.
Pourcentage	% de la taille de police utilisée.
Héritage	Propriété *héritée*. Utiliser la valeur *normal* pour annuler l'héritage.

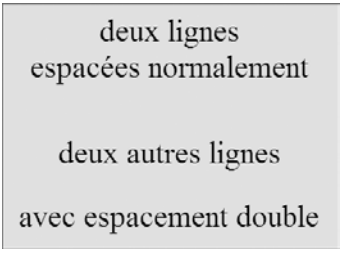


Figure 4–6 *Modification de l'interlignage avec la propriété* `line-height`

Espacement entre les lettres

La propriété `letter-spacing` fixe l'espacement entre les lettres d'un mot, aussi appelé *crénage* ou encore *interlettrage*.

TABLEAU 4–15 **Propriété letter-spacing**

Propriété	`letter-spacing`
Exemples	`.pluslarge { letter-spacing: 0.5em; }` `.moinslarge { letter-spacing: -1px; }`
Valeurs possibles	`normal` (par défaut), valeur **positive** ou **négative**, respectivement pour augmenter ou diminuer l'interlettrage. Mêmes unités que les tailles de polices, sauf %.
Héritage	Propriété *héritée*. Utiliser la valeur *normal* pour annuler l'héritage.

Espacement entre les mots

Comme son nom l'indique, la propriété `word-spacing` ajuste l'espacement entre les mots.

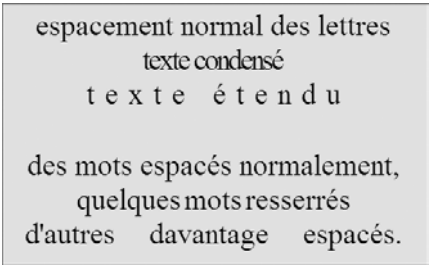


FIGURE 4–7 *Propriétés* `letter-spacing` *pour condenser ou étirer le texte,* `word-spacing` *pour écarter ou resserrer les mots entre eux*

TABLEAU 4–16 **Propriété word-spacing**

Propriété	`word-spacing`
Exemples	`.grandsespaces { word-spacing: 0.5em; }` `.petitsespaces { word-spacing: -2px; }`

Tableau 4–16 Propriété word-spacing (suite)

Valeurs possibles	`normal` (par défaut), valeur **positive** ou **négative**, respectivement pour augmenter ou diminuer l'espace entre les mots. Mêmes unités que les tailles de polices, sauf %.
Héritage	Propriété *héritée*. Utiliser la valeur *normal* pour annuler l'héritage.

Conservation des espaces et sauts de ligne saisis

Appliquée à un paragraphe ou un bloc de texte, la propriété `white-space` indique s'il faut ou non conserver tous les espaces et retours à la ligne tapés dans le code XHTML. En même temps, elle permet d'accepter ou non le retour à la ligne automatique, qui se produit normalement lorsque le texte arrive sur le bord droit du bloc.

Tableau 4–17 Propriété white-space

Propriété	`white-space`
Exemple	`p.extraits { white-space: pre; }`
Valeurs possibles	`normal` (par défaut) : affichage d'un seul espace à la place de tous les espaces et retours à la ligne qui séparent deux mots ; le retour à la ligne s'effectue avec la balise ` ` et est automatique en fin de ligne du bloc de texte. `nowrap` : espaces successifs fusionnés et retours à la ligne ignorés comme pour la valeur `normal` ; par contre, il n'y a pas de retour automatique en fin de ligne, seul ` ` peut produire un retour à la ligne. `pre` : conservation de tous les espaces et de tous les retours à la ligne saisis dans le code source, mais il n'y a pas de retour automatique en fin de ligne du bloc de texte. `pre-line` : espaces successifs fusionnés en un seul, conservation des retours à la ligne du code source ; le retour automatique en fin de ligne est activé. `pre-wrap` : tous les espaces du code source sont conservés, mais les retours à la ligne qui ne sont pas indiqués par ` ` sont ignorés ; le retour automatique en fin de ligne est activé.
Héritage	Propriété *héritée*. Utiliser la valeur *normal* pour annuler l'héritage.

Modification du curseur de la souris

La propriété `cursor` n'a pas d'effet sur l'affichage de la page en elle-même, mais sur l'apparence du curseur de la souris lorsqu'il passe sur le texte concerné.

TABLEAU 4–18 **Propriété cursor**

Propriété	`cursor`
Exemple	`.aide { cursor: help }`
Valeurs possibles	**`auto`** (valeur par défaut) : la forme est fonction du contexte, **`default`** : généralement une flèche blanche, **`crosshair`** : croix noire, **`pointer`** : main, **`move`** : quadruple-flèche de déplacement, et doubles-flèches de redimensionnement orientées (nord n - sud s - est e - ouest w) : **`n-resize`**, **`s-resize`**, **`e-resize`**, **`w-resize`**, **`ne-resize`**, **`sw-resize`**, **`nw-resize`**, **`se-resize`**. Voir les différents types de curseurs sur la figure 4-8.
Héritage	Propriété *héritée*. Utiliser la valeur *normal* pour annuler l'héritage.

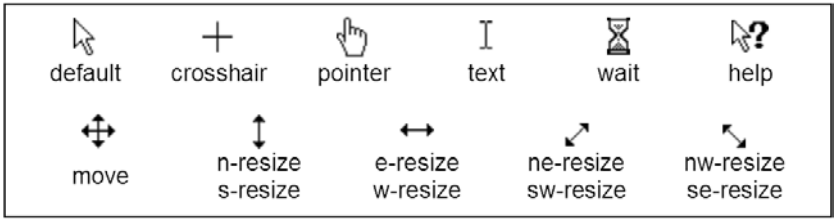


FIGURE 4–8 *Différents types de curseur*

Affichage automatique d'un contenu

La propriété `content` s'utilise avec les pseudo-éléments **`:before`** et **`:after`**. Elle permet d'afficher automatiquement, avant ou après l'élément concerné, un texte, un numéro, des guillemets, une image, etc.

Tableau 4–19 Propriété content

Propriété	`content`

Exemples	`p.note:before { content: "Nota bene : "; }`
	`p.remarque:before {content: url(crayon.gif);}`
	`li:before { content: "["`
	`counter(chapitre,lower-roman) "]";}`
	`.citation.before { content: open-quote; }`
	`.citation.after { content: close-quote; }`
	`img:after { content: attr(title); }`

Valeurs possibles	**chaîne de caractères** : à écrire entre guillemets (valeur par défaut : la chaîne vide `""`). **`url(fichier)`** : fichiers à utiliser (image à afficher, son à jouer, etc.). **`counter(nom)`** ou **`counter(nom,style)`** : style à choisir parmi les valeurs possibles de `list-style-type` (style par défaut : décimal), **`counters(nom, chaîne)`** ou **`counters(nom, chaîne, style)`** : avec chaîne séparatrice (par exemple `". "` pour obtenir §3 puis § 3.1, 3.2, ...). **`open-quote`**, **`close-quote`** : guillemets de début/fin définis par la propriété `quote`, **`no-open-quote`**, **`no-close-quote`** : pas de guillemets, mais un niveau d'imbrication de guillemets est décompté, pour les prochains guillemets qui seront affichés par content, **`attr(propriété)`** : valeur de l'attribut indiqué de l'élément, chaîne vide si l'attribut est absent.
Héritage	*Non.*

Guillemets à utiliser

La propriété **`quotes`** sert à définir les types de guillemets qui serviront aux différents niveaux d'imbrication.

Ces guillemets seront utilisables avec la balise <q> ou la propriété `content` lorsqu'elle affiche des guillemets ouvrants ou fermants : `content: open-quote;` ou `content: close-quote;`

TABLEAU 4–20 **Propriété quotes**

Propriété	`quotes`
Exemples	`q { quotes: '"' '"' "'" "'"; }` `q.guill2 { quotes: "«" "»" "<" ">"; }`
Valeurs possibles	Sous forme de chaînes de caractères : • guillemets d'ouverture puis de fermeture, pour le premier niveau de guillemets ; • puis éventuellement guillemets d'ouverture, puis de fermeture pour le deuxième niveau (guillemets imbriqués), etc. • et ainsi de suite pour le nombre de niveaux d'imbrication souhaités.
Héritage	Propriété *héritée*.

Réinitialisation d'un compteur

Il s'agit, avec la propriété `counter-reset`, de remettre à zéro ou d'initialiser à une valeur définie le compteur dont le nom est indiqué.

Ce compteur sera utilisable par la propriété `content`.

TABLEAU 4–21 **Propriété counter-reset**

Propriété	`counter-reset`
Exemples	`h1 { counter-reset: chapitre; }` `h1.nouveau { counter-reset: numpage -1; }`
Valeurs possibles	**Nom du compteur** puis éventuellement **valeur initiale** (si elle est différente de 0) : *nombre entier* positif ou négatif, ou **none** = pas de compteur (c'est la valeur par défaut).
Héritage	*Non*.

ATTENTION **Réinitialisation de plusieurs compteurs**

Si deux compteurs doivent être réinitialisés pour le même élément, il faut réunir ces deux réinitialisations, en n'écrivant qu'une seule fois la propriété `counter-reset`. Exemple :

```
h1 { counter-reset: page 2 section 1; }
```

Incrémentation d'un compteur

Chaque fois qu'un compteur est utilisé à l'aide de la propriété `content`, il s'incrémente d'une valeur donnée, qui peut être définie à l'aide de la propriété `counter-increment`.

Tableau 4–22 Propriété counter-increment

Propriété	`counter-increment`
Exemples	`h2 { counter-increment: chapitre; }` `.instruction { counter-increment: numligne 10; }`
Valeurs possibles	**Nom du compteur** puis éventuellement la **valeur d'incrémentation** (si elle est différente de 1) : nombre entier positif ou négatif, ou **none** = pas d'incrémentation du compteur (valeur par défaut).
Héritage	*Non*.

> À noter **Ordre des opérations**
> L'incrémentation du compteur s'effectue **avant** son utilisation.

Sens de l'écriture

Certaines langues qui s'écrivent de droite à gauche nécessitent l'emploi de la propriété **`direction`** pour préciser le sens de lecture.

Tableau 4–23 Propriété direction

Propriété	`direction`
Exemple	`body { direction: ltr; }` `.yiddish { direction: rtl; }`
Valeurs possibles	**`ltr`** : de gauche à droite (*left to right*) - valeur par défaut ; **`rtl`** : de droite à gauche (*right to left*).
Héritage	Propriété *héritée*.

> À NOTER **Caractères Unicode**
>
> Le sens de lecture des différents encodages *Unicode* est reconnu automatiquement. Leur utilisation nous dispense donc de spécifier cette propriété `direction`.

Écriture bidirectionnelle

Cette propriété **`unicode-bidi`** (texte unicode bidirectionnel) est rarement utilisée ; elle permet d'utiliser plusieurs sens de lecture dans un même bloc.

Elle peut servir aux amateurs d'exotisme, pour inclure des citations en arabe, farsi, hébreu ou urdu à l'intérieur d'un paragraphe en français...

TABLEAU 4–24 **Propriété unicode-bidi**

Propriété	**`unicode-bidi`**

Exemples	`span.citation { direction: rtl;`
	`unicode-bidi: embed;}`
	`span.sens2 { direction: rtl;`
	`unicode-bidi: bidi-override; }`

Valeurs possibles	**`normal`** (par défaut) : à l'intérieur de chaque groupe de mots homogène, les caractères Unicode s'écrivent dans leur sens d'écriture naturel, fonction de la langue et reconnu automatiquement.
	`embed` : les caractères s'écrivent dans leur sens naturel, les groupes de mots homogènes (composés de caractères qui s'écrivent dans le même sens) sont placés dans l'ordre défini par la propriété **`direction`** (voir la figure 4-9). **`bidi-override`** : tous les caractères sont écrits les uns après les autres, dans le sens indiqué par la propriété **`direction`**.
Héritage	*Non*.

Exemple **Lignes comprenant des textes en français et en hébreu**

Cet exemple est inspiré du test de conformité CSS 2 de Daniel Glazman, disponible sur le site des Éditions Eyrolles (modèle de formatage, groupe de tests 6) à l'adresse suivante :

- **http://www.editions-eyrolles.com/css2/tests/vfm/vfm14.htm**

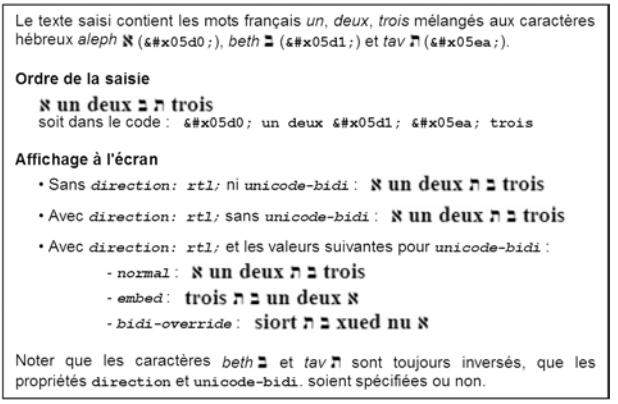


Figure 4–9 *Utilisation de la propriété* `unicode-bidi`

Bordures

Les propriétés de bordure s'appliquent aux blocs de texte et aux éléments remplacés, comme les images.

Style de bordure

La propriété `border-style` précise le type des traits de contour à afficher autour des blocs de texte concernés.

> Attention **Propriété obligatoire pour afficher une bordure**
>
> La valeur initiale de `border-style` étant `none`, il est indispensable de préciser un style de bordure pour en afficher une.
> Tant qu'un style de bordure n'a pas été défini, donner une épaisseur et une couleur de bordure ne change rien à l'affichage.

TABLEAU 4–25 **Propriété border-style**

Propriété	`border-style`
Exemples	`h2 { border-style: solid; }` `p.note { border-style: double; }`
Valeurs possibles	**`none`** (valeur par défaut) ou **`hidden`** : aucune bordure, **`solid`** : trait plein, **`dotted`** : pointillés, **`dashed`** : tirets, **`double`** : trait plein double, **`groove`** : en creux, **`ridge`** : en relief, **`inset`** : creux ombré, **`outset`** : relief ombré. Voir la figure 4-10.
Héritage	*Non.*

PRÉCISION **Différence entre `none` et `hidden`**

C'est dans les tableaux qu'il existe une différence entre **`none`** et **`hidden`** :

- **`none`** = aucune bordure, sauf si une cellule voisine en possède une ;
- **`hidden`** = aucune bordure, dans tous les cas.

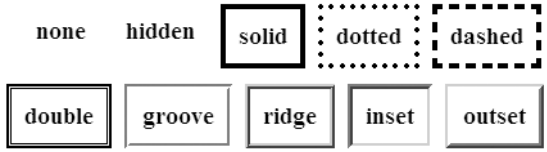


FIGURE 4–10 *Différents types de bordure*

Styles de bordure pour chaque côté

Il existe quatre propriétés distinctes pour définir un style de bordure sur chacun des quatre côtés de l'élément concerné.

TABLEAU 4–26 **Style de bordure pour chaque côté**

Propriétés	**`border-top-style`** : style de la bordure du haut, **`border-right-style`** : style de la bordure de droite, **`border-bottom-style`** : style de la bordure du bas, **`border-left-style`** : style de la bordure de gauche.

Raccourci **Utilisation de `border-style` pour singulariser chaque côté**

La propriété générale **`border-style`** peut être utilisée pour préciser le style de bordure sur chaque côté :

- avec **deux valeurs** : ① haut et bas, ② droite et gauche
```
p { border-style: solid double; }
```
- avec **trois valeurs** : ① haut, ② droite et gauche, ③ bas
```
p { border-style: dashed solid dotted; }
```
- avec **quatre valeurs** : ① haut , ② droite, ③ bas, ④ gauche
```
p { border-style: dashed dotted
                  solid double; }
```

Épaisseur de bordure

L'épaisseur du trait de contour est indiquée par la propriété **`border-width`**.

Tableau 4–27 **Propriété border-width**

Propriété	**`border-width`**
Exemple	`p.note { border-width: 2px; }`
Valeurs possibles	**`thin`** = bordure fine, **`medium`** = bordure moyenne, **`thick`** = bordure épaisse ou **valeur numérique** en **em**, **px**, ... (mais pas en %)
Héritage	*Non.*

Épaisseur de bordure pour chaque côté

L'épaisseur de bordure peut être précisée pour chacun des quatre côtés de l'élément concerné, à l'aide de quatre propriétés distinctes.

Tableau 4–28 **Épaisseur de bordure pour chaque côté**

Propriétés	**`border-top-width`** : épaisseur de la bordure du haut, **`border-right-width`** : épaisseur de la bordure de droite, **`border-bottom-width`** : épaisseur de la bordure du bas, **`border-left-width`** : épaisseur de la bordure de gauche.

RACCOURCI

Utilisation de `border-width` pour singulariser chaque côté

La propriété générale **`border-width`** peut être utilisée pour préciser l'épaisseur de bordure sur chaque côté :

- avec **deux valeurs** : ① haut et bas, ② droite et gauche
  ```
  p { border-width: 1em 2em; }
  ```
- avec **trois valeurs** : ① haut, ② droite et gauche, ③ bas
  ```
  p { border-width: thin medium thick; }
  ```
- avec **quatre valeurs** : ① haut , ② droite, ③ bas, ④ gauche
  ```
  p { border-width: 1px 3px 3px 1px; }
  ```

Couleur de bordure

Par défaut, le contour d'un bloc est de la même couleur que le texte (valeur de la propriété **`color`**, si elle a été définie pour cet élément, sinon noir par défaut). Pour modifier cette couleur, il faut utiliser la propriété **`border-color`**.

TABLEAU 4–29 **Propriété border-color**

Propriété	**`border-color`**
Exemple	`div.remarque { border-color: gray; }` `p.utile { border-color: #ff0088; }`
Valeurs possibles	**nom de couleur** prédéfini ou **code RVB** ; **`transparent`** = bordure invisible
Héritage	*Non.*

Couleur de bordure pour chaque côté

Quatre propriétés distinctes permettent de définir la couleur de bordure sur chacun des quatre côtés de l'élément concerné.

TABLEAU 4–30 **Couleur de bordure pour chaque côté**

Propriétés	**`border-top-color`** : couleur de la bordure du haut, **`border-right-color`** : couleur de la bordure de droite, **`border-bottom-color`** : couleur de la bordure du bas, **`border-left-color`** : couleur de la bordure de gauche.

Raccourci

Utilisation de `border-color` pour singulariser chaque côté

La propriété générale **`border-color`** peut être utilisée pour préciser la couleur de bordure sur chaque côté :

- avec **deux valeurs** : ① haut et bas, ② droite et gauche
  ```
  p { border-color: blue red; }
  ```
- avec **trois valeurs** : ① haut, ② droite et gauche, ③ bas
  ```
  p { border-color: blue gray green; }
  ```
- avec **quatre valeurs** : ① haut , ② droite, ③ bas, ④ gauche
  ```
  p { border-color: blue gray gray blue; }
  ```

Raccourci pour toutes les propriétés de bordure

L'ensemble des propriétés qui définissent les bordures (épaisseur, style et couleur) peut être déclaré à l'aide du raccourci **`border`**.

Tableau 4–31 **Propriété raccourcie border**

Propriété	**`border`**
Exemple	`h2.chapitre { border: 5px gray groove; }`
Valeurs possibles	Toutes les valeurs des propriétés **`border-width`** (facultative), **`border-style`** (obligatoire) et **`border-color`** (facultative).
Héritage	*Non.*

Important **Propriété `border` : le style de bordure est obligatoire**

Lorsqu'une des propriétés, épaisseur, style ou couleur de bordure, n'est pas précisée dans ce raccourci, elle est initialisée à sa valeur par défaut.

Il en résulte que *le style de bordure est obligatoire,* sa valeur par défaut étant none (pas de bordure, quelles que soient l'épaisseur et la couleur choisies).

Raccourci des propriétés de bordure pour chaque côté

Il existe quatre raccourcis distincts pour définir les propriétés de bordure sur chacun des quatre côtés de l'élément concerné.

TABLEAU 4–32 **Raccourcis des propriétés de bordure pour chaque côté**

Propriétés	**`border-top`** : propriétés de la bordure du haut, **`border-right`** : propriétés de la bordure de droite, **`border-bottom`** : propriétés de la bordure du bas, **`border-left`** : propriétés de la bordure de gauche.
Exemple	`.titre { border-top: 3px solid red;` `border-right: 1px dotted red;}`
Valeurs possibles	Toutes les valeurs des propriétés **`border-width`** (facultative), **`border-style`** (obligatoire) et **`border-color`** (facultative).
Héritage	*Non.*

Contour superposé à un élément

La propriété **`outline`** affiche une bordure qui se superpose à l'élément, sans augmenter ses dimensions.

Les propriétés utilisables sont les suivantes :

- **`outline-style`** : mêmes valeurs que `border-style`, sauf `hidden` ;
- **`outline-width`** : mêmes valeurs que `border-width` ;
- **`outline-color`** : mêmes valeurs que `border-color`, plus la valeur `invert` (inverse de la couleur de fond) ;
- **`outline`** : raccourci pour `outline-width`, `outline-style` et `outline-color`.

> À NOTER **Héritage et interprétation**
>
> - Ces propriétés `outline`, `outline-width`, `outline-style`, `outline-color` ne sont *pas héritées*.
> - Elles sont interprétées par Firefox depuis sa version 1.5, mais ne sont pas prises en compte par Internet Explorer 6 et 7.

Images et couleurs d'arrière-plan

Les propriétés suivantes s'appliquent aux blocs de texte et aux éléments remplacés, comme les images.

Elles servent à agrémenter l'arrière-plan de l'élément concerné, soit d'une image, soit d'une couleur unie. Attention cependant à éviter une cacophonie de couleurs !

Couleur d'arrière-plan

Il est possible de définir une couleur d'arrière-plan sur l'ensemble d'un bloc, à l'aide de la propriété `background-color`.

Tableau 4–33 Propriété background-color

Propriété	`background-color`
Exemple	`p.relief { background-color: yellow; }`
Valeurs possibles	**nom de couleur** prédéfini ou **code RVB** ; `transparent` (valeur par défaut).
Héritage	Cette propriété *n'est pas* héritée, mais la valeur par défaut `transparent` laisse voir la couleur de l'élément conteneur ou se trouvant en dessous.

Image d'arrière-plan

La propriété `background-image` permet d'afficher une image en arrière-plan d'un bloc. Sa valeur est l'adresse de l'image, qui peut être relative (nom du fichier image situé par rapport au dossier qui contient la page web) ou absolue (URL se terminant par un nom de fichier image).

Tableau 4–34 Propriété background-image

Propriété	`background-image`

Exemples	`body {background-image:`
	`  url(images/maison.png);}`
	`.pub {background-image:`
	`  url(http://www.sncf.com/logo.gif);}`

TABLEAU 4–34 **Propriété background-image (suite)**

Valeurs possibles	`url` (nom d'image avec chemin relatif ou absolu) ou **none** : aucune image (valeur par défaut).
Héritage	*Non.*

> À NOTER **Guillemets autour des noms de fichiers images**
>
> Grâce aux parenthèses de `url(...)`, les guillemets ou apostrophes, qui logiquement entourent le nom du fichier image ou l'URL, sont *facultatifs*.

Répétition ou non de l'image d'arrière-plan

Lorsque l'image est en arrière-plan d'un bloc et possède une dimension inférieure à celui-ci, elle est automatiquement répétée, horizontalement et verticalement. Pour annuler l'une de ces répétitions ou les deux, il faut l'indiquer à l'aide de la propriété `background-repeat`.

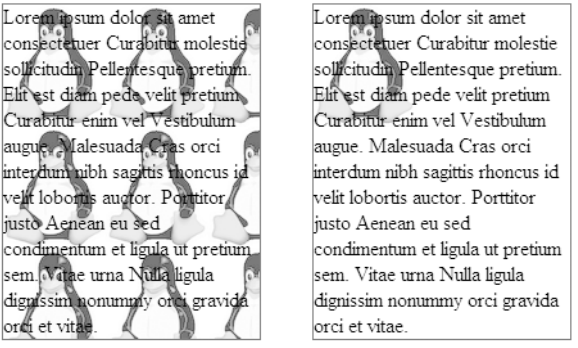


FIGURE 4–11 *Image d'arrière-plan plus petite que le bloc, avec répétition (valeur par défaut), puis avec la propriété* `background-repeat: no-repeat`

TABLEAU 4–35 **Propriété background-repeat**

Propriété	`background-repeat`
Exemples	`body { background-repeat: repeat-y; }` `.pub { background-repeat: no-repeat; }`

Tableau 4–35 Propriété background-repeat (suite)

Valeurs possibles	`repeat` : répétition horizontale et verticale (valeur par défaut). `repeat-x` : répétition horizontale seulement. `repeat-y` : répétition verticale seulement. `no-repeat` : pas de répétition.
Héritage	*Non*.

Alignement de l'image d'arrière-plan

Avec la propriété `background-position`, il est possible de préciser la position horizontale et la position verticale de l'image d'arrière-plan à l'intérieur du bloc dont elle constitue le fond.

Cette position est exprimée par rapport aux bords de l'élément.

Tableau 4–36 Propriété background-position

Propriété	`background-position`
Exemples	`body { background-position: center top; }` `.pub { background-position: left center; }`

Valeurs possibles	**Une ou deux valeurs**, données par des **noms** ou des **nombres** (dimension relative ou absolue, souvent exprimée en %).
	Première valeur pour l'alignement **horizontal** :
	`left`, `center` (valeur par défaut), `right`
	ou **nombre** (`0%` = `left`, `100%` = `right`).
	Deuxième valeur pour l'alignement **vertical** :
	`top`, `center` (valeur par défaut), `bottom`
	ou **nombre** (`0%` = `top`, `100%` = `bottom`).

Pourcentage	% de la taille de la boîte elle-même.
Héritage	*Non*.

À noter Utilisation de nombres pour `background-position`

- Ne mélangez pas valeurs fixes (par exemple 10px) et relatives (comme 20%).
- Il est possible d'indiquer des nombres négatifs, pour « rogner » l'image.

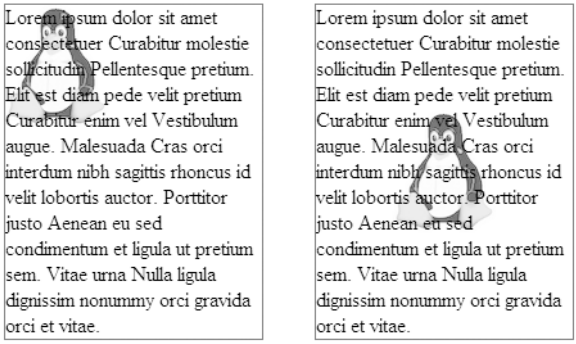


FIGURE 4–12 *Image d'arrière-plan dans sa position par défaut (coin supérieur gauche), puis placée au centre du bloc.*

Fixation de l'image d'arrière-plan

Lorsque l'internaute fait défiler une page contenant une image d'arrière-plan, celle-ci se déplace sur l'écran en même temps que le texte, car elle est attachée au coin supérieur gauche du bloc auquel elle est associée. Si cette image doit rester fixe lorsque le texte défile sur l'écran, il faut le signaler avec la propriété **`background-attachment`**.

TABLEAU 4–37 Propriété background-attachment

Propriété	**`background-attachment`**
Exemples	`body { background-attachment: scroll; }` `.pub { background-atachment: fixed; }`
Valeurs possibles	**`scroll`** : l'image défile avec le contenu (valeur par défaut) ; **`fixed`** : l'image reste fixe lors du défilement, seul défile le contenu qui est au premier plan.
Héritage	*Non.*

Raccourcis pour les arrière-plans

Comme pour les bordures, il existe une propriété raccourcie pour l'image d'arrière-plan et ses caractéristiques.

Tableau 4–38 Raccourci background

Propriété	`background`
Exemple	`h1 { background: blue url(logo.png) 50% repeat-x fixed; }`
Valeurs possibles	Valeurs de `background-color`, `background-image`, `background-repeat`, `background-attachment`, `background-position` dans un ordre quelconque.
Héritage	*Non*.

Listes à puces ou numérotées

Nous allons étudier à présent les propriétés associées aux listes. Elles permettent de leur appliquer des types de puce, des images en guise de puces ou des numéros.

Type de puce ou de numérotation

La propriété `list-style-type` indique quel type générique de puce ou quel mode de numérotation il faut utiliser pour la liste concernée.

Tableau 4–39 Propriété list-style-type

Propriété	`list-style-type`
Exemples	`ul { list-style-type: square; }` `ol { list-style-type: upper-roman; }`
Valeurs possibles	*Liste à puces* : `disc` (cercle plein - valeur par défaut), `circle` (cercle vide), `square` (carré plein).

TABLEAU 4–39 **Propriété list-style-type (suite)**

	Liste numérotée : **`decimal`** (1, 2…- valeur par défaut), **`decimal-leading-zero`** (01, 02…), **`lower-roman`** (i,ii…), **`upper-roman`** (I,II…), **`georgian`**, **`armenian`**, **`lower-latin`** = **`lower-alpha`** (a, b, c…), **`upper-latin`** = **`upper-alpha`** (A, B, C…), **`lower-greek`** (α, β,γ …). *Pas de puce ni de numéro* : **`none`**.
Héritage	Propriété *héritée*. Retour à la valeur initiale : `disc` ou `decimal`.

Utilisation d'une image comme puce

Grâce à la propriété **`list-style-image`**, n'importe quelle image peut être utilisée comme puce. Il faudra évidemment s'assurer que la taille, la forme et la couleur de cette image conviennent pour un tel usage.

TABLEAU 4–40 **Propriété list-style-image**

Propriété	**`list-style-image`**

Exemples	`ul { list-style-image:`
	`  url(image/puce.gif); }`
	`li { list-style-image:`
	`  url(http://www.top.org/logo.gif);}`

Valeurs possibles	**`url`** (nom d'image avec chemin relatif ou absolu) ou **`none`** : aucune image (valeur par défaut).
Héritage	Propriété *héritée*. Pour retrouver la valeur initiale, utiliser `none`.

À NOTER **Guillemets facultatifs**

Comme pour la propriété `background-image`, les guillemets ou apostrophes autour du nom de fichier image ou d'une URL sont *facultatifs*, en raison de la présence des parenthèses dans l'expression `url(...)`.

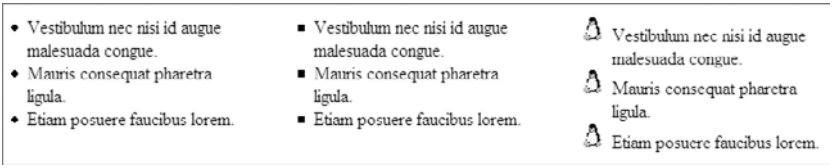


Figure 4–13 *Listes avec différentes puces : disque plein (par défaut), carré plein (*`list-style-type: square`*) et image (avec la propriété* `list-style-image`*)*

Position de la puce

La puce, l'image qui la remplace ou le numéro se trouveront, suivant la valeur de la propriété **`list-style-position`** et comme le montre la figure 4-14 :

- soit à gauche du paragraphe (c'est la configuration standard) ;
- soit à l'intérieur du paragraphe, avec dans ce cas un retrait de première ligne pour chaque item de la liste.

Tableau 4–41 Propriété list-style-position

Propriété	**`list-style-position`**
Exemples	`ul { list-style-position: outside; }` `ol { list-style-position: inside; }`
Valeurs possibles	**`outside`** : la puce est dans la marge (valeur par défaut) ; **`inside`** : la puce fait partie de la première ligne du paragraphe.
Héritage	Propriété *héritée*. Retour à la valeur initiale avec `outside`.

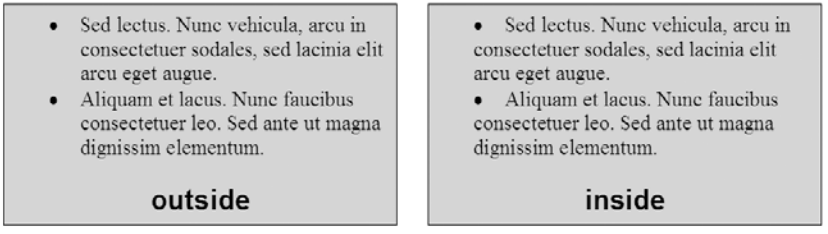


Figure 4–14 *Puces à l'intérieur ou à l'extérieur des paragraphes*

Raccourci pour toutes les propriétés de liste

L'ensemble des propriétés qui permettent de paramétrer les listes (type de puce ou de numérotation, fichier image remplaçant les puces, position des puces ou numéros) peut être défini à l'aide du raccourci **`list-style`**.

TABLEAU 4–42 **Propriété raccourcie list-style**

Propriété	**`list-style`**
Exemple	`li { list-style: circle inside; }`
Valeurs possibles	Toutes les valeurs (facultatives) de **`list-style-type`**, **`list-style-image`** et **`list-style-position`**.
Héritage	Cette propriété est *héritée*, comme chacune des propriétés dont elle est un raccourci.

Les tableaux

Les propriétés de style qui suivent permettent de préciser la mise en forme des tableaux. Rappelons à cette occasion qu'en XHTML, les tableaux ne sont pas utilisés pour la mise en page : nous aurons l'occasion d'en parler plus en détail dans le chapitre qui suit, à propos du positionnement des blocs de texte dans la page.

Largeur fixe ou variable des colonnes ou du tableau

Par défaut, les largeurs de colonne d'un tableau sont automatiques : elles s'adaptent à leur contenu. Pour obtenir des colonnes de largeur fixe, il faut utiliser la propriété **`table-layout`**.

TABLEAU 4–43 **Propriété table-layout**

Propriété	**`table-layout`**
Exemple	`table { table-layout: fixed; }`

Tableau 4–43 Propriété table-layout (suite)

Valeurs possibles	`auto` : largeur automatique (valeur par défaut) ou `fixed` : largeur fixe.
Héritage	Propriété *héritée*. Pour retrouver la valeur initiale, utiliser `auto`.

> **À noter** **Dimension du tableau avec `fixed`**
>
> Avec la valeur `fixed`, le tableau prend toute la largeur disponible (sur la page ou dans son conteneur), sauf si une dimension est précisépour ce tableau avec `width`.

Recouvrement des bordures

La propriété `border-collapse` sert à indiquer s'il y aura fusion ou non des bordures qui se touchent dans le tableau (voir la figure 4-15) :

- bordures contiguës de deux cellules voisines ;
- bordure du tableau et bordure d'une cellule qui se trouve en bord de tableau.

Tableau 4–44 Propriété border-collapse

Propriété	`border-collapse`
Exemple	`table, td { border: solid 1px red;` `border-collapse: collapse; }`
Valeurs possibles	`collapse` : fusion des bordures, `separate` : séparation des bordures (voir la figure 4-15).
Héritage	Propriété *héritée*.

> **Attention** **Valeur par défaut**
>
> La valeur par défaut de la propriété `border-collapse` est `collapse` en CSS 2, et `separate` en CSS 2.1. Il est donc préférable de toujours en spécifier explicitement la valeur, quelle qu'elle soit.

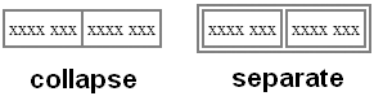


FIGURE 4–15 *Fusion ou séparation des bordures avec* `border-collapse`

Espacement entre les bordures de cellules

La taille de l'espace qui se trouve entre les bordures de deux cellules adjacentes est réglable à l'aide de la propriété **`border-spacing`**. Bien sûr, ceci ne vaut que pour les cellules dont les bordures sont distinctes, donc pour lesquelles la propriété `border-collapse` a pour valeur `separate`.

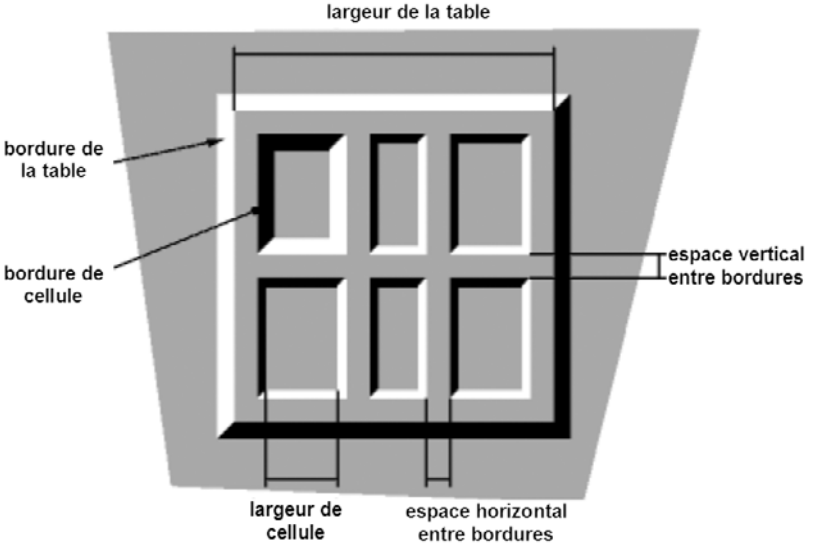


FIGURE 4–16 *Schéma d'un tableau d'après la traduction des normes CSS 2 du W3C : http://www.yoyodesign.org/doc/w3c/css2/tables.html#propdef-border-spacing*

> ATTENTION **Navigateurs récalcitrants**
>
> Cette propriété `border-spacing` n'est pas prise en compte par Internet Explorer 6 et 7.

Tableau 4–45 Propriété border-spacing

Propriété	`border-spacing`
Exemples	`table { border-collapse: separate; border-spacing: 5px; }` `table { border-collapse: separate; border-spacing: 2px 5px; }`
Valeurs possibles	Un ou deux **nombres positifs**, dans les mêmes unités que les tailles de police (px, em, ex) sauf %. **Une valeur** : espacement pour toutes les bordures. **Deux valeurs** : espacement *horizontal* et espacement *vertical*.
Héritage	Propriété *héritée*.

Contour des cellules vides

Afficher ou masquer le contour des cellules vides, voilà ce que va indiquer la propriété `empty-cells`, uniquement dans le cas où les bordures sont distinctes (`border-collapse: separate;`).

Tableau 4–46 Propriété empty-cells

Propriété	`empty-cells`
Exemple	`table { border-collapse: separate; empty-cells: show; }`
Valeurs possibles	`show` = afficher (valeur par défaut) ou `hide` = masquer.
Héritage	Propriété *héritée*. Pour retrouver la valeur initiale, utiliser `show`.

> **Attention** **Prise en charge par les navigateurs**
>
> Cette propriété `empty-cells` n'est pas prise en compte par Internet Explorer 6 et 7.

Position du titre du tableau

Placée à l'intérieur des balises `<table>...</table>`, la balise `<caption>` donne un titre au tableau, qui se trouve initialement au-dessus de celui-ci. Il est cependant possible de le placer en dessous, en donnant la valeur adéquate à la propriété **`caption-side`**.

TABLEAU 4–47 Propriété caption-side

Propriété	**`caption-side`**
Exemple	`caption { caption-side: bottom; }`
Valeurs possibles	**`top`** : titre au-dessus du tableau (valeur par défaut), **`bottom`** : titre sous le tableau.
Héritage	*Non.*

> ATTENTION **Navigateurs et normes**
>
> Cette propriété `caption-side` n'est pas prise en compte par Internet Explorer 6 et 7. Dans ces deux versions, le titre du tableau est toujours placé au-dessus, quelle que soit la valeur attribuée à `caption-side`.

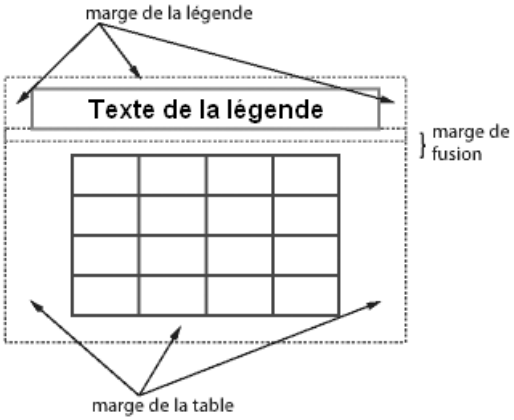


FIGURE 4–17 *Un tableau et sa légende, d'après les normes du W3C traduites : http://www.yoyodesign.org/doc/w3c/css2/tables.html#propdef-caption-side*

Alignement sur la virgule

Déjà rencontrée dans son utilisation pour les blocs de texte, la propriété **`text-align`** précise l'alignement horizontal du texte. Elle s'emploie dans une cellule de tableau de la même façon que dans un paragraphe.

Cependant, cette propriété appliquée à un tableau nous propose une autre possibilité, celle d'aligner le texte sur un caractère donné : par exemple, le choix de la valeur **`","`** pour ce caractère permet l'alignement de nombres sur leur virgule décimale.

Tableau 4–48 Propriété text-align utilisée dans les tableaux

Propriété	**`text-align`**
Exemples	`td.nombre { text-align: ","; }` `td.montant { text-align: "&euro;"; }`
Valeurs possibles	**Chaîne de caractères** : par exemple "," pour l'alignement sur la virgule, **`left`** : aligné à gauche (valeur par défaut), **`right`** : aligné à droite, **`center`** : centré, **`justify`** : justifié.
Héritage	Propriété *héritée*. Pour retrouver la valeur initiale, utiliser `left`.

Alignement vertical des cellules

La propriété **`vertical-align`** permet d'indiquer quel doit être l'alignement d'une cellule par rapport à la rangée de cellules dont elle fait partie, lorsque la taille de cette rangée est supérieure à celle de la cellule.

Tableau 4–49 Propriété vertical-align

Propriété	**`vertical-align`**
Exemples	`.commentaire { vertical-align: top; }` `.titre { vertical-align: middle; }`
Valeurs possibles	**`baseline`** : alignement normal sur la première ligne (valeur par défaut), **`top`** : alignement sur le haut de la rangée de cellules, **`middle`** : alignement au milieu de la rangée, **`bottom`** : alignement sur le bas de la rangée.

TABLEAU 4–49 Propriété vertical-align (suite)

Héritage	Propriété *héritée*. Pour retrouver la valeur initiale, utiliser `baseline`.

La ligne de base d'une rangée de cellules (sur laquelle se « poseront » les lettres de la première ligne) est la même pour toutes les cellules d'une même ligne. Dans la figure 4-18, la première ligne de la boîte de la cellule 2 est la plus haute des premières lignes de cellules, donc c'est elle qui détermine la « ligne de base » de la rangée.

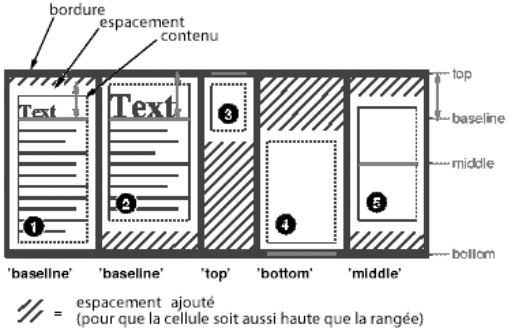


FIGURE 4–18 *Alignement vertical de cellules dans une rangée, d'après la page : http://www.yoyodesign.org/doc/w3c/css2/tables.html#height-layout (le site http://www.yoyodesign.org traduit une partie du site du W3C : http://www.w3.org)*

Nous voilà maintenant en possession des propriétés CSS qui permettent la mise en forme de nos pages web. Cependant, nous ne les avons pas encore toutes détaillées : il nous reste à étudier les propriétés liées à la position des blocs, ainsi qu'à leurs dimensions et à leurs marges. Ce sujet étant suffisamment vaste à lui tout seul, il fera l'objet du chapitre suivant...

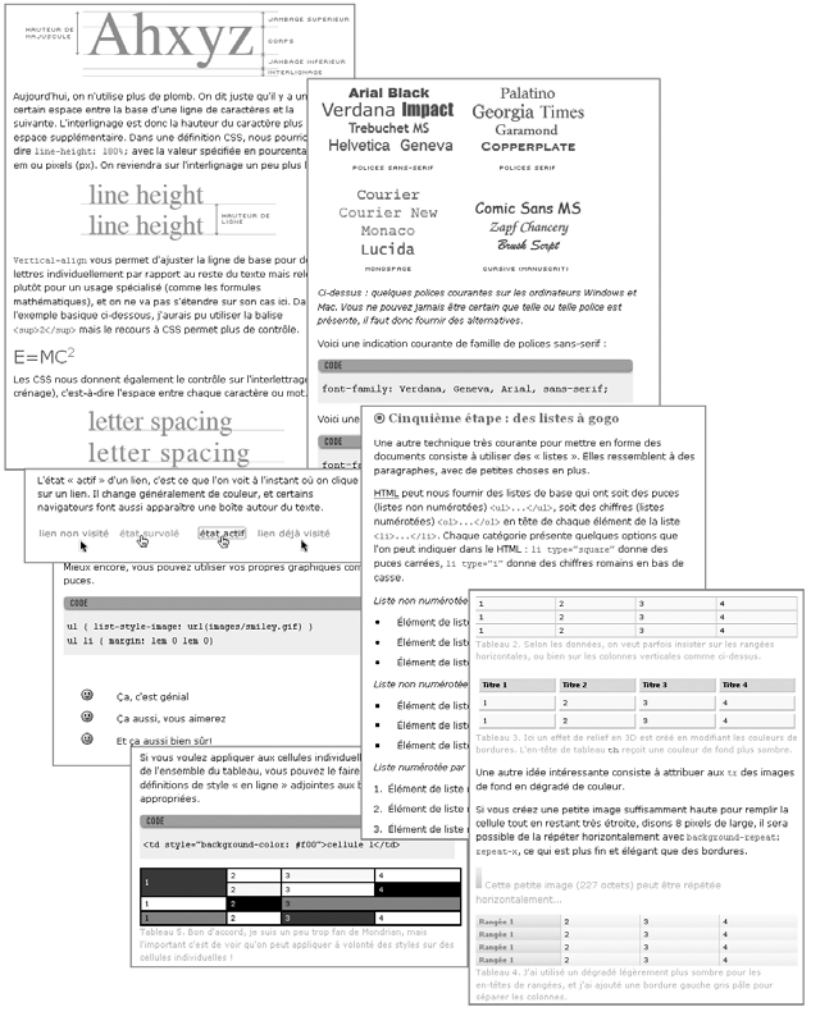


Figure 4–19 *Exemples de mise en forme (texte, liens, liste à puces, tableaux, ...) extraits du site http://www.pompage.net, à la rubrique : « CSS, on reprend tout à zéro ! », traduction d'articles de Joe Gillespie publiés sur http://www.wpdfd.com.*

chapitre

5


Une demonstration de ce qu'on peut accomplir lorsqu'on utilise les CSS pour la conception web. Sélectionnez n'importe quelle feuille de style listée pour charger le résultat sur cette page.
CSS Zen Garden
The Beauty of CSS Design
Téléchargez les fichiers d'exemple html et css
Select a design
Love Is In The Air
by Nele Goetz
Greece Remembrance
by Pierre-Leo Bourbonnais
Hengarden
by Mr. Khmerang
Hoops - Tournament Edition
by David Marshall Jr.
Obsequience
by Pierce Gleeson
Red Paper
by Rob Soule
Chien
by Alex Miller
The Road to Enlightenment
Les reliques passées des sélecteurs spécifique aux navigateurs, des DOMs incompatibles, et du manque de support des CSS encombrent un long chemin sombre et morne.
Aujourd'hui, nous devons nous clarifier l'esprit et nous débarassez des pratiques passées. La révélation de la véritable nature du Web est maintenant possible, grâce aux efforts infatigables des gens du W3C, du WaSP et des créateurs des principaux navigateurs.
Le Jardin Zen css vous invite à vous relaxer et à méditer sur les leçons importantes des maîtres. Commencez à voir clairement. Apprenez à utiliser ces techniques (bientôt consacrées par l'usage) de manière neuve et revigorante. Ne faites qu'Un avec le Web.
So What is This About?

Positionnement des blocs

En CSS, la mise en page consiste à placer les blocs de texte et les images comme des boîtes imbriquées, juxtaposées ou superposées dans la page.

Sommaire

Fini, l'embrouillamini des tableaux imbriqués dans d'autres tableaux ! Les feuilles de style nous proposent une méthode claire pour la mise en page : elle consiste à positionner les éléments dans la page ou dans leur bloc conteneur.

Toutefois, il faudra au préalable définir les dimensions et les marges, intérieures et extérieures, de chaque élément.

Marges et dimensions d'un bloc

Il est important de comprendre comment sont calculées les dimensions des « boîtes », c'est-à-dire des blocs contenant texte et images.

Les dimensions d'une boîte peuvent être fixées, ainsi que ses marges intérieures (à l'intérieur des bordures) et ses marges extérieures (à l'extérieur des bordures).

Marges externes autour d'un bloc

Les marges externes d'un bloc sont situées au-delà de ses bordures. Elles servent à espacer les blocs entre eux. Elles sont définies sur chacun des côtés à l'aide des propriétés `margin-top` en haut, `margin-right` à droite, `margin-bottom` en bas et `margin-left` à gauche, ou globalement par la propriété raccourcie `margin`.

TABLEAU 5–1 Propriétés définissant les marges extérieures

Propriétés	**`margin-left, margin-right, margin-top, margin-bottom`**
Exemples	`p { margin-left: 4em; margin-right: 3em; }` `p { margin-top: 5px; margin-bottom: 6px; }`
Valeurs possibles	**`auto`**, **taille relative** (conseillée) en **em**, **ex**, **%**, **px** ou **taille fixe** en **pt**, **pc**, **cm**, **mm**, **in**.
Pourcentages	% de la **largeur** du bloc conteneur, même pour `margin-top` et `margin-bottom`.
Héritage	*Non.*

À noter **Utilisation des marges externes**

- Pour les blocs juxtaposés ou imbriqués, les marges mitoyennes sont confondues : par exemple, la marge inférieure d'un bloc et la marge supérieure du bloc suivant sont fusionnées. Cependant, ce n'est pas le cas pour les blocs flottants ou positionnés.
- La valeur `auto` s'emploie avec `margin-left` et `margin-right` : elle signifie que ces deux marges doivent être égales, ce qui revient à centrer l'élément concerné dans son bloc conteneur.
- En utilisant des valeurs négatives pour les marges externes, il est possible de superposer des blocs.
- Les éléments en ligne n'ont pas de marge supérieure, ni inférieure ; pour les marges de gauche et de droite, la valeur `auto` correspond à 0.

Raccourci pour les marges externes

La propriété **`margin`** simplifie la définition des marges externes, en remplaçant les quatre propriétés précédentes.

Tableau 5–2 **Propriété raccourcie margin**

Propriété	**`margin`**

Exemples	`p { margin: 0; }`
	`p { margin: 3em 5em; }`
	`p { margin: 5% 10% 8%; }`
	`p { margin: 15px 10px 20px 15px; }`

Valeurs possibles	**Une valeur** : définit toutes les marges extérieures. **Deux valeurs** : ① marges du haut et du bas égales, ② marges de gauche et de droite égales **Trois valeurs** : ① marge du haut, ② marges de gauche et de droite égales, ③ marge du bas **Quatre valeurs** : ① marge du haut, ② marge de droite, ③ marge du bas, ④ marge de gauche.
Pourcentages	% de la **largeur** du bloc conteneur, même pour les marges du haut et du bas.
Héritage	*Non.*

ASTUCE **Ordre des propriétés**

Pour les marges extérieures ou intérieures comme pour les bordures, il est facile de se rappeler l'ordre dans lequel elles sont définies, car il suffit de commencer par le haut et de tourner dans le sens des aiguilles d'une montre : haut - droite - bas - gauche.

Marges internes d'un bloc

Les marges intérieures d'un bloc se trouvent à l'intérieur de ses bordures. Leur présence évite que le texte ne soit collé au cadre du bloc qui le contient. Elles sont définies soit sur chaque côté à l'aide des propriétés `padding-top` en haut, `padding-right` à droite, `padding-bottom` en bas et `padding-left` à gauche, soit globalement avec la propriété raccourcie `padding`.

TABLEAU 5–3 **Propriétés définissant les marges intérieures**

Propriétés	**`padding-left, padding-right, padding-top, padding-bottom`**
Exemples	`p { padding-left: 20px; padding-right: 15px; }` `p { padding-top: 5%; padding-bottom: 10%; }`
Valeurs possibles	**Valeur de taille relative** (conseillée) en **em**, **ex**, **%**, **px** ou de **taille fixe** en **pt**, **pc**, **cm**, **mm**, **in**.
Pourcentages	% de la **largeur** du bloc conteneur, même pour `padding-top` et `padding-bottom`.
Héritage	*Non.*

À NOTER **Utilisation des marges internes**

- La valeur par défaut des marges internes est 0.
- Les marges internes ne peuvent pas être négatives.
- La valeur `auto` n'existe pas pour les marges intérieures.
- Les éléments en ligne n'ont ni marge supérieure, ni marge inférieure.

Raccourci pour les marges internes

La propriété `padding` simplifie la définition des marges internes, en remplaçant les quatre propriétés précédentes.

Tableau 5–4 Propriété raccourcie padding

Propriété	`padding`

Exemples	`p { padding: 5ex; }`
	`p { padding: 10px 0; }`
	`p { padding: 2em 1em 3em; }`
	`p { margin: 5% 8% 6% 10%; }`

Valeurs possibles	**Une valeur** : définit toutes les marges intérieures. **Deux valeurs** : ① marges du haut et du bas égales, ② marges de gauche et de droite égales **Trois valeurs** : ① marge du haut, ② marges de gauche et de droite égales, ③ marge du bas **Quatre valeurs** : ① marge du haut, ② marge de droite, ③ marge du bas, ④ marge de gauche.
Pourcentages	% de la **largeur** du bloc conteneur, même pour les marges du haut et du bas.
Héritage	*Non.*

Largeur fixe pour un bloc ou une image

Il est possible de choisir une largeur fixe pour un bloc de texte ou une image, en utilisant la propriété `width`.

Tableau 5–5 Propriété width

Propriété	`width`
Exemples	`div { width: 300px; }` `.menu { width: 20%; }`
Valeurs possibles	`auto` (par défaut), **taille relative** (conseillée) en **em**, **ex**, **%**, **px** ou **taille fixe** en **pt**, **pc**, **cm**, **mm**, **in**.
Pourcentages	% de la **largeur** du bloc conteneur.
Héritage	*Non.*

> ATTENTION **Utilisation de `width`**
>
> - Les valeurs de `width` ne peuvent pas être négatives.
> - Internet Explorer 6 interprète `width` comme une largeur minimum, et non comme une largeur fixe. En revanche, Firefox et les versions plus récentes d'Internet Explorer respectent la norme sur ce point.

Hauteur fixe pour un bloc ou une image

La propriété **`height`** permet de définir une hauteur fixe pour un bloc de texte ou une image.

TABLEAU 5–6 **Propriété height**

Propriété	**`height`**
Exemples	`div { height: 50%; }` `img#logo { height: 10em; }`
Valeurs possibles	**`auto`** (par défaut), **taille relative** (conseillée) en **em**, **ex**, **%**, **px** ou **taille fixe** en **pt**, **pc**, **cm**, **mm**, **in**.
Pourcentages	% de la **hauteur** du bloc conteneur si celle-ci est fixée, sinon c'est la valeur `auto` qui est appliquée.
Héritage	*Non*.

> ATTENTION **Utilisation de `height`**
>
> - Les valeurs de `height` ne peuvent pas être négatives.
> - Internet Explorer 6 interprète `height` comme une hauteur minimum, et non comme une hauteur fixe. En revanche, Firefox et Internet Explorer à partir de sa version 7 interprètent correctement cette propriété.

Largeur et hauteur totales d'un bloc

Le modèle de boîte de la figure 5-1 montre comment calculer la largeur et la hauteur totales d'un bloc.

Pour obtenir la **largeur totale** d'un bloc, il faut additionner :

- les marges extérieures de gauche et de droite : `margin-left` + `margin-right` ;
- deux fois l'épaisseur de la bordure : 2 × `border-width` ;
- les marges intérieures de gauche et de droite : `padding-left` + `padding-right` ;
- la largeur du contenu : `width`.

Pour obtenir la **hauteur totale** d'un bloc, il faut additionner :

- les marges extérieures du haut et du bas : `margin-top` + `margin-bottom` ;
- deux fois l'épaisseur de la bordure : 2 × `border-width` ;
- les marges intérieures du haut et du bas : `padding-top` + `padding-bottom` ;
- la hauteur du contenu : `height`.

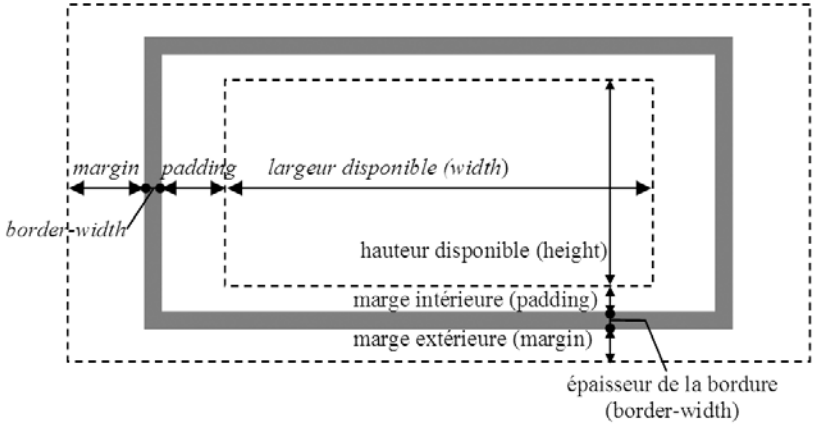


Figure 5–1 *Dimensions des boîtes*

Lorsque les marges horizontales sont égales, le calcul de la largeur totale est plus simple :

```
Largeur totale = 2 × (margin + border-width
                      + padding) + width.
```

Il en est de même pour le calcul de la hauteur totale, lorsque les marges verticales sont égales :

```
Hauteur totale = 2 × (margin + border-width
                      + padding) + height.
```

> À NOTER **Prise en compte des bordures**
>
> Les calculs qui précèdent supposent que le bloc concerné est encadré par une bordure uniforme, avec la même épaisseur d'encadrement à gauche et à droite pour le calcul de la largeur, ou bien en haut et en bas pour celui de la hauteur. Dans le cas contraire :
> - si le bloc ne comprend pas de bordure, il suffit de soustraire `border-width`, cela simplifie la formule ;
> - si les bordures ne sont pas uniformes, il faut ajouter séparément l'épaisseur de chacune des bordures concernées.

Problèmes de marges avec Internet Explorer 6

Tant que la version 6 d'Internet Explorer (IE 6 pour les habitués) reste utilisée pour une part qui n'est pas négligeable, nous devrons tenir compte du fait que ce navigateur ne respecte pas certaines normes du W3C.

En particulier pour les dimensions des boîtes, il inclut les marges intérieures dans `width` et `height` :

```
Largeur totale pour IE6 = margin-left + width +
                          margin-right
Hauteur totale pour IE6 = margin-top + height +
                          margin-bottom
```

Pour résoudre cette question et obtenir un rendu identique de la page web quel que soit le navigateur, il existe une méthode de contournement : elle consiste à remplacer les marges intérieures d'un bloc par des marges extérieures appliquées à tous les éléments qu'il contient.

Exemple de code HTML à mettre en forme :

```
<div id="fretin">
  <p>Poissons de mer</p>
  <p>Poissons d'eau douce</p>
</div>
```

Le style initial comprend les propriétés suivantes :

```
* { margin: 0; }
div#fretin {
        border: solid 1px;
        width: 180px;
        padding-left: 30px;
}
```

et donne des blocs de largeurs différentes sur Firefox et Internet Explorer 6, comme le montre la figure 5-2.

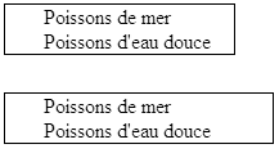


Figure 5–2 *Largeurs de bloc différentes sur Internet Explorer 6 (en haut) et Firefox (en bas)*

Pour résoudre le problème, il suffit de modifier la feuille de style :

- suppression de la marge interne `padding-left` du bloc `<div>` et augmentation de sa largeur `width` de 30 pixels pour conserver les mêmes dimensions ;
- ajout d'une marge externe `margin-left` de 30 pixels pour les paragraphes `<p>` inclus dans le bloc `<div>`.

La nouvelle feuille de style comprend donc les propriétés suivantes :

```
* { margin: 0; }
div#fretin {
        border: solid 1px;
        width: 210px;
}
div#fretin p { margin-left: 30px; }
```

et donne des blocs de même largeur avec Firefox et Internet Explorer 6 (voir la figure 5-3).

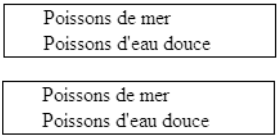


FIGURE 5–3 *Affichage identique sur Internet Explorer 6 (en haut) et sur Firefox (en bas)*

> **À NOTER Marges extérieures initialisées à zéro**
>
> Une feuille de style commence souvent par la règle `* { margin: 0; }`, qui est utilisée dans cet exemple. Cette règle de style met à zéro les marges externes de tous les éléments de la page web. En effet, la valeur par défaut `auto` pour les marges externes est interprétée de différentes façons par les navigateurs. La mise à zéro de ces marges résout donc le problème.

Largeur ou hauteur minimum

Les propriétés `min-width` et `min-height` définissent respectivement la largeur et la hauteur minimum d'un bloc.

TABLEAU 5–7 Propriétés min-width et min-height

Propriétés	**`min-width`**, **`min-height`**
Exemples	`h1 { min-width: 50%; }` `div.remarque { min-height: 50em; }`
Valeurs possibles	**none** = 0 (valeur par défaut), **taille relative** (conseillée) en **em**, **ex**, **%**, **px**, **taille fixe** en **pt**, **pc**, **cm**, **mm**, **in**.
Pourcentages	% de la **largeur** du bloc conteneur pour `min-width`, % de la **hauteur** du bloc conteneur pour `min-height`.
Héritage	*Non.*

À NOTER
Valeurs numériques pour `min-width` et `min-height`
Les valeurs numériques attribuées aux propriétés `min-width` et `min-height` doivent être positives.

ATTENTION **Navigateur récalcitrant**
Internet Explorer 6 ne reconnaît pas les propriétés `min-width` et `min-height`, nous verrons plus loin comment contourner ce problème, dans le chapitre consacré à l'adaptation des pages aux différents navigateurs. En revanche, Firefox et Internet Explorer à partir de sa version 7 interprètent ces propriétés.

Largeur ou hauteur maximum

Les propriétés `max-width` et `max-height` limitent respectivement la largeur et la hauteur d'un bloc.

À NOTER
Valeurs numériques pour `max-width` et `max-height`
Les valeurs numériques attribuées aux propriétés `max-width` et `max-height` doivent être positives.

TABLEAU 5–8 **Propriétés max-width et max-height**

Propriétés	**`max-width`**, **`max-height`**
Exemples	`p { max-width: 80%; }` `#extraits { max-height: 200px; }`
Valeurs possibles	**none** : pas de limite (valeur par défaut), **taille relative** (conseillée) en **em**, **ex**, **%**, **px**, **taille fixe** en **pt**, **pc**, **cm**, **mm**, **in**.
Pourcentages	% de la **largeur** du bloc conteneur pour `max-width`, % de la **hauteur** du bloc conteneur pour `max-height`.
Héritage	*Non.*

> ATTENTION **Navigateur récalcitrant**
>
> Internet Explorer 6 ne reconnaît pas les propriétés `max-width` et `max-height`. Il existe des solutions de remplacement, mais uniquement en Javascript. Firefox et Internet Explorer à partir de sa version 7 interprètent correctement ces propriétés.

Position des éléments

Chaque bloc peut être placé de différentes façons à l'intérieur de la page web : par rapport à d'autres blocs, ou bien à un endroit précis du bloc qui le contient, ou encore à un emplacement fixe sur la page.

Flux normal des éléments

À l'intérieur de chaque bloc, les éléments se placent au fur et à mesure, suivant le flux normal :

- les uns à la suite des autres pour les éléments en ligne : `<strong>`, `<em>`, `<span>`, `<img>`, ...
- les uns en dessous des autres pour les éléments de type bloc : `<p>`, `<div>`, `<h2>`, ...

Dans le flux normal, les dimensions d'un bloc sont les suivantes :

- **Largeur par défaut** = largeur disponible dans le bloc conteneur ;
- **Hauteur par défaut** = celle du contenu, 0 si le bloc ne contient rien.

Les blocs qui se succèdent dans le flux normal sont séparés entre eux par leurs marges extérieures (ces marges étant fusionnées entre deux blocs).

Principe du positionnement des blocs

Seuls peuvent être positionnés les blocs de texte et les éléments « remplacés », comme les images. Pour positionner des éléments en ligne, il faut les transformer en blocs à l'aide de la propriété `display: block;`

Par positionnement, les blocs peuvent être :

- juxtaposés,
- fixés par rapport à la position de leur bloc conteneur,
- ou encore fixés par rapport à la page.

Le positionnement nous permet de superposer les blocs, comme des *calques*. D'ailleurs, ce nom de « calque » est parfois donné à la balise `<div>`, qui est souvent utilisée comme bloc conteneur.

Si les blocs qui sont au premier plan ont un fond transparent (valeur par défaut de la couleur de fond), les blocs situés en dessous restent visibles.

Il est possible de choisir la position des blocs et de modifier leur ordre vertical de superposition, à l'aide de la propriété `z-index` que nous étudierons plus loin.

Types de position possibles

Un bloc peut être positionné de façon **normale**, **relative**, **absolue**, **fixe** ou **flottante**. Nous allons préciser ici à quoi correspondent ces différents types de positionnement.

Position normale

Lorsque sa position n'est pas précisée, un bloc se place dans le *flux normal* de la page web.

Position relative, absolue ou fixe

Il est possible de placer un élément en indiquant un *décalage* (en haut, en bas, à gauche, à droite) :

- par rapport à sa position dans le flux normal : c'est la « position relative » (propriété `position: relative;`);
- par rapport au bloc conteneur : c'est la « position absolue » (propriété `position: absolute;`);
- par rapport à l'écran : c'est la « position fixe » (propriété `position: fixed;`).

Dans chacun de ces trois cas, il faut indiquer un ou deux décalage(s) :

- un premier décalage à partir du haut (par exemple : `top: 2px;`) ou du bas (par exemple : `bottom: 10%;`);
- un deuxième à partir de la gauche (par exemple : `left: 5em;`) ou de la droite (par exemple : `right: 10ex;`).

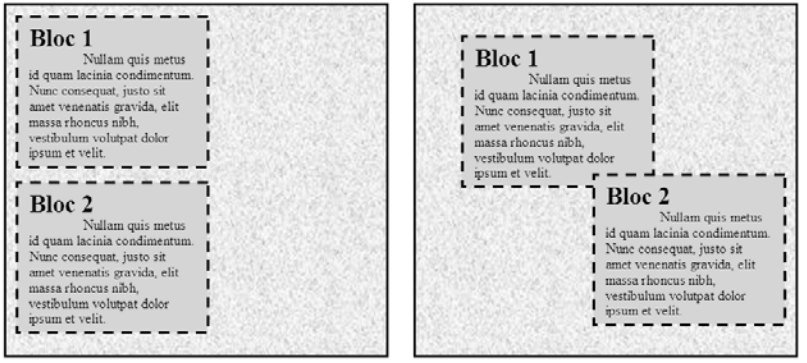


FIGURE 5–4 *À gauche, les blocs 1 et 2 sont dans le flux normal de la page ; à droite, ces deux blocs sont placés en positionnement absolu à l'intérieur de la page.*

Position flottante

Un élément peut enfin être déclaré « flottant » à gauche ou à droite, avec la propriété **`float`** qui s'écrit alors respectivement `float: left;` ou `float: right;`

Le bloc est placé le plus à gauche ou le plus à droite possible, tout en gardant sa position verticale dans la boîte de son conteneur.

Le contenu qui suit encadre alors cette boîte flottante, comme le montre la figure 5-5. S'il y a plusieurs éléments flottants, ils s'alignent côte à côte, avec retour à la ligne automatique lorsqu'est atteint le bord du bloc conteneur.

Utilisation des différents types de positionnement

Voici en résumé la façon d'utiliser les différents positionnements disponibles en CSS.

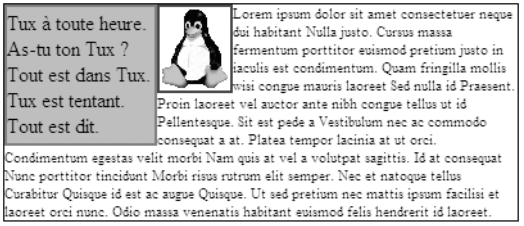


Figure 5–5 *Le bloc de texte sur fond gris et l'image sont flottants à gauche. Le texte qui suit habille ces deux blocs.*

Important

Hauteur d'un bloc qui contient des éléments flottants

La dimension d'un bloc conteneur ne prend pas en compte celle des éléments flottants qu'il contient : les blocs flottants débordent de leur conteneur.

Le bloc qui suivra risque donc de se superposer aux éléments flottants ou de se trouver à côté d'eux, comme le montre la figure 5-6. Pour éviter cela, il suffit d'attribuer au deuxième bloc la propriété `clear: both;` qui interdit la présence d'éléments flottants sur le côté (nous étudierons cette propriété plus loin).

Figure 5–6 *Les blocs flottants peuvent déborder de leur conteneur.*

Élément dans le flux (position normale)

- Soit aucune propriété de positionnement, soit **`position: static;`**
- Ce type de positionnement est à utiliser aussi souvent que possible.
- Les blocs sont affichés les uns sous les autres, les éléments en ligne sont placés côte à côte.

Position relative

- Propriété `position: relative;`
- Pour décaler ou superposer un élément par rapport à ses « frères ».

Position absolue

- Propriété `position: absolute;`
- Pour découper la page en zones, sans utiliser de tableaux.
- Pour disposer à un endroit précis un menu, un encadré, une image...

Position fixe

- Propriété `position: fixed;`
- Pour conserver en permanence un élément à l'écran (il ne bouge pas lors du défilement).
- Pour éviter l'utilisation de cadres (balise `<frame>`).

> ATTENTION **Les blocs positionnés en absolu, en fixe ou en flottant sortent du flux**
>
> Mis à part les éléments positionnés en relatif, pour lesquels il n'y a qu'un décalage par rapport à leur position initiale, tous les éléments positionnés d'une autre manière – donc en absolu, en fixe ou en flottant – « sortent du flux ».
> Cela entraîne ce que nous avons noté précédemment dans le cas des blocs flottants : si un bloc ne contient que des éléments positionnés, sa hauteur est nulle, donc la suite de la page remonte comme si le bloc précédent n'existait pas.
> La solution consiste à préciser une hauteur pour le bloc conteneur, ce qui nécessite de connaître la hauteur du ou des blocs qu'il contient et qui seront positionnés.
> Lors de la conception de nos pages, gardons ce point à l'esprit pour éviter de voir, après le positionnement de nos blocs, des éléments se superposer ou d'autres disparaître !

Élément flottant

- Propriété `float: left;` ou `float: right;`
- Pour placer des éléments côte à côte, en fonction de la place disponible dans la fenêtre d'affichage ou le bloc conteneur.
- Pour habiller une image par du texte, pour une galerie d'images ou une suite de menus...

> Important **Remarques pour la position absolue**
>
> Un bloc positionné se place par rapport au bloc qui le contient (son conteneur), mais **seulement si celui-ci est lui-même positionné**, que ce soit en position relative ou absolue.
> Si ce n'est pas le cas (conteneur dans le flux ou flottant), le bloc à positionner remonte de parent en parent jusqu'au premier bloc positionné (jusqu'à `<body>` s'il n'y en a pas) et se place par rapport à lui.
> Pour placer en position absolue un bloc dans un autre, il est donc important de vérifier si le conteneur est lui-même positionné. Si ce n'est pas le cas, il faut donner à ce bloc conteneur une position relative avec un décalage nul, ce qui ne modifiera pas sa position.

Type de positionnement d'un bloc

Au cours des exemples précédents, nous avons rencontré plusieurs fois la propriété `position` qui permet, comme son nom l'indique, de définir un type de position pour l'élément concerné.

Tableau 5–9 **Propriété position**

Propriété	`position`
Exemples	`p.note {position: relative; left: -5px;}` `#menu {position: absolute; top: 0; right: 10%;}`
Valeurs possibles	`static` : positionnement dans le flux normal (valeur par défaut). `relative` : décalage par rapport à la position dans le flux. `absolute` : positionnement par rapport au bloc conteneur. `fixed` : positionnement par rapport à l'écran.
Héritage	*Non.*

Décalages indiquant la position d'un bloc

Le positionnement utilise les décalages `top` (haut), `bottom` (bas), `left` (gauche) et `right` (droite).

> À NOTER **Décalages `top`, `bottom`, `left` et `right`**
>
> - Les valeurs négatives sont possibles pour `top`, `bottom`, `left` et `right`.
> - Si `top` et `bottom` sont spécifiés simultanément, c'est `top` qui est pris en compte.
> - Si `left` et `right` sont spécifiés simultanément, c'est `left` qui est pris en compte.

TABLEAU 5–10 **Propriétés top, bottom, left et right**

Propriété	`top, bottom, left, right`

Exemples	`p.note { position: relative;`
	`top: 5px; left: 10px;}`
	`div.menu { position: absolute;`
	`top: 30%; right: 20%;}`
	`#remarque {position: relative; top: 2em;}`

Valeurs possibles	**none** : pas de décalage (valeur par défaut), **valeur relative** (conseillée) en **em**, **ex**, **%**, **px**, **valeur fixe** en **pt**, **pc**, **cm**, **mm**, **in**.
Héritage	*Non.*

Niveau d'empilement des blocs

Lorsque plusieurs blocs sont superposés, ils s'empilent suivant l'ordre de leur arrivée dans le code XHTML. Cet arrangement peut toutefois être modifié, grâce à la propriété `z-index`.

TABLEAU 5–11 **Propriété z-index**

Propriété	`z-index`
Exemples	`ul.menu { position: relative; z-index: 10; }` `#logo {position: absolute; top: 0; z-index: -5;}`

Tableau 5–11 **Propriété z-index (suite)**

Valeurs possibles	`auto` : même niveau d'empilement que la boîte parent (valeur par défaut) ou **nombre entier** positif, nul ou négatif.
Héritage	*Non.*

Important **Utilisation de la propriété `z-index`**

- La propriété `z-index` n'est prise en compte que lorsqu'elle s'applique à un bloc qui est positionné, bien que certaines versions de certains navigateurs l'acceptent sans autre formalité. Si ce n'est pas le cas, le bloc concerné peut être positionné tout en conservant son emplacement : il suffit de lui donner une position relative, sans décalage.
- Plus la valeur de `z-index` est élevée, plus le bloc se trouve en haut dans la superposition des blocs.
- La transparence du fond (valeur par défaut de `background-color`) permet de voir le contenu des boîtes situées plus bas dans la superposition.

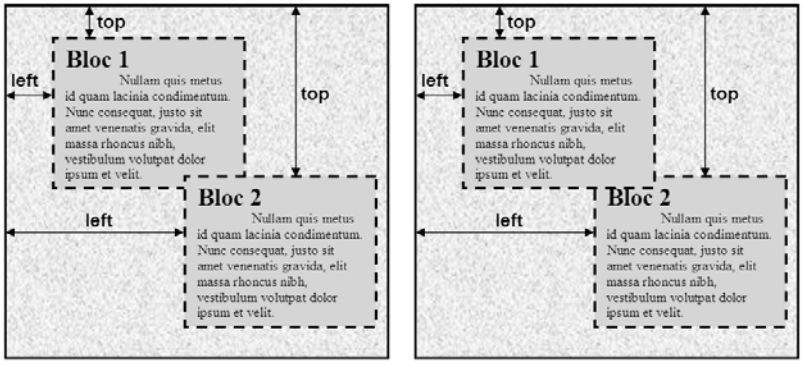


Figure 5–7 *Les blocs 1 et 2 sont positionnés ici par rapport aux limites haute et gauche de leur bloc conteneur ; à droite, le bloc 1 a un niveau d'empilement* `z-index` *plus élevé que le bloc 2.*

Transformation en bloc flottant

La propriété `position` ne permet pas d'effectuer le positionnement flottant d'un bloc, c'est la propriété spécifique **`float`** qui est utilisée pour cela.

TABLEAU 5–12 **Propriété float**

Propriété	**`float`**
Exemples	`img.vignettes { float: left; }` `div.infos { float: right; }`
Valeurs possibles	**`none`** : pas de flottement (valeur par défaut), **`left`** : élément flottant calé à gauche, **`right`** : élément flottant calé à droite.
Héritage	*Non.*

Pas d'éléments flottants sur le côté

S'il reste de la place, le contenu qui suit un élément flottant dans le code XHTML va se trouver à côté de celui-ci : c'est ainsi qu'une image flottante peut être « habillée » par du texte.

Pour obliger le navigateur à afficher un élément à la ligne, sous le plus bas des blocs flottants, il faut le lui préciser en attribuant à cet élément la propriété **`clear`**.

TABLEAU 5–13 **Propriété clear**

Propriété	**`clear`**
Exemples	`h1 { clear: both; }` `.remarque { clear: left; }`
Valeurs possibles	**`none`** : boîtes flottantes autorisées à gauche et à droite (valeur par défaut), **`left`** : pas d'élément flottant sur la gauche, **`right`** : pas d'élément flottant sur la droite. **`both`** : aucun élément flottant, ni à gauche, ni à droite.
Héritage	*Non.*

Affichage ou non d'un élément

La propriété `visibility` permet de masquer occasionnellement un élément tout en réservant dans la page la place qu'il occuperait normalement, donc sans perturber la mise en page.

Tableau 5–14 Propriété visibility

Propriété	`visibility`
Exemple	`.retrait { visibility: hidden; }`
Valeurs possibles	`visible` : l'élément est visible (valeur par défaut), `hidden` : l'élément est masqué, mais il occupe toujours le même espace dans la page, `collapse` = `hidden`, sauf dans les tableaux, où l'espace est libéré lorsqu'il s'agit d'une ligne entière ou d'une colonne entière.
Héritage	*Non.*

Affichage des débordements

L'affichage des éléments qui débordent de leur bloc conteneur sera fonction de la propriété `overflow` associée à ce bloc : ces éléments pourront être visibles, masqués ou accessibles grâce à une barre de défilement.

Tableau 5–15 Propriété overflow

Propriété	`overflow`
Exemples	`p { overflow: scroll; }` `#cadre1 { overflow: hidden; }`
Valeurs possibles	`visible` : le débordement est visible (valeur par défaut), `hidden` : le débordement est masqué, `scroll` : affichage dans tous les cas d'une barre de défilement, qui permettra l'accès à un débordement éventuel, `auto` : une barre de défilement apparaît, mais seulement en cas de débordement.
Héritage	*Non.*

Zone visible d'une boîte

Appliquée à un bloc, la propriété `clip` permet d'y définir un rectangle qui en sera la zone visible. Ce rectangle sera déterminé grâce à des valeurs de décalage à partir des bords du bloc. Il pourra être plus grand que le bloc si ces valeurs sont négatives (débordement) et constituera un « rognage » si ces décalages sont des nombres positifs.

TABLEAU 5–16 **Propriété clip**

Propriété	`clip`
Exemples	`p { clip: rect(10px 20px 5px 15px); }` `p#cadre2 { clip: rect(0 2em 2em 5em); }`
Valeurs possibles	**`auto`** : la zone visible est la boîte entière (valeur par défaut) ou **`rect(top, right, bottom, left)`** : la zone visible est un *rectangle* défini par les quatre valeurs entre parenthèses. Ces valeurs indiquent des décalages par rapport aux côtés respectifs de la boîte (valeurs possibles pour chacun d'eux : `auto` qui équivaut à 0, valeur positive, valeur négative).
Héritage	*Non*.

À NOTER **Rectangle visible défini avec la propriété `clip`**

Lorsque la propriété `clip: rect(top, right, bottom, left);` définit un rectangle visible, le contenu qui se trouve en dehors de la zone définie est considéré comme un débordement. Il est donc traité suivant la valeur de la propriété `overflow` : par défaut, le débordement est visible.

Dans l'expression **`rect(top, right, bottom, left);`**, les décalages `top`, `right`, `bottom` et `left` peuvent être exprimés en taille relative (ce qui est conseillé) : em, ex, px, ou bien en taille fixe : pt, pc, cm, mm, in.

Changement de type d'élément

Chaque élément de la page fait partie d'une catégorie, comme les éléments en ligne, les blocs, etc. Cependant, il est parfois nécessaire de changer le type d'un élément.

Par exemple, pour appliquer à un élément en ligne (par exemple, un lien) une propriété liée aux blocs, il faut d'abord le transformer en bloc. C'est ce que permet la propriété **`display`**.

Tableau 5–17 Propriété display

Propriété	**`display`**

Exemples	`p.secret { display: none; }`
	`.cellule { display: table-cell; }`
	`span.bloc { display: block; }`

Valeurs possibles	**`inline`** : élément en ligne (valeur par défaut) **`block`** : bloc **`list-item`** : élément de liste **`inline-block`** : élément en ligne remplacé **`run-in`** : bloc ou élément en ligne, suivant le contexte **`table`** : tableau, **`inline-table`** : tableau en ligne **`table-cell`** : cellule de tableau, **`table-row`** : ligne de tableau, **`table-column`** : colonne de tableau **`table-caption`** : titre de tableau **`table-row-group`** : groupe de lignes **`table-column-group`** : groupe de colonnes de tableau **`table-header-group`** : groupe d'en-têtes de tableau **`table-footer-group`** : groupe de pieds de tableau **`none`** : l'élément est invisible et ses dimensions sont nulles.
Héritage	*Non.*

À noter **Différence entre `visibility: hidden;` et `display: none;`**

Ces deux règles permettent de masquer un objet dans la page.

- Avec `visibility: hidden;` l'objet est masqué, mais occupe toujours la même place qu'auparavant dans la page.
- Avec `display: none;` l'objet devient invisible également, mais sa place n'est plus réservée sur la page (aucune boîte n'est générée).

Délimitation des blocs

Même en voyant la page web s'afficher à l'écran, il n'est pas toujours facile de comprendre la position et les dimensions des blocs qui la composent.

Il existe une méthode toute simple, mais qui rend d'immenses services, pour voir ces blocs à l'écran : il suffit de les délimiter par une bordure, en leur appliquant temporairement la propriété suivante :

```
border: 1px solid red;
```

Une fois ces blocs encadrés, comme le montre la figure 5-8, il est beaucoup plus facile de voir comment ils sont organisés entre eux, comment sont prises en compte les dimensions et les marges, quelles sont les différences d'interprétation entre deux navigateurs, etc.

Cette méthode peut être affinée par l'utilisation de couleurs d'encadrement variées pour les différents blocs, de façon à mieux les repérer.

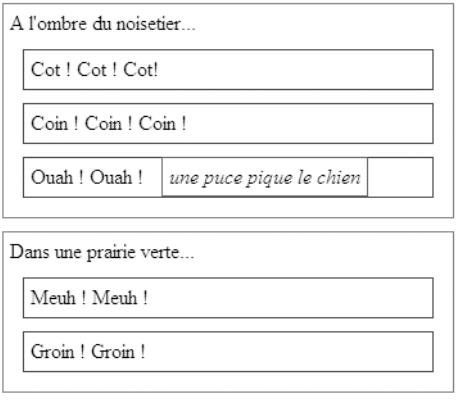


FIGURE 5–8 *Vous souvenez-vous du passage sur l'héritage, au début de cet ouvrage ? Une fois les blocs encadrés, leur imbrication et leurs marges sont nettement visibles.*

Exemples de positionnement

Pour comprendre les différents types de positionnement et leur utilisation, rien de tel qu'un exemple.

Il s'agit d'une page web complète, qui parle de nature et utilise tous les types de positionnement : un peu de fraîcheur et beaucoup d'explications !

Voici le code de cette page, dont la feuille de style sera expliquée en détail par la suite et qui produit le résultat montré par la figure 5-9.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
          "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
<head>
  <meta http-equiv="content-type"
        content="text/html; charset=utf-8" />
  <title>La nature : Fleurs et plantes</title>
  <style type="text/css">
  <!--
* { margin: 0; padding: 0; }

img#frise { display: block; (1)
            width: 100%; height: 30px; }

div#titre { background-color: skyblue;
            text-align: center; height: 60px;
            position: relative; } (2)
#titre img { position: absolute;
             top: 5px; left: 25%; (3)
             height: 50px; width: 40px; }
#titre h1, #titre h2 { margin: 0 auto; (4)
                       width: 200px; color: green; }
#titre h1 { background-color: khaki; } (5)
#titre h2 { position: relative;
            top: -10px; left: 50px; } (6)
h2#titre_fixe { position: fixed;
                top: 200px; left: 0;  (7)
                width: 80px; padding: 20px;
                background-color: lightgreen;
                text-align: center;}
```

```
div#galerie { position: absolute;
              top: 90px; left: 120px;} ❽
#galerie img { margin: 20px; width: 200px; height: 230px;
               float: left; } ❾
  -->
  </style>
</head>

<body>
  <img id="frise" src="nuages.jpg" alt="" title="" /> ❿
  <div id="titre"> ⓫
  <img src="arbre.gif" alt="logo arbre" title="" /> ⓬
<h1>LA NATURE</h1> ⓭
<h2>EN IMAGES</h2> ⓮
  </div>
  <h2 id="titre_fixe">Fleurs<br />et<br />plantes</h2> ⓯
  <div id="galerie"> ⓰
<img src="img01.jpg" alt="Chêne"
     title="Chêne" />
<img src="img02.jpg" alt="Pin"
     title="Pin" />
<img src="img03.jpg" alt="Cognassier"
     title="Cognassier" />
<img src="img04.jpg" alt="Palmier"
     title="Palmier" />
<img src="img05.jpg" alt="Pommier"
     title="Pommier" />
<img src="img06.jpg" alt="Saule"
     title="Saule" />
  </div>
</body>

</html>
```

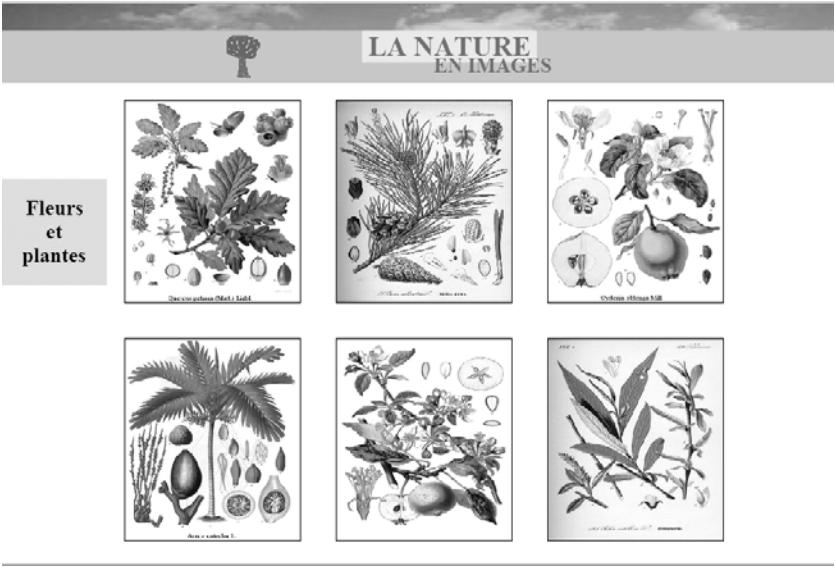


Figure 5–9 *Affichage de notre code exemple. Les illustrations proviennent du site www.wikipedia.org.*

Image du haut (nuages)

Cette image ⑩ a été insérée pour différencier, du point de vue du positionnement, le haut de l'élément <body> (haut de la page) et le haut du bloc (bandeau uni contenant le titre principal).

> **À noter** **Suppression de l'espace vertical sous une image**
>
> Un espace vertical apparaissait initialement sous l'image ; pour le faire disparaître, il a fallu déclarer cette image comme bloc : `display: block` ❶.

Image de l'arbre en position absolue

La figure 5-10 montre les modifications apportées à la position du logo en forme d'arbre, qui correspond à l'image ⓬.

1. Initialement, l'arbre est positionné dans le flux. Il se place en haut de son bloc conteneur (dont le contenu est centré par une propriété de style). Le bloc suivant (titre de niveau 1 « La nature » ⓭) est placé en dessous.
2. Cette image d'arbre est positionnée en absolu, par la règle ❸ :

   ```
   #titre img { position: absolute; top: 5px; left: 25%; }
   ```

   Comme le bloc parent `<div id="titre">` n'est pas positionné, la position de l'arbre est calculée à partir du bloc `<body>`, c'est-à-dire du début de la page.
3. Le même positionnement étant conservé pour l'arbre, la règle `div#titre { position: relative; }` ❷ sert à placer le bloc parent. La position de l'arbre est alors calculée à partir des limites du bloc `<div id="titre">` ⓫ qui le contient.

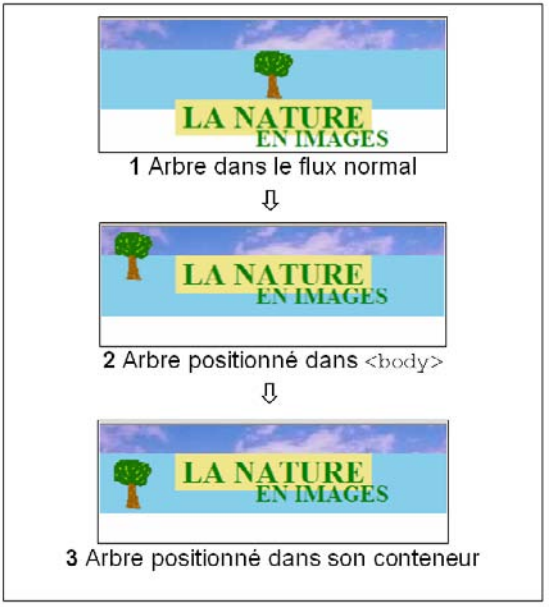


FIGURE 5–10 *Les différentes étapes du positionnement de l'arbre*

Sous-titre « En images » en position relative

Sur la figure 5-11 apparaissent le titre « La nature » ⑬ et le sous-titre « En images » ⑭ de la page.

Dans la première image, ils sont affichés dans le flux normal. Le sous-titre se place en dessous du titre principal, centré comme lui.

Grâce à un positionnement relatif du sous-titre « En images », il est possible de décaler celui-ci vers le haut et vers la droite, en utilisant la règle de style : `#titre h2 { position: relative; top: -10px; left: 50px; }` ⑥ qui produit :

- un décalage négatif à partir du haut, pour que le sous-titre chevauche le bloc contenant le titre « La nature » ;
- un décalage positif à partir de la gauche, pour décaler le sous-titre vers la droite.

Le résultat obtenu est visible sur la deuxième image de la figure 5-11.

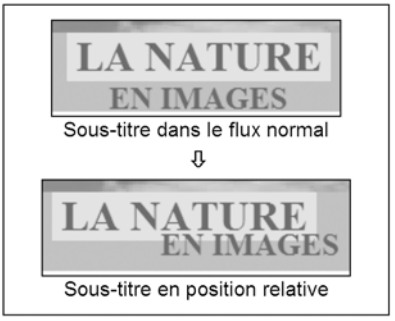


Figure 5–11 *Décalage relatif du sous-titre par rapport à sa position normale*

Centrage horizontal du titre

Le centrage horizontal des blocs <h1> ⑬ et <h2> ⑭ qui contiennent le titre « La nature » et le sous-titre « En images » s'effectue à l'aide de la règle :

```
#titre h1, #titre h2 { margin: 0 auto; } ④
```

Toutefois, cette règle est mal interprétée par Internet Explorer 6, qui ne centre pas les blocs de cette façon. Pour ce navigateur, il faut ajouter une règle d'alignement qui concerne le bloc conteneur :

```
div#titre { text-align: center; } ❷
```

Selon les normes CSS, cette dernière règle ne sert qu'au centrage des éléments en ligne (texte et images) et ne concerne pas les blocs. Néanmoins, Internet Explorer 6 l'utilise aussi pour le centrage des blocs.

Titre latéral fixé sur l'écran

Le titre « Fleurs et plantes » ⓯, qui se trouve à gauche de la page, doit rester visible et à la même position sur l'écran, même lorsque la page défile. La figure 5-12 illustre ce mécanisme.

Nous utiliserons pour ceci la règle suivante :

```
h2#titre_fixe { position: fixed; top: 200px; left: 0; } ❼
```

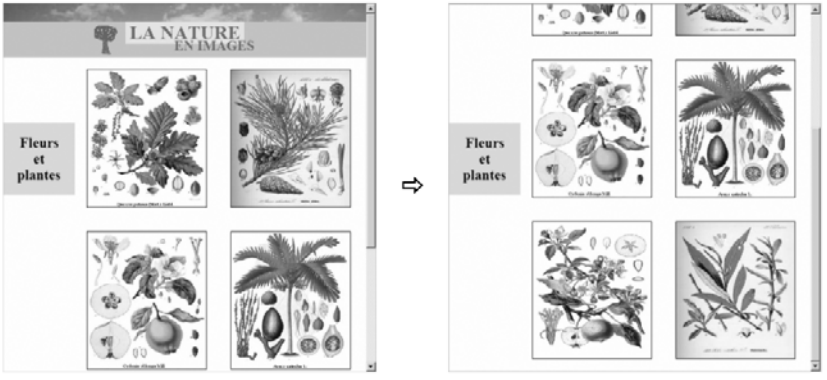


FIGURE 5–12 *Le titre « Fleurs et plantes » reste fixe lors du défilement de la page.*

> ATTENTION **Navigateur récalcitrant**
>
> Contrairement à Firefox et Internet Explorer à partir de sa version 7, Internet Explorer 6 ne reconnaît pas cette propriété `position: fixed;`. Il existe toutefois des solutions de remplacement en Javascript.

Position absolue pour la galerie d'images

Le bloc ⑯ qui contient toutes les images doit se juxtaposer au titre latéral « Fleurs et plantes » ⑮. Pour effectuer cette mise en page, le CSS nous évite les affres des tableaux en nous proposant un positionnement absolu avec la règle ⑧ :

```
div#galerie { position: absolute; top: 90px; left: 120px; }
```

Les décalages sont exprimés par rapport à l'élément parent de `<div id="galerie">` ⑯, donc par rapport à `<body>`. Les 90 pixels correspondent à la hauteur cumulée des deux blocs du haut (frise ⑩ et bandeau de titre ⑪) et les 120 pixels à la largeur du bloc latéral « Fleurs et plantes » ⑮.

Le résultat se trouve sur la figure 5-13, où le bloc qui contient la galerie d'images est entouré de pointillés.

Figure 5–13 *Le bloc qui contient la galerie d'images est placé dans la page, en positionnement absolu.*

Images côte à côte en position flottante

Les images qui constituent le corps de la page sont alignées côte à côte, avec retour à la ligne automatique en fonction de la largeur de la fenêtre. C'est le positionnement flottant à gauche qui réalise cette fonction :

```
#galerie img { float: left; } ❾
```

Ce flottement s'effectue à l'intérieur du bloc conteneur, soit ici le bloc `<div id="galerie">` ⓰. La figure 5-14 montre bien que le nombre d'images par ligne s'adapte à la largeur de la fenêtre.

PRÉCISION Quelques efforts de présentation

Au départ, les images flottantes vont s'agglutiner les unes aux autres de façon peu esthétique. Pour une présentation harmonieuse, il faudra :

- donner une taille homogène à toutes les images, en utilisant les propriétés `width` et `height` ;
- régler les marges avec `margin` ou `padding`, pour espacer les images entre elles.

Notez que des images en mode portrait et en mode paysage peuvent cohabiter dans un positionnement flottant, à condition de former des blocs de dimensions homogènes : la somme taille + marge doit rester constante, en largeur comme en hauteur.

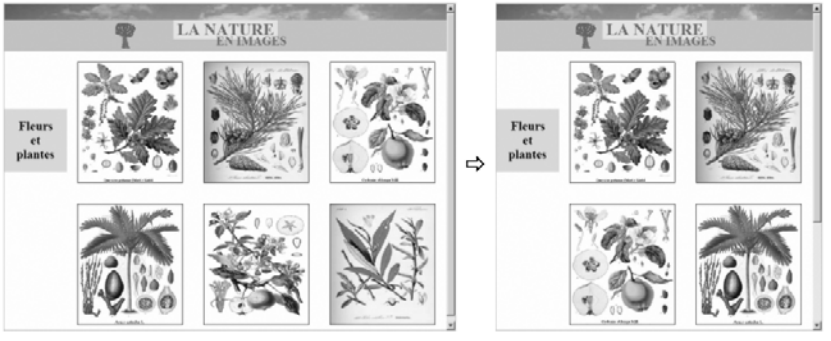


FIGURE 5–14 *Grâce au positionnement flottant, le nombre d'images sur chaque ligne s'adapte à la largeur de la fenêtre.*

À NOTER **Habillage d'une image**

Le positionnement flottant permet aussi d'habiller une image avec du texte, comme le montre la figure 5-15 : l'image du pingouin Tux est flottante à droite, elle est donc habillée par le texte qui la suit dans le code de la page.

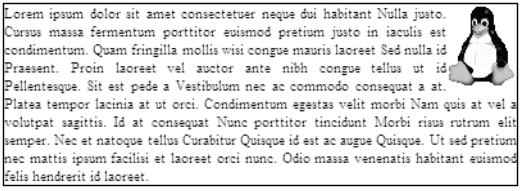


Figure 5–15 *Image flottante à droite, habillée par le texte qui la suit.*

Centrage d'éléments à l'intérieur des blocs

Il est souvent nécessaire de centrer textes, images ou blocs, soit horizontalement, soit verticalement. Voici, de façon simple, comment procéder avec les feuilles de style.

Dans chaque cas, il faudra distinguer :

- le centrage des *éléments en ligne* (texte, images...), qui s'affichent les uns à la suite des autres ;
- le centrage des *blocs* (`<div>`, `<p>`, `<h1>`, ...), éléments qui se placent les uns sous les autres et qui n'utilisent pas les mêmes propriétés.

Centrage horizontal

Mieux vaut éviter la vieille balise HTML `<center>` ou les attributs `align= "center"` à l'intérieur des balises : ces méthodes sont à la fois obsolètes et déconseillées, car elles mélangent mise en forme et contenu de la page.

Centrage horizontal d'éléments en ligne

Pour centrer horizontalement *un élément en ligne*, il suffit d'attribuer à son *bloc conteneur* la propriété suivante :

```
text-align: center;
```

Centrage horizontal de blocs

Pour centrer horizontalement *un bloc* à l'intérieur de son conteneur, le principe consiste à lui attribuer des marges identiques à gauche et à droite, en utilisant les propriétés suivantes :

```
margin-left: auto; margin-right: auto;
```

Cependant, l'ancienne version Internet Explorer 6, encore assez souvent utilisée, n'interprète pas correctement cette marge automatique. En revanche, elle centre les blocs avec la même propriété que celle qui sert à centrer les éléments en ligne. Spécialement pour ce navigateur, il faudra donc associer au *bloc conteneur* la propriété :

```
text-align: center;
```

> À NOTER **Largeur du bloc à centrer**
>
> Pour que le centrage horizontal change quelque chose à l'affichage, il faut bien sûr que le bloc à centrer soit plus étroit que son bloc conteneur.

Centrage vertical

Là, il n'y a aucun risque d'utiliser une balise périmée, car il n'en existe pas en HTML pour centrer verticalement ! Il faudra utiliser plusieurs propriétés pour placer l'élément au milieu de son conteneur.

Centrage vertical d'éléments en ligne

Pour centrer verticalement un élément en ligne, il faut déclarer une *hauteur de ligne* égale à la hauteur de l'élément. Exemple :

```
p.milieu { height: 150px; line-height: 150px; }
```

À noter **Centrage d'éléments comprenant plusieurs lignes**

La technique précédente ne fonctionne que si les éléments à centrer tiennent sur une seule ligne.
Pour un élément à centrer verticalement qui comporte deux ou plusieurs lignes, la méthode suivante est applicable :
XHTML

```
<span>texte sur une ligne</span>
<span class="sur2lignes">texte plus long,<br />
                         sur deux lignes</span>
```

CSS

```
span { height: 2em; line-height: 2em; }
span.sur2lignes { line-height: 1em; }
```

Cette technique est généralisable à un nombre quelconque de lignes, avec `line-height = height / nombre de lignes`.

Centrage vertical de blocs

Le bloc à centrer verticalement doit évidemment avoir une hauteur inférieure à celle de son conteneur.

Le centrage vertical d'un bloc utilise le principe suivant, qui est illustré par la figure 5-16.

- Le haut du bloc à centrer est d'abord placé au milieu de son conteneur, en position absolue (à 50 % du haut du bloc conteneur).
- Une marge de valeur négative est alors appliquée en haut du bloc ; elle représente la moitié de la hauteur de l'élément à centrer.

Attention **Le bloc conteneur doit être positionné**

Étant donné que le bloc à centrer verticalement est placé en position absolue à l'intérieur de son conteneur, il faut songer que ce bloc conteneur doit être lui-même positionné. Si ce n'est pas le cas, il faudra lui attribuer la propriété `position: relative;`.

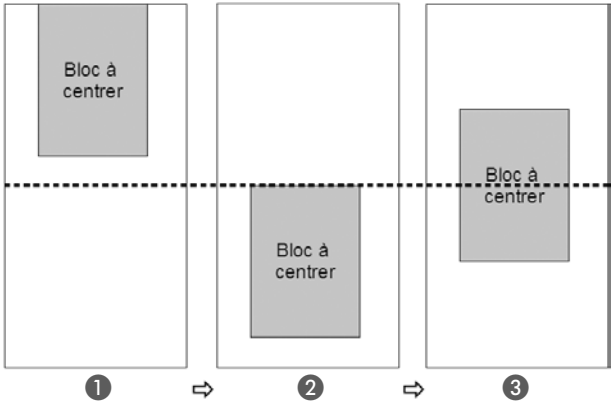


FIGURE 5–16 ❶ *Position initiale du bloc à centrer ;* ❷ *le haut du bloc à centrer est placé au milieu du bloc conteneur ;* ❸ *une marge négative est appliquée en haut du bloc à centrer : la moitié de sa hauteur.*

Exemple de centrage vertical

Pour le conteneur

```
position: relative;
```

Pour le bloc à centrer

```
position: absolute;
top: 50%;
height: 140px;
margin-top: -70px;
/* - height/2 ; attention au signe moins */
```

VARIANTE **Centrage horizontal d'un bloc**

Cette méthode peut aussi être appliquée au centrage horizontal d'un bloc. Exemple :

```
p { position: absolute;
    left: 50%; margin-left: -150px; width: 300px;}
```

Le bloc en question est placé au milieu de son conteneur à partir de la gauche. Ensuite, une marge négative lui est affectée à gauche, de la moitié de sa largeur.

Avec ces propriétés liées aux blocs et au positionnement d'éléments, nous avons à présent fini le tour des propriétés de style CSS, du moins celles destinées à mettre en forme les pages web affichées à l'écran.

Le prochain chapitre va nous montrer qu'il existe également des propriétés de style destinées à d'autres médias. Nous en découvrirons quelques-unes, liées à l'impression.

Figure 5–17 *Les pages web sont constituées de blocs. Ceux-ci sont nettement visibles dans cet exemple, extrait du site http://www.csszengarden.com/tr/francais (version « Par avion », par Emiliano Pennisi - http://www.peamarte.it/01/metro.html).*

chapitre

6

Différents types de média

Le principal média utilisé est l'affichage à l'écran, mais il en existe d'autres, notamment l'impression et le son.

Sommaire

Sans le dire et par une tacite connivence avec les navigateurs Internet, nous avons utilisé un type de média, toujours le même : le média *affichage à l'écran*.

Les feuilles de style permettent cependant de gérer d'autres médias. Parmi tous ceux qui sont proposés en CSS, les deux principaux sont l'impression et la diffusion de son.

Types de média

À l'intérieur d'une feuille de style interne ou externe, il est possible de déclarer le type de média concerné par un groupe de propriétés. Cette déclaration est facultative ; elle s'effectue sous la forme :

```
@media xxx {    /* xxx= type de média concerné */
  ...Propriétés pour ce type de média...
}
```

Exemple

```
style type="text/css">
<!--
  @media screen {
        * { font-style: Arial, sans-serif; }
  }
  @media print {
        * { font-style: "Times New Roman", serif; }
  }
  @media all {
        * { background-color: white; }
  }
-->
</style>
```

Les types de média possibles sont les suivants :

- **`all`** : tous les médias sont concernés par les propriétés indiquées ;
- **`aural`** : diffusion de son ;
- **`braille`** : écran tactile pour les non-voyants ;

- `embossed` : impression en braille ;
- `handheld` : appareil de poche (visuel sur petit écran ou sonore) ;
- `print` : impression papier ;
- `projection` : affichage par vidéoprojecteur (sur grand écran) ;
- `screen` : écran couleur (affichage classique sur un ordinateur) ;
- `speech` : lecture vocale ;
- `tty` : impression avec caractères de largeur fixe (sur « télétype ») ;
- `tv` : télévision.

> À noter **Emplacement de la déclaration du type de média**
>
> Le type de média peut être spécifié :
> - à l'intérieur d'une feuille de style interne, comme dans l'exemple précédent ;
> - dans l'appel à une feuille de style externe, comme ci-après :
>
> ```
> <link  rel="stylesheet" type="text/css"
>        media="print"    href="impression.css">
> ```

Média paginé : styles pour l'impression

Les propriétés liées à l'impression d'un document concernent sa taille, les sauts de page, ainsi que la gestion des veuves et des orphelines.

Vous ne connaissez pas l'histoire de la veuve, ni celle de l'orpheline ? Préparez un mouchoir, je vous la raconte (rassurez-vous, elle finit bien !). Dans les deux cas, il s'agit d'un paragraphe dont une ligne est toute seule, perdue sur une autre page, ce qui est fâcheux d'un point de vue esthétique.

Pour la veuve, c'est une pauvre petite ligne qui se trouve seule en haut d'une page, alors que tout le reste du paragraphe auquel elle appartient se trouve en bas de la page précédente. Dans ce cas, il pourra être demandé qu'une ligne se détache du paragraphe pour lui tenir compagnie.

Quant à l'orpheline, elle est isolée en bas d'une page, tandis que toute sa famille, c'est-à-dire le reste du paragraphe se trouve en haut de la page suivante. Alors, par une bonté du logiciel, elle pourra le rejoindre sur cette nouvelle page.

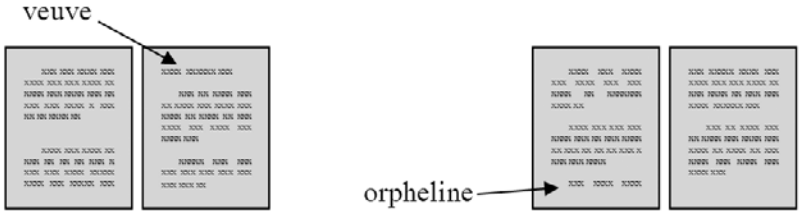


FIGURE 6–1 *Veuve et orpheline*

PARENTHÈSE

L'effet « veuve et orpheline » dans un traitement de texte

En général, les traitements de texte gèrent les veuves et les orphelines. C'est pourquoi, dans certains cas, l'ajout d'un saut de ligne dans une page entraîne le passage de deux lignes vers la page suivante, ou parfois la suppression d'une ligne de la page courante fait revenir deux lignes de la page suivante.

Voici les propriétés liées à l'impression, qui seront donc en général placées entre accolades, dans la partie `@media print { }` de la feuille de style.

Gestion des veuves

La propriété `widows` permet de définir le nombre *minimum* de lignes d'un même paragraphe à afficher ensemble en haut d'une page. En dessous de ce nombre, la suppression des « veuves » est activée : une ou plusieurs lignes du même paragraphe, qui se trouvaient en bas de la page précédente, sont transférées en haut de la nouvelle page.

TABLEAU 6–1 **Propriété widows**

Propriété	`widows`
Exemple	`body { widows: 3; }`
Valeurs possibles	**Nombre entier** (valeur par défaut : 2) qui indique le nombre minimum de lignes qui peuvent rester en haut d'une page.
Héritage	Propriété *héritée*. Pour retrouver la valeur intiale, indiquez 2.

Gestion des orphelines

La propriété `orphans` indique le nombre minimum de lignes d'un même paragraphe à afficher en bas d'une page. Si ce nombre n'est pas atteint, ces lignes sont considérées comme « orphelines », et sont donc transférées en haut de la page suivante.

Tableau 6–2 **Propriété orphans**

Propriété	`orphans`
Exemple	`body { orphans: 3; }`
Valeurs possibles	**Nombre entier** (valeur par défaut : 2) qui indique le nombre minimum de lignes qui peuvent rester en bas d'une page.
Héritage	Propriété *héritée*. Pour retrouver la valeur intiale, indiquez 2.

Saut de page avant

Associée à un élément de la page, la propriété `page-break-before` permet d'indiquer s'il doit éventuellement être imprimé en haut d'une nouvelle page.

Tableau 6–3 **Propriété page-break-before**

Propriété	`page-break-before`
Exemple	`.chapitre { page-break-before: always; }`
Valeurs possibles	`auto` : le saut de page s'effectue automatiquement lorsque le bas de la page est atteint (valeur par défaut), `always` : saut de page obligatoire avant, `avoid` : pas de saut de page avant, `left` : un ou deux saut(s) de page avant, pour que l'élément *qui suit* l'élément concerné se trouve dans une page de gauche, `right` : un ou deux saut(s) de page avant, pour que l'élément *qui suit* l'élément concerné se trouve dans une page de droite.
Héritage	*Non*.

Saut de page après

La propriété **page-break-**after sert à préciser si un élément de la page doit être ou non suivi d'un saut de page.

TABLEAU 6–4 **Propriété page-break-after**

Propriété	**page-break-after**
Exemple	`.conclusion { page-break-after: left; }`
Valeurs possibles	**auto** : le saut de page s'effectue automatiquement lorsque le bas de la page est atteint (valeur par défaut), **always** : saut de page obligatoire après, **avoid** : pas de saut de page après, **left** : un ou deux saut(s) de page après, pour que l'élément *qui suit* l'élément concerné se trouve dans une page de gauche, **right** : un ou deux saut(s) de page après, pour que l'élément *qui suit* l'élément concerné se trouve dans une page de droite.
Héritage	*Non.*

Coupure par un saut de page

Certaines parties de texte perdent leur sens ou sont beaucoup moins lisibles si elles se trouvent coupées par un saut de page. Cette coupure peut être évitée grâce à la propriété **page-break-inside**.

TABLEAU 6–5 **Propriété page-break-inside**

Propriété	**page-break-inside**
Exemple	`table { page-break-inside: avoid; }`
Valeurs possibles	**auto** : le saut de page s'effectue automatiquement lorsque le bas de la page est atteint, l'élément concerné pouvant donc se trouver sur deux pages (valeur par défaut), **avoid** : pas de saut de page *à l'intérieur* de l'élément concerné : il doit se trouver entier sur une seule page.
Héritage	Propriété *héritée*. Pour retrouver la valeur initiale, utilisez `auto`.

Dimensions d'une page

Les dimensions et l'orientation des pages sont spécifiées à l'aide de la propriété `size`. Le sélecteur peut être par exemple `@page` pour indiquer « toutes les pages » ; nous examinerons plus loin les types de sélecteur possibles.

Tableau 6–6 **Propriété size**

Propriété	`size`
Exemple	`@page { size: landscape; }`
Valeurs possibles	`auto` : dimensions de la page standard (valeur par défaut), `landscape` (orientation « paysage ») `portrait` (orientation « portrait »), ou deux valeurs séparées par un espace : `largeur hauteur` (valeurs exprimées généralement en cm ou en in - pas de %).
Héritage	Propriété *héritée*. Pour retrouver la valeur initiale, utilisez `auto`.

Sélecteur de page

Le sélecteur `@page` permet de spécifier les caractéristiques de pagination (taille, orientation portrait ou paysage, marges...). Il peut être complété par `:first`, `:left` ou `:right` pour n'attribuer les propriétés en question qu'à une partie du document.

Pour toutes les pages

```
@page { margin: 1.5cm; }
```

Pour la première page

```
@page :first { margin-top: 5cm; }
```

Pour toutes les pages de gauche

```
@page :left { margin-right: 2cm; }
```

Pour toutes les pages de droite

```
@page :right { margin-left: 2cm; }
```

Pages nommées

Il est possible de donner un nom à un type de page et de lui associer des propriétés de mise en forme, en utilisant la syntaxe suivante :

```
@page nom_de_page_choisi { propriétés_associées }
```

Exemples :

```
@page paysage { size: landscape; }
@page formatA5 { size: 10cm 15cm; }
```

Par la suite, un élément peut faire référence à ce nom de page pour en prendre toutes les caractéristiques. Il faut pour cela utiliser la propriété page, décrite ci-après.

Référence à un type de page

À ne pas confondre avec le sélecteur @page, la propriété **page** sert à nommer un ou plusieurs éléments. Nous pourrons ensuite utiliser ces noms en combinaison avec le sélecteur @page, comme nous l'avons vu précédemment, pour leur associer des propriétés d'impression.

TABLEAU 6–7 **Propriété page**

Propriété	**page**

Exemple	`img { page: paysage; }`
	`#notice { page: formatA5; }`
	Les types de page nommés `paysage` et `formatA5` auront été définis précédemment, par exemple :
	@page paysage { size: landscape; }
	@page formatA5 { size: 10cm 15cm; }

Valeurs possibles	**auto** : pas de nom de page associée (valeur par défaut), ou **nom d'une page** défini par `@page xxx { ... }`.
Héritage	Propriété *héritée*. Pour retrouver la valeur initiale, utilisez `auto`.

> Attention **Saut de page automatique**
>
> Si deux éléments qui se suivent sont associés à des types de page différents, alors un saut de page est automatiquement inséré entre eux. La même règle est d'ailleurs appliquée par les traitements de texte : il ne peut pas y avoir deux mises en forme différentes pour la même page. Par exemple, dans un texte en mode portrait, si un paragraphe est associé au mode paysage, un saut de page sera automatiquement inséré avant et après.

Média sonore : fonctions audio

Les propriétés audio concernent la mise en forme des extraits sonores utilisés. Elles seront placées, dans la feuille de style, à l'intérieur des accolades dans `@media aural { ... }` ou `@media speech { ... }`.

> Attention **Propriétés peu reconnues**
>
> Actuellement, ces propriétés sont peu ou pas prises en compte par les principaux navigateurs.

Figure 6–2 *Les fonctions audio s'intégrant de plus en plus à l'informatique, elles seront progressivement prises en compte par les navigateurs web.*

Voici un résumé de ces propriétés sonores. Pour tout détail à leur sujet, consulter le site officiel du W3C www.w3c.org ou sa traduction française,

sur www.yoyodesign.org. Plus précisément, l'index de toutes les propriétés CSS 2 se trouve à l'adresse :
http://www.yoyodesign.org/doc/w3c/css2/indexlist.html.

`cue-before`	son à diffuser avant un élément
`cue-after`	son à diffuser après un élément
`cue`	raccourci pour `cue-before` et `cue-after`
`pause-before`	pause avant un élément sonore
`pause-after`	pause après un élément sonore
`pause`	pause avant et après un élément sonore
`play-during`	diffusion d'un son en arrière-plan
`volume`	volume moyen du son
`azimuth`	angle horizontal de l'origine du son
`elevation`	angle vertical de l'origine du son

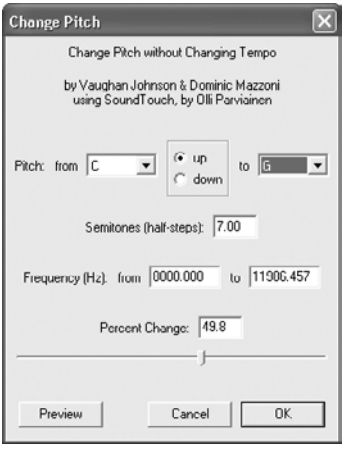


FIGURE 6–3 *Le réglage « pitch » dans le logiciel libre Audacity de traitement du son (extrait du site http://sourceforge.net)*

`pitch`	fréquence moyenne à utiliser
`pitch-range`	étendue des variations de tonalité
`richness`	clarté, c'est-à-dire portée de la voix

`stress`	ampleur des inflexions de voix à l'accentuation
`speech-rate`	vitesse de lecture du texte
`voice-family`	types de voix à utiliser (analogue à `font-family` pour les polices de caractères)
`speak`	façon de lire le texte
`speak-header`	façon de lire les en-têtes de tableau
`speak-numeral`	façon de lire les nombres
`speak-punctuation`	lecture ou non des signes de ponctuation

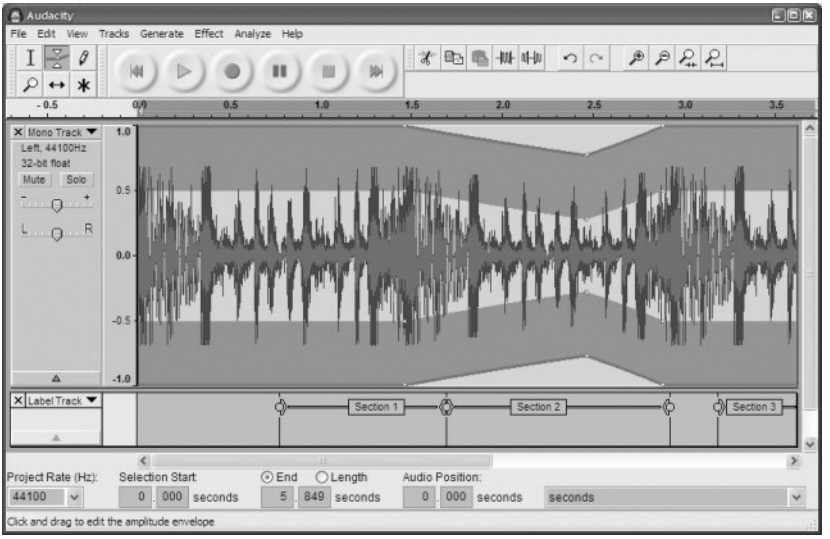


Figure 6–4 *Écran principal du logiciel libre Audacity, éditeur de son très apprécié (extrait du site http://sourceforge.net)*

Il était intéressant de faire le tour de ces propriétés, destinées à d'autres médias que l'affichage à l'écran : même si elles ne sont pas encore généralisées à l'ensemble des navigateurs, elles le seront bien un jour !

D'ailleurs, même les propriétés destinées au média « écran » ne sont pas toujours interprétées de façon identique par tous les navigateurs. Le chapitre qui suit nous montre comment adapter nos pages en fonction de ces défauts...

chapitre

7


Le meilleur des Firefox à ce jour
Avec plus de 15 000 améliorations, Firefox 3 est plus rapide, plus sûr et plus pratique que jamais.
Firefox 3
Téléchargement gratuit
Internet Explorer 7
Bienvenue dans Internet Explorer 7
Prenez vos repères
La facilité avant tout
Protection dynamique
OPERA
Opera Browser Ver: 9.51
Explore our newest features
Quick Find
New browser engine
Safari
Le meilleur navigateur au monde.
Désormais aussi sur Windows.
Konqueror
Konqueror Homepage
Konqueror - Web Browser, File Manager - and more!

Règles spécifiques à certains navigateurs

Pour pallier les interprétations hors normes de certains navigateurs, il est parfois nécessaire d'utiliser des solutions de contournement. Voici quelques techniques pour résoudre les problèmes les plus courants.

Sommaire

Nous commençons maintenant à nous débrouiller avec les feuilles de style, cependant il reste un problème à résoudre. Par exemple, si notre mise en page produit de magnifiques résultats avec Firefox, Opera et d'autres navigateurs, il se peut qu'elle se trouve toute chamboulée sur certaines versions d'Internet Explorer ! Effectivement, nous devrons tester nos pages sur différentes versions de navigateurs et procéder à quelques adaptations si besoin est.

Test des pages sur plusieurs navigateurs

Lors de l'écriture d'une page, il est important de contrôler au fur et à mesure le résultat affiché. Certains éditeurs proposent une fenêtre d'aperçu rapide qui simule le navigateur, mais il est utile de vérifier la page dans les conditions réelles d'affichage.

Pour cela, nous ouvrirons deux fenêtres en parallèle, celle de l'éditeur utilisé pour écrire le code et une autre qui affiche la page dans un navigateur. Le test continu de la page s'effectue alors de la façon suivante :

- après chaque écriture ou modification d'une partie du code, il faut penser à enregistrer la page ;
- ensuite, il suffit de basculer vers la fenêtre du navigateur (par exemple avec la combinaison des touches *Alt* et *Tabulation*) puis d'actualiser l'affichage (c'est souvent la touche *F5* qui est affectée à cette fonction).

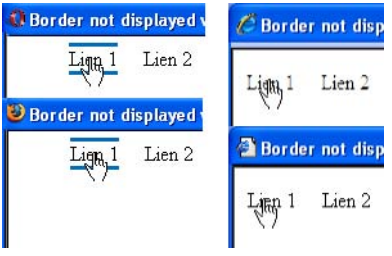


FIGURE 7–1 *Exemple de page qui s'affiche correctement sur Opera et Firefox (à gauche) et qui se comporte différemment sur Internet Explorer 6 et 7 (à droite). Extrait de la page http://css.tests.free.fr/exemple33.php.*

La vérification avec différentes versions de plusieurs navigateurs ne s'effectuera qu'une fois la page terminée, sans quoi le développement deviendrait beaucoup trop long. Nous utiliserons donc un seul navigateur pour tester notre page tout au long de sa construction. Mais alors, lequel choisir ?

Il nous faut un navigateur qui interprète correctement les standards et qui soit d'utilisation courante, autant que possible. La première condition élimine le navigateur très utilisé Internet Explorer dans ses versions 6 et 7, car il est trop éloigné des normes XHTML/CSS, notamment Internet Explorer 6. Un bon choix consiste à utiliser Firefox version 2 ou 3, qui respecte les normes depuis fort longtemps et qui est le deuxième navigateur le plus courant.

Une fois la page écrite et testée dans ces conditions, il faudra en vérifier l'affichage sur d'autres navigateurs et d'autres versions, par exemple :

- Internet Explorer 6, 7, 8 ;
- Firefox 2 et 3 ;
- éventuellement Safari, Opera, Konqueror... dans leur dernière version.

Les différences d'affichage devraient être mineures ou inexistantes avec la plupart de ces logiciels, parfois plus importantes sur Internet Explorer 7. Mais c'est avec Internet Explorer 6 qu'il y aura le plus d'écarts.

Choix **Quels navigateurs et quelles versions faut-il viser ?**

Évidemment, il ne sera pas possible de créer des pages qui s'afficheront de façon parfaite sur toutes les versions de tous les navigateurs en remontant jusqu'à Mathusalem ! Encore qu'à l'époque de ce dernier, les navigateurs étaient plus sur la mer que dans un ordinateur...
Il faudra donc sélectionner et ne tester que les principaux de ces « butineurs », appelés *browsers* en anglais. Aussi est-il intéressant de consulter les statistiques qui donnent le taux d'utilisation de chaque version des différents navigateurs. Elles sont accessibles à partir de la page :

▸ **http://www.w3schools.com/browsers/**

Attention à l'interprétation de ces statistiques :

- d'une part, ces données sont des moyennes mondiales qui peuvent varier suivant le pays ;
- d'autre part, elles peuvent être à pondérer en fonction du type de public visé. Par exemple, un public d'informaticiens utilisera plus souvent un navigateur conforme aux standards du W3C.

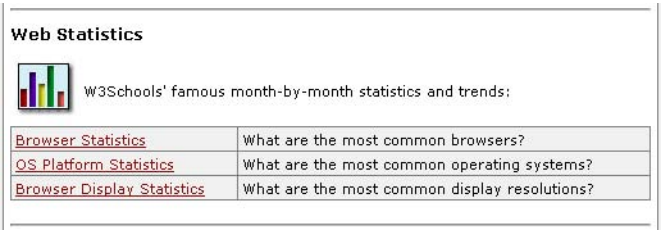


FIGURE 7–2 *Extrait de la page http://www.w3schools.com/browsers/ : accès aux statistiques d'utilisation des différents navigateurs Internet.*

Malgré l'apparition d'Internet Explorer 7 et l'arrivée d'Internet Explorer 8, le navigateur Internet Explorer 6 reste encore utilisé par un certain nombre d'internautes qui n'ont pas effectué la mise à jour vers une version ultérieure. Ceci pose bien des problèmes, car cette version 6 fait depuis longtemps figure de vilain petit canard : elle interprète certaines propriétés CSS d'une manière personnelle, à l'écart des normes, et en ignore d'autres.

La version Internet Explorer 7 nécessite parfois aussi quelques adaptations. Heureusement (mieux vaut tard que jamais), Microsoft a fait un effort dans la conformité aux standards XHTML-CSS pour concevoir Internet Explorer 8 : leurs ingénieurs ont enfin reçu la consigne d'ouvrir toutes les pages de la notice !

> A NOTER **Installer différentes versions d'un même logiciel**
>
> Pour afficher notre page dans plusieurs versions d'un même logiciel, il faudra installer celles-ci sur notre ordinateur. Cette cohabitation risque d'entraîner une certaine confusion ou même d'être impossible comme pour Internet Explorer (il est intégré au système d'exploitation Windows et toute nouvelle version installée remplace la précédente).
>
> Pour résoudre ce problème, il suffit d'utiliser :
>
> - pour Firefox, une version portable du logiciel, sans installation et utilisable à partir d'une clef USB ou d'un disque dur externe ;
> - pour Internet Explorer, les versions dites « standalone » qui pourront cohabiter avec celle intégrée au système d'exploitation.

Du point de vue du codage, la question qui se pose est la suivante : comment conserver notre page correctement conçue et l'adapter en même temps au comportement spécifique de tel ou tel navigateur ? Nous allons étudier quelques-unes des techniques qui permettent de résoudre ce casse-tête.

Adaptation du code aux navigateurs

Pour contourner les bogues d'Internet Explorer 6 ou 7 (appelés en raccourci IE 6 et IE 7), il est parfois nécessaire :

- d'écrire des balises XHTML ou des règles de style conformes aux normes, mais qu'il vaut mieux voir ignorées par IE 6 ou 7, car eux ne les interprètent pas correctement ;
- d'en utiliser d'autres, destinées uniquement à Internet Explorer 6 ou 7.

Appelées *hacks* en anglais, ces *astuces* dans le code XHTML ou dans les sélecteurs des feuilles de style nous donnent la possibilité d'effectuer cette sélection de navigateurs.

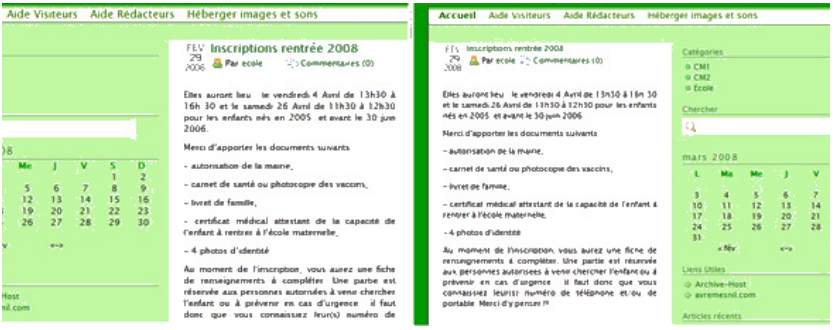


Figure 7–3 *Bien que le code XHTML/CSS soit le même, les blocs sont placés différemment dans Internet Explorer (à gauche) et dans Firefox (à droite). Extrait de la page : http://www.geekeden.com/lofiversion/index.php/t11262.html.*

Balises XHTML conditionnelles

Comme vous l'avez compris, les problèmes à contourner viendront essentiellement d'Internet Explorer, en particulier des versions 6 et antérieures, ainsi que – dans une moindre mesure – de la version 7.

Etait-ce à cause du pressentiment que ce serait bigrement utile ? Toujours est-il que Microsoft a mis au point une syntaxe permettant de réserver ou d'exclure une partie de code XHTML pour une ou plusieurs versions d'Internet Explorer.

La voici sous une forme adaptée par Laurent Denis pour qu'elle corresponde à du code XHTML valide. Elle est disponible sur la page http://www.blog-and-blues.org/articles/Les_syntaxes_de_commentaires_conditionnels_pour_IE_Windows.

Code réservé à une ou plusieurs versions d'Internet Explorer

```
<!--[if lte IE 6]> 1
  <link rel="stylesheet" type="text/css"
                         href="style-ie6.css" /> 2
<![endif]--> 3
```

La condition 1 signifie : « Si le navigateur est une version d'Internet Explorer égale ou antérieure à la version 6 (soit IE 6, IE 5.5, IE 5, ...) ».

L'expression « lte IE6 » veut dire « less than or equal », c'est-à-dire *inférieur ou égal à IE 6*.

Le code XHTML 2 n'est interprété que par les versions d'Internet Explorer indiquées dans la condition 1. Il est ignoré par les navigateurs autres qu'Internet Explorer, en raison de la présence de marques de commentaires `<!--` et `-->` sur les lignes 1 et 3.

Dans le cas présent, l'appel de la feuille de style `style-ie6.css` par la balise `link` 2 ne sera pris en compte que par Internet Explorer dans ses versions 6 et antérieures. Internet Explorer 7 ou 8 ainsi que les autres navigateurs ne liront pas cette ligne, donc cette feuille de style ne leur sera pas appliquée.

Pour donner un autre exemple, un contenu XHTML sera interprété uniquement par Internet Explorer 7 si la ligne 1 est remplacée par :

```
<!--[if IE 7]>
```

D'une manière générale, la condition entre crochets s'écrit de la façon montrée par les exemples suivants :

condition	signification	traduction
`IE`		toute version d'Internet Explorer
`IE7`		égal à IE 7
`lt IE 7`	less than	antérieur à IE 7
`lte IE 6`	less than or equal	antérieur ou égal à IE 6
`gte IE 6`	greater than or equal	postérieur ou égal à IE 6
`gt IE 6`	greater than	postérieur à IE 6

Code ignoré par une ou plusieurs versions d'Internet Explorer

```
<!--[if !IE]> <--> 1
   <link rel="stylesheet" type="text/css"
                          href="style-sauf-ie.css" /> 2
<!--> <![endif]--> 3
```

Notez bien que si c'est le *point d'exclamation* qui entraîne la négation de la condition, il faut en même temps quitter le mode *commentaires* pour que le code XHTML soit bien pris en compte par les autres navigateurs, et y revenir pour écrire le code de fin `endif` spécifique à Internet Explorer. C'est pourquoi les lignes 1 et 3 sont différentes du cas précédent.

Dans l'exemple ci-dessus, la balise `link` 2 et donc la feuille de style `style-sauf-ie.css` seront ignorées par toutes les versions d'Internet Explorer mais prises en compte par les autres navigateurs.

Prenons un nouvel exemple, dans lequel le code XHTML ne doit *pas* être interprété par les versions Internet Explorer 6 et antérieures. Il faut alors écrire la ligne 1 de la façon suivante :

```
<!--[if !lte IE 6]> <--> 1
```

Ces méthodes permettent d'affecter des feuilles de style spécifiques aux navigateurs qui ont un comportement hors normes.

Lorsqu'il y a peu de règles de style à modifier, il n'est pas nécessaire de créer des feuilles CSS différentes : il suffit d'adapter les règles aux navigateurs, comme nous allons le voir maintenant.

Règles de style en fonction des navigateurs

Vous l'aurez deviné, les « certains navigateurs » pour lesquels il faut adopter des règles de style spécifiques sont Internet Explorer 6 et 7, en raison de leur interprétation erronée de certains standards CSS.

À l'opposé et toujours pour la même raison, il sera pratique de pouvoir écrire des règles de style qui seront interprétées par tous les navigateurs, sauf Internet Explorer 6 ou 7.

Ce sont les *sélecteurs* des règles de style qui nous permettront d'effectuer ce distinguo.

Règles de style pour Internet Explorer 6 et versions antérieures

Cette méthode permet d'écrire entre accolades un ensemble de propriétés CSS qui ne seront prises en compte que par IE 6 et qui seront ignorées par IE 7 ainsi que par tous les autres navigateurs : Firefox, Opera, Safari, Konqueror, ...

Exemple

```
* html p { ... propriétés de style ... }
```

L'astérisque * représente n'importe quelle balise. Cette règle s'adresse donc à une balise (ici <p>) incluse dans une balise <html>, elle-même incluse dans une balise quelconque *.

Or, la balise <html> étant la première de la page, elle ne peut pas être incluse dans une autre. Cette règle n'est donc jamais interprétée, sauf par Internet Explorer 6 et versions inférieures, qui ne tiennent pas compte de cette restriction.

Règles de style pour Internet Explorer 7 seul

Sur certains points, Internet Explorer 7 respecte mieux les normes que la version 6, mais hélas il n'interprète pas correctement toutes les propriétés CSS. Voici une technique pour s'adresser à IE 7 en particulier.

Exemple

```
*:first-child+html div{ ... propriétés de style ... }
```

Courageux et inventif, celui qui a imaginé ce sélecteur ! J'ai trouvé celui-ci sur le site de David Hammond, http://www.webdevout.net/css-hacks. L'expliquer n'aurait guère d'intérêt (balise `html` juxtaposée au premier enfant de n'importe quelle balise...), il suffit de savoir que cette règle sera lue par IE 7, mais pas par IE versions 6 et antérieures, ni par les autres navigateurs.

> À noter **Règles de style pour Internet Explorer 6 et 7**
>
> Pour écrire des règles de style destinées aux versions Internet Explorer 7 et inférieures, il suffit d'utiliser les deux types de sélecteurs, soit en écrivant les deux lignes, soit en écrivant ces deux sélecteurs séparés par une virgule, comme le montre cet exemple :
> `* html h1, *:first-child+html h1{ styles}`

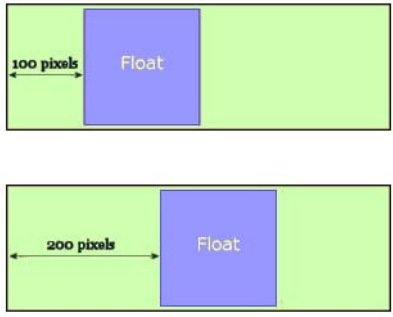


Figure 7–4 *La propriété margin attribuée à un bloc flottant est correctement prise en compte par Firefox (en haut), mais pas par Internet Explorer 6 (en bas). Extrait de la page http://cweben.wordpress.com/category/css/.*

Les deux techniques que nous venons d'étudier permettent de relier une règle de style à un type de navigateur, c'est-à-dire un ensemble de propriétés placées entre accolades.

Nous allons maintenant voir qu'il est possible de jouer sur l'écriture d'une seule propriété CSS pour l'affecter à Internet Explorer 6 ou 7.

Propriétés de style pour IE 6 ou 7

Les quelques astuces qui suivent permettent de gérer de façon plus souple l'adaptation de nos feuilles de style aux navigateurs récalcitrants aux normes que sont Internet Explorer 6 et 7.

Il s'agit d'écrire une propriété donnée de façon à ce qu'elle concerne seulement un de ces navigateurs ou bien les deux.

Exemple de propriété pour IE 6

```
_width: 120px;
```

Le caractère de soulignement accolé à la propriété la rend invalide pour la plupart des navigateurs, même pour IE 7. Par contre Internet Explorer versions 6 et antérieures prendront bien en compte cette propriété de style.

Exemple de propriété pour IE 6 et IE 7

```
*height: 100%;
```

Son nom étant précédé d'une étoile, la propriété de style ne sera pas interprétée par les navigateurs conformes, sauf Internet Explorer versions 7, 6 et antérieures qui prendront en compte cette propriété CSS, ignorant la présence de cette étoile.

Autre exemple de propriété spécifique à IE 6 et IE 7

```
width: expression(120 + "px");
```

Le mot-clé `expression(...)`, contenant entre parenthèses un assemblage ou un calcul en Javascript, est interprété par Internet Explorer seulement. C'est donc un équivalent de l'étoile accolée au nom de la propriété.

> Attention **Bye bye, la validation CSS !**
>
> Les styles écrits de cette façon, avec un caractère accolé au nom de la propriété de style ou utilisant une expression Javascript, sont parfois bien pratiques mais ils ne passent pas le test de validation CSS du W3C ! Pour conserver un code valide, mieux vaut alors utiliser ces méthodes de façon provisoire pendant la mise au point, puis revenir à une des méthodes précédentes pour le codage définitif de la page.

Après l'affectation de règles de style à l'attention de certaines versions d'Internet Explorer, voici des techniques pour réserver certains styles à ceux des navigateurs qui les interprètent sans problème.

Règles pour navigateurs modernes

Le terme « navigateurs modernes » englobe ceux qui respectent (depuis longtemps déjà...) les normes XHTML/CSS : Firefox, Opera, Konqueror, KHTML, Safari... Internet Explorer 7 les rejoindra ou non, en fonction de son comportement par rapport aux styles utilisés. Il nous faudra donc deux types d'astuces, avec ou sans IE 7.

Styles pour les navigateurs modernes et IE 7

Pour qu'une règle de style ne soit pas prise en compte par Internet Explorer 6 et versions inférieures, il suffit d'écrire par exemple :

```
html>body p { ... propriétés de style ..... }
```

La technique consiste à faire précéder le sélecteur (ici p) par `html>body`.

Toutes les balises d'une page sont dans <body> qui est toujours un enfant direct de la balise <html>. Par conséquent, cet ajout n'apporte aucune restriction au sélecteur : les navigateurs modernes lisent bien cette règle, ainsi qu'Internet Explorer 7.

Cependant, ce combinateur « enfant direct », noté `>`, n'est pas reconnu par Internet Explorer 6 et versions inférieures, qui ignorent donc la règle.

> ATTENTION **Pas d'espace autour du signe >**
>
> Il ne faut pas introduire d'espace autour du signe > dans l'expression `html>body`, sinon Internet Explorer 5 prendra en compte cette règle, sans pour autant l'interpréter mieux qu'IE 6...

Styles pour les navigateurs modernes sans IE 7

Voici une dernière astuce pour écrire des règles de style à destination des navigateurs modernes qui interprètent correctement les CSS, donc en excluant Internet Explorer 7 et les versions antérieures. Par exemple :

```
html>/**/body p { ... propriétés de style ..... }
```

Le sélecteur de cette règle ressemble à celui du cas précédent, à ceci près qu'il inclut des balises de commentaires sans aucun caractère à l'intérieur `/**/`. Elles sont comprises comme telles par les navigateurs modernes, qui vont lire sans problème cette règle de style. Par contre, elles perturbent IE 7 qui va donc ignorer la règle entière, de même que les versions 6 et antérieures d'Internet Explorer.

Après toutes ces astuces de sélecteurs et autres, il nous reste à uniformiser les marges de nos éléments, avant de détailler un certain nombre de techniques destinées à pallier certains défauts du plus « dur à cuire » des navigateurs actuellement utilisés, Internet Explorer 6.

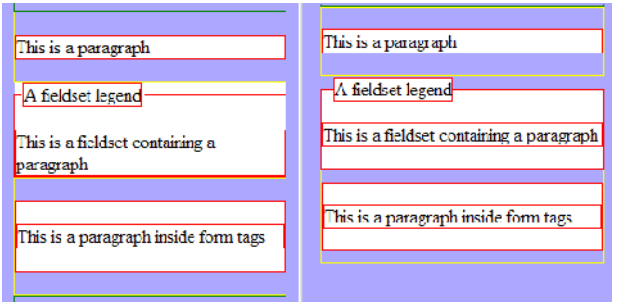


FIGURE 7–5 *La disposition des blocs est correcte sur le navigateur Opera (à droite), mais pas sur Internet Explorer 6 (à gauche). Extrait de l'image http://muffinresearch.co.uk/code/xhtmlandcss/defcss.gif.*

Marges par défaut

Les différents navigateurs ne proposent pas toujours les mêmes marges par défaut, qu'elles soient extérieures ou intérieures. Il est donc préférable de les annuler dès le début de la feuille de style, et de les régler ensuite pour chaque type de balise, en fonction des besoins.

La règle qui met à zéro les marges externes et internes pour tous les éléments de la page web est la suivante :

```
* { margin: 0; padding: 0; }
```

> **A noter** **Remise à zéro complète et ciblée**
>
> Eric Meyer propose une feuille de style plus complexe, qui réinitialise les attributs de nombreuses balises pour obtenir un comportement uniforme des différents navigateurs. Elle est disponible à l'adresse :
>
> - **http://meyerweb.com/eric/thoughts/2007/05/01/reset-reloaded/**
>
> En comparaison, notre étoile toute simple en guise de sélecteur et nos deux propriétés `margin` et `padding` paraissent bien rustiques ! Cependant, cette petite règle de style nous suffira largement dans un premier temps.

Règles spécifiques à Internet Explorer

Les pages suivantes de ce chapitre ont pour but de donner des solutions à certains problèmes parmi les plus courants posés par Internet Explorer 6.

Projet IE 7

Pour pallier les nombreuses lacunes d'Internet Explorer 6 (IE 6) dans la reconnaissance des standards CSS, Dean Edwards avait lancé le projet IE 7, disponible à l'adresse http://dean.edwards.name/IE7/.

> ATTENTION **Ne pas confondre IE 7 et IE 7 !**
>
> Le projet IE 7 est constitué d'un ensemble de programmes développés par un indépendant ; c'est un complément à appliquer aux pages web pour compenser les lacunes d'Internet Explorer 6. Il ne faut surtout pas confondre ce projet avec le successeur officiel d'IE 6, le navigateur Internet Explorer 7.

FIGURE 7–6 *Extrait de la page web de Dean Edwards consacrée au projet IE 7*

Le projet IE 7 comprend plusieurs fichiers et contient des fonctions Javascript qui permettent de compenser les dysfonctionnements d'Internet Explorer 6 ; il suffit de les placer dans un dossier et de les appeler dans l'en-tête de la page web. C'est une solution globale, qui alourdit le poids de la page mais a le mérite de régler quantité de défauts en une seule fois.

Cependant, lorsqu'il s'agit de régler un seul type de problème, il est possible d'utiliser un des « bidouillages » (*hacks* en anglais) qui suivent. Le « poids » de la page en octets est alors moins élevé.

Largeur ou hauteur minimum

Internet Explorer 6 ne reconnaît pas les propriétés **`min-width`** et **`min-height`**, alors qu'elles sont correctement interprétées à partir de la version 7 de ce navigateur.

Par ailleurs, IE 6 interprète mal les propriétés `width` et `height` : au lieu de comprendre largeur *fixe* ou hauteur *fixe*, il traduit largeur *minimum* ou hauteur *minimum*.

Il est alors possible de tirer profit de ce deuxième problème pour résoudre le premier :

- La largeur minimum s'écrit `min-width` selon la norme, et `width` pour Internet Explorer 6.
- La hauteur minimum s'exprime normalement par `min-height`, mais avec `height` pour Internet Explorer 6.

Les règles qui indiquent largeur ou hauteur minimum s'écrivent donc, par exemple :

```
div#menu { min-width: 25%; _width: 25%; }
div#titre { min-height: 100px; _height: 100px; }
```

Dans cet exemple, Internet Explorer ignorera les propriétés `min-width` et `min-height` et les remplacera par les propriétés `width` et `height`, auxquelles il donnera le même sens. Pour que seul ce navigateur interprète `width` et `height`, un caractère de soulignement est placé devant ces propriétés.

Position fixe

La propriété **`position: fixed`** qui permet de figer un élément sur l'écran, indépendamment du défilement, n'est pas reconnue par Internet Explorer 6. En revanche, les versions 7 et 8 la prennent bien en compte.

Technique de rattrapage de position

Une solution de remplacement consiste à calculer en permanence, à l'aide du langage Javascript, la position verticale que doit avoir l'élément concerné.

Le défilement modifie les paramètres utilisés dans cette instruction Javascript, ce qui entraîne donc un nouveau calcul de position.

Exemple

```
/* Règle CSS correcte : */
#logo { position: fixed; top: 20px; right: 10px; }

/* Règle pour Internet Explorer 6 */
#logo { _position: absolute;
         top: expression(body.scrollTop + 20 + "px"); }
```

Ici, la position du logo doit être à 20 pixels du haut de l'écran. Pour IE 6, le logo est en position absolue dans <body>, à une distance du haut égale à la hauteur de page cachée par le défilement + 20 pixels.

Stabilisation de l'affichage

Il reste encore un problème à régler : l'affichage du bloc fixe n'est pas très stable. En effet, lorsque la page défile, le rattrapage de la position du bloc se voit et ce dernier tremblote à l'écran.

Le remède est simple et efficace, mais il ne s'invente pas : il faut déclarer une image de fond dans <body> ! Cela peut être un carré transparent d'un pixel de côté ou encore plus simplement le mot `null` à la place du fichier image. Voici la règle à ajouter :

```
body { background: url(null) fixed ; }
```

> À NOTER **Limitation de cette technique**
>
> Cette méthode ne fonctionne pas avec les <!DOCTYPE...> officiels du W3C. Elle fonctionne sans <!DOCTYPE...>, ou avec un commentaire <!-- ... --> avant cette ligne, ce qui revient au même. Dans ce cas apparaissent d'autres problèmes avec IE 6, comme la marge interne `padding` qui n'est pas reconnue lorsqu'elle s'applique à des images. Nous atteignons ici les limites du bricolage, puisque le code XHTML n'est plus valide.

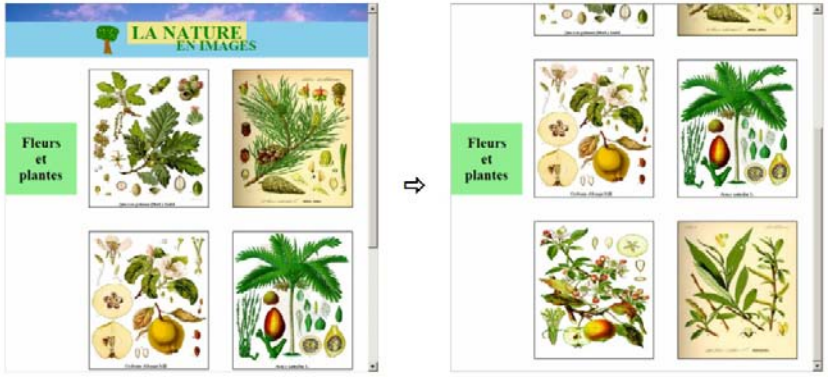


Figure 7–7 *L'exemple vu précédemment, avec un titre fixe sur l'écran, devra utiliser la solution spécifique à Internet Explorer, en plus de la propriété* `position: fixed`.

Espace vertical sous une image

Dans Internet Explorer, il arrive qu'une image soit suivie d'un espace en dessous, que la mise à zéro de tous les `margin` et `padding` du monde n'arrivera pas à enlever.

Pour supprimer cet espacement, il faut transformer l'image en bloc, à l'aide de la propriété :

```
display: block;
```

> À noter **Autres méthodes**
>
> Il existe deux autres techniques, cousines entre elles, pour supprimer cet espacement vertical après une image dans Internet Explorer :
> - supprimez tous les espaces et sauts de ligne entre la fin de la balise image et la balise qui la suit ;
> - ou bien encadrez l'image par un bloc `<div>...</div>`, sans espace ni saut de ligne entre la balise image et `</div>`.

FIGURE 7–8 *En haut, l'image des nuages est suivie d'un espace dans Internet Explorer ; en bas, le problème a été résolu par la propriété* `display: block`.

Transparence des images PNG

Contrairement à Firefox, Opera ou à ses propres successeurs Internet Explorer 7 et 8, Internet Explorer 6 n'affiche pas les niveaux de transparence « progressive » (que l'on appelle canal alpha) des images PNG. Il les remplace par un fond gris.

En revanche, il lit bien les images GIF, qui n'ont qu'un seul niveau de transparence possible.

Affichage d'une image PNG transparente avec Internet Explorer 6

L'affichage de PNG transparents avec IE 6 seul s'obtient :

- en utilisant une image GIF transparente de 1 pixel de côté (fichier à créer, appelé ici `blank.gif`) ;
- et en appelant un filtre Microsoft.

Exemple pour IE 6

```
<img src="blank.gif"
  style=" width: 150px; height: 100px;
    filter:progid:
    DXImageTransform.Microsoft.AlphaImageLoader(
        src='exemple.png', sizingMethod='scale');" />
```

L'image officiellement affichée est l'image GIF, qui n'est qu'un malheureux pixel transparent.

L'image qui nous intéresse, avec plusieurs niveaux de transparence, s'appelle ici `exemple.png`.

Dans la formule barbare qui invoque le filtre Microsoft, le nom `AlphaImageLoader` fait référence au canal alpha des niveaux de transparence.

Affichage d'une image PNG transparente sur tous les navigateurs

La formule précédente n'affiche l'image PNG que sur Internet Explorer. Pour qu'elle s'affiche sur tous les autres navigateurs, une astuce consiste à ajouter l'image PNG comme image de fond de la balise `<img ... />`, avec :

```
style=" background-image: url(exemple.png);
        _background-image: none;"
```

La deuxième propriété, qui commence par un caractère de soulignement, n'est interprétée que par Internet Explorer. Elle permet de masquer l'image de fond pour ce seul navigateur.

Exemple complet, pour tous les navigateurs

```
<img src="blank.gif"
  style=" background-image: url(exemple.png);
          _background-image: none;
          width: 150px; height: 100px;
    filter:progid:
    DXImageTransform.Microsoft.AlphaImageLoader(
        src='exemple.png',sizingMethod='scale');" />
```

> Attention **Dimensions de l'image**
>
> Dans la balise `<img ... />`, les dimensions **`width`** et **`height`** sont *obligatoires* et doivent absolument correspondre à celles du fichier image.

Dimensions d'affichage modifiées

Il est possible d'afficher l'image PNG avec une taille différente de celle qu'elle avait lors de son enregistrement, mais bien sûr, au prix d'une petite complication...

Était-ce nécessaire de compliquer la formule précédente ? La question mérite d'être posée ! Toutefois des lecteurs intrépides réclament cette solution inédite, et puis il est vrai qu'elle peut rendre service. Alors, la voici :

```
<img src="exemple.png"
  style=" _display: none;
          width: 150px; height: 100px;" />

<img src="blank.gif"
  style=" display: none; _display: inline;
          width: 150px; height: 100px;
    filter:progid:
    DXImageTransform.Microsoft.AlphaImageLoader(
        src='exemple.png',sizingMethod='scale');" />
```

La première balise affiche l'image PNG telle quelle, à destination des navigateurs qui reconnaîtront ses niveaux de transparence. Pour que cette image soit ignorée par Internet Explorer 6 seulement, l'astuce `_display: none` est utilisée.

La deuxième balise ressemble à celle de la première solution. Seulement, elle ne concerne plus maintenant qu'Internet Explorer 6, d'où les propriétés ajoutées : `display: none; _display: inline;`. La première supprime l'affichage de cette balise pour tous les navigateurs et la seconde la rétablit comme élément en ligne, mais uniquement pour Internet Explorer 6.

Depuis les sélecteurs en fonction des navigateurs jusqu'à l'affichage délicat des images PNG sur Internet Explorer 6, nous avons parcouru un certain nombre d'astuces et terminé ce dernier chapitre par des lignes de code bien compliquées !

Nous pourrons nous en affranchir avec joie lorsque ce navigateur IE 6 n'équipera plus qu'une partie négligeables des visiteurs de nos pages. Consultons de temps en temps les statistiques d'équipement des internautes !

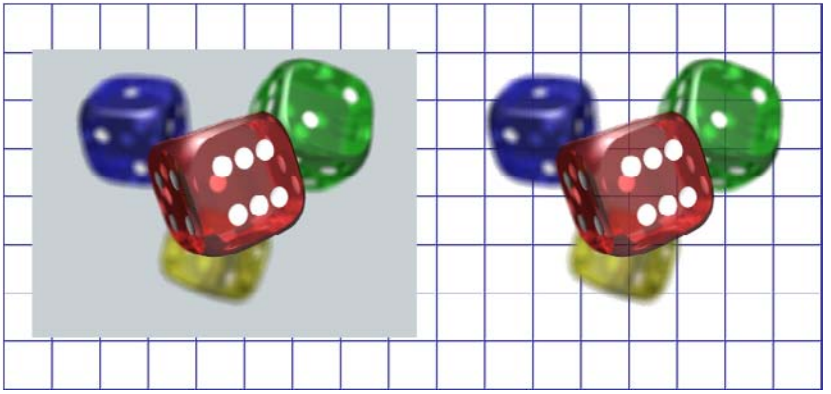
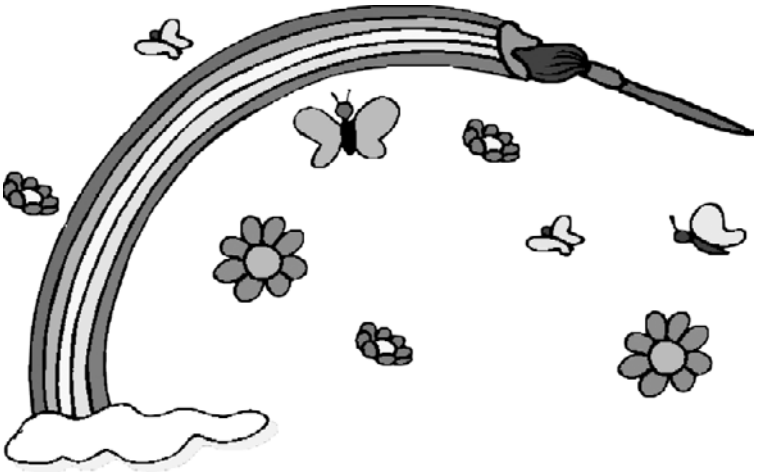


Figure 7–9 *Affichage par Internet Explorer 6 d'une image PNG avec plusieurs niveaux de transparence : avec la balise image normale, puis avec la solution spécifique à ce navigateur (image provenant du site http://www.wikipedia.fr).*

Ce livre se termine par des annexes dans lesquelles vous trouverez les codes et noms de couleurs, le taux de compréhension des balises XHTML et du CSS par différents navigateurs, ainsi qu'un résumé des propriétés CSS puis des références web et bibliographiques.

annexe

A



Couleurs

Dans cette annexe sont répertoriées, d'abord les 16 couleurs de base du HTML, et ensuite toutes les couleurs nommées.

Sommaire

« Des goûts et des couleurs, il ne faut point discuter »... Cependant, quels que soient les choix effectués, il faut ensuite les transcrire ! En général, les éditeurs HTML nous offrent la possibilité de choisir visuellement une couleur et affichent automatiquement le code correspondant.

Dans un premier temps, pour rester simple, un tableau nous donne les 16 couleurs de bases du HTML. Du classique, mais du solide !

Ensuite, après un petit détour par les « couleurs sûres », ceux qui sont allergiques aux codes numériques, qu'ils soient décimaux ou hexadécimaux, trouveront la liste complète des couleurs qui portent un nom.

Les 16 couleurs de base

Voici, classées par ordre alphabétique de leur nom en français, les 16 couleurs de base du HTML.

TABLEAU A–1 Les 16 couleurs de base du HTML

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Blanc	`white`	`#ffffff`	`rgb(255,255,255)`
Bleu	`blue`	`#0000ff`	`rgb(000,000,255)`
Bleu foncé	`navy`	`#000080`	`rgb(000,000,128)`
Bleu-vert	`teal`	`#008080`	`rgb(000,128,128)`
Cyan	`aqua`	`#00ffff`	`rgb(000,255,255)`
Gris clair	`silver`	`#c0c0c0`	`rgb(192,192,192)`
Gris foncé	`gray`	`#808080`	`rgb(128,128,128)`
Jaune	`yellow`	`#ffff00`	`rgb(255,255,000)`
Marron	`maroon`	`#800000`	`rgb(128,000,000)`
Noir	`black`	`#000000`	`rgb(000,000,000)`
Rose	`fuchsia`	`#ff00ff`	`rgb(255,000,255)`
Rouge	`red`	`#ff0000`	`rgb(255,000,000)`

Tableau A–1 Les 16 couleurs de base du HTML (suite)

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Vert	`green`	`#008000`	`rgb(000,128,000)`
Vert brillant	`lime`	`#00ff00`	`rgb(000,255,000)`
Vert olive	`olive`	`#808000`	`rgb(128,128,000)`
Violet	`purple`	`#800080`	`rgb(128,000,128)`

Rappel Code RVB

Le code RVB (Rouge - Vert - Bleu) ou RGB en anglais (Red - Green - Blue) consiste à fournir l'intensité de chacune de ces trois couleurs dans l'ordre, de trois façons possibles :

- soit en hexadécimal, chaque composante étant exprimée sur deux chiffres, compris entre `00` et `ff` ;
- soit en décimal, l'intensité de chaque couleur étant codée à l'aide de trois chiffres, compris entre `000` et `255`, avec la fonction `rgb(xx,xx,xx)` ;
- soit encore en pourcentage, puisque dans l'expression `rgb(xx,xx,xx)`, le code `xx` de chaque couleur peut être aussi un pourcentage compris entre `0%` et `100%`.

Couleurs sûres

Il existe une liste de 216 couleurs dites sûres (dont peu sont nommées), qui donnent le même résultat sur toutes les configurations, notamment celles qui sont limitées à 256 couleurs.

Par définition, une couleur est « sûre » si chacune de ses composantes RVB en hexadécimal vaut `00`, `33`, `66`, `99`, `cc`, ou `ff`.

Il était recommandé, il y a un certain nombre d'années, de ne choisir que parmi ces couleurs sûres pour ne pas avoir de surprise à l'affichage sur certaines configurations modestes.

Néanmoins, la technique a beaucoup évolué depuis et à présent, cette restriction de notre palette aux 216 couleurs sûres n'est plus nécessaire : les caracté-

ristiques des cartes graphiques de base (couleurs définies sur 16 ou 24 bits) permettent maintenant de profiter des 16 millions de couleurs disponibles.

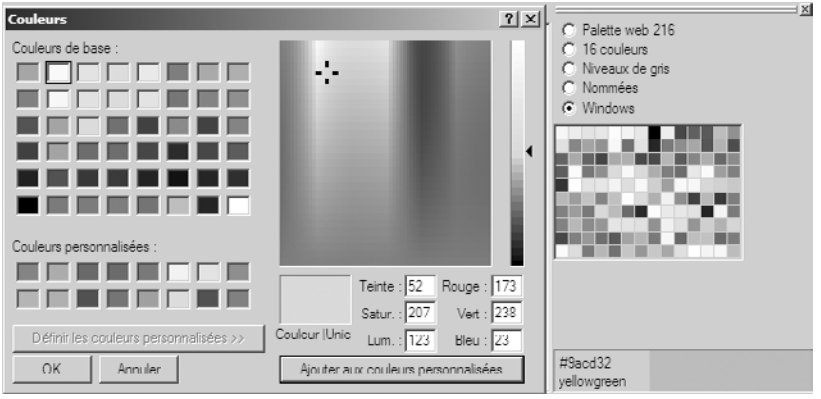


FIGURE A–1 *Un large choix de couleurs : copie d'écran du logiciel PsPad*

Liste de toutes les couleurs nommées

Sauf pour les couleurs simples ou fréquemment utilisées, le code RVB « Rouge - Vert - Bleu » hexadécimal n'est pas très parlant : à quoi ressemble la couleur `#adff2f` ? Même exprimée sous la forme `rgb(173,255,47)` ou encore `rgb(68%,100%,18%)`, cela ne nous dit pas grand-chose...

Une alternative plaisante consiste donc à utiliser les noms de couleurs prédéfinis, du moins pour celles qui en possèdent un. Pour reprendre l'exemple précédent, le nom `greenyellow` nous indique clairement qu'il s'agit d'un vert qui tire sur le jaune.

Le tableau suivant classe par teinte toutes les couleurs (X)HTML nommées. Il provient du travail très intéressant d'Alain Beyrand, webmestre du site http://www.pressibus.org. La page des couleurs est disponible à l'adresse : http://www.pressibus.org/perso/html/frcouleurs.html.

Pour voir les couleurs associées à ces noms, consultez ce site Internet ou essayez les en XHTML.

Tableau A–2 Couleurs nommées de ton BEIGE

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Beige	`beige`	`#f5f5dc`	`rgb(245,245,220)`
Beige blanc antique	`antiquewhite`	`#faebd7`	`rgb(250,235,215)`
Beige blanc Dalmond	`blanchedalmond`	`#ffebcd`	`rgb(255,235,205)`
Beige bisque	`bisque`	`#ffe4ba`	`rgb(255,228,186)`
Beige citron-soie	`lemonchiffon`	`#fffacd`	`rgb(255,250,205)`
Beige crème de papaye	`papayawhip`	`#ffefd5`	`rgb(255,239,213)`
Beige mocassin	`moccasin`	`#ffe4b5`	`rgb(255,228,181)`
Beige pêche	`peachpuff`	`#ffdab9`	`rgb(255,218,185)`

Tableau A–3 Couleurs nommées de ton BLANC

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Blanc	`white`	`#ffffff`	`rgb(255,255,255)`
Blanc coquillage	`seashell`	`#fff5ee`	`rgb(255,245,238)`
Blanc dentelle ancienne	`oldlace`	`#fdf5e6`	`rgb(253,245,230)`
Blanc fantôme	`ghostwhite`	`#f8f8ff`	`rgb(248,248,255)`
Blanc floral	`floralwhite`	`#fffaf0`	`rgb(255,250,240)`
Blanc ivoire	`ivory`	`#fffff0`	`rgb(255,255,240)`
Blanc fumée	`whitesmoke`	`#f5f5f5`	`rgb(245,245,245)`
Blanc lavande	`lavenderblush`	`#fff0f5`	`rgb(255,240,245)`
Blanc lin	`linen`	`#faf0e6`	`rgb(250,240,230)`
Blanc menthe	`mintcream`	`#f5fffa`	`rgb(245,255,250)`
Blanc neige	`snow`	`#fffafa`	`rgb(255,250,250)`

TABLEAU A–4 **Couleurs nommées de ton BLEU**

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Bleu	`blue`	`#0000ff`	`rgb(000,000,255)`
Bleu acier	`steelblue`	`#4582b4`	`rgb(070,130,180)`
Bleu acier clair	`lightsteelblue`	`#b0c4de`	`rgb(176,196,222)`
Bleu Alice	`aliceblue`	`#f0f8ff`	`rgb(240,248,255)`
Bleu ardoise	`slateblue`	`#6a5acd`	`rgb(106,090,205)`
Bleu ardoise foncé	`darkslateblue`	`#483d88`	`rgb(072,061,139)`
Bleu ardoise moyen	`mediumslateblue`	`#7b68ee`	`rgb(123,104,238)`
Bleu azur	`azure`	`#f0ffff`	`rgb(240,255,255)`
Bleu bleuet	`cornflowerblue`	`#6495ed`	`rgb(100,149,237)`
Bleu cadet	`cadetblue`	`#5f9ea0`	`rgb(095,158,160)`
Bleu ciel	`skyblue`	`#87cdeb`	`rgb(135,205,235)`
Bleu ciel clair	`lightskyblue`	`#87cefa`	`rgb(135,206,250)`
Bleu ciel profond	`deepskyblue`	`#00bfff`	`rgb(000,191,255)`
Bleu clair	`lightblue`	`#add8e6`	`rgb(173,216,230)`
Bleu foncé	`darkblue`	`#00008b`	`rgb(000,000,139)`
Bleu indigo	`indigo`	`#4b0082`	`rgb(075,000,130)`
Bleu lavande	`lavender`	`#e6e6fa`	`rgb(230,230,250)`
Bleu marin	`navy`	`#000080`	`rgb(000,000,128)`
Bleu de minuit	`midnightblue`	`#191970`	`rgb(025,025,112)`
Bleu moyen	`mediumblue`	`#0000cd`	`rgb(000,000,205)`
Bleu poudre	`powderblue`	`#b0e0e6`	`rgb(176,224,230)`
Bleu Pressibus	`pressibusblue`	`#000099`	`rgb(000,000,153)`
Bleu royal	`royalblue`	`#4169e1`	`rgb(065,105,225)`
Bleu toile	`dodgerblue`	`#1e90ff`	`rgb(030,144,255)`
Bleu violet	`blueviolet`	`#262be2`	`rgb(250,235,215)`

Tableau A–5 Couleurs nommées de ton BRUN

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Brun	brown	#a5292a	rgb(000,255,255)
Brun bois rustique	burlywood	#deb887	rgb(222,184,135)
Brun chocolat	chocolate	#d2691e	rgb(210,105,030)
Brun cuir	saddlebrown	#8b4513	rgb(139,069,019)
Brun kaki	khaki	#f0e68c	rgb(240,230,140)
Brun kaki foncé	darkkhaki	#bdb76b	rgb(189,183,107)
Brun marron	maroon	#800000	rgb(128,000,000)
Brun Pérou	peru	#cd8540	rgb(205,133,064)
Brun rosé	rosybrown	#bc8f8f	rgb(188,143,143)
Brun roux	tan	#d2b48c	rgb(210,180,140)
Brun sableux	sandybrown	#f4a460	rgb(244,164,096)
Brun terre de Sienne	sienna	#a0522d	rgb(160,082,045)

Tableau A–6 Couleurs nommées de ton CYAN - TURQUOISE

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Cyan	cyan	#00ffff	rgb(000,255,255)
Cyan clair	lightcyan	#e0ffff	rgb(224,255,255)
Cyan foncé	darkcyan	#008b8b	rgb(000,139,139)
Turquoise	turquoise	#40e0d0	rgb(064,224,208)
Turquoise foncé	darkturquoise	#00ced1	rgb(000,206,209)
Turquoise moyen	mediumturquoise	#48d1cc	rgb(072,209,204)
Turquoise pâle	paleturquoise	#afeeee	rgb(175,238,238)

TABLEAU A–7 Couleurs nommées de ton GRIS

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Gris	gray	#808080	rgb(128,128,128)
Gris ardoise	slategray	#708090	rgb(112,128,144)
Gris ardoise clair	lightslategray	#778899	rgb(119,136,153)
Gris ardoise foncé	darkslategray	#2f4f4f	rgb(047,079,079)
Gris argent	silver	#c0c0c0	rgb(192,192,192)
Gris clair	lightgrey	#d3d3d3	rgb(211,211,211)
Gris Gainsboro	gainsboro	#dcdcdc	rgb(220,220,220)
Gris mat	dimgray	#696969	rgb(105,105,105)

TABLEAU A–8 Couleur nommée NOIR et codes des NUANCES DE GRIS

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Noir	black	#000000	rgb(000,000,000)
(gris très foncé)		#333333	rgb(051,051,051)
(gris foncé)		#666666	rgb(102,102,102)
(gris moyen)		#999999	rgb(153,153,153)
(gris clair)		#cccccc	rgb(204,204,204)

TABLEAU A–9 Couleurs nommées de ton JAUNE

Nom en français	Nom HTML	Code hexa-décimal	Code décimal
Jaune	yellow	#ffff00	rgb(255,255,000)
Jaune blanc Navajo	navajowhite	#fffead	rgb(255,222,173)
Jaune blé	wheat	#f5deb3	rgb(245,222,179)
Jaune clair	lightyellow	#f4ffe0	rgb(244,255,224)
Jaune doré	goldenrod	#daa520	rgb(218,165,032)

Tableau A–9 Couleurs nommées de ton JAUNE (suite)

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Jaune doré clair	lightgoldenrodyellow	#fafad2	rgb(250,250,210)
Jaune doré foncé	darkgoldenrod	#b8840b	rgb(184,132,011)
Jaune doré pâle	palegoldenrod	#eee8aa	rgb(238,232,170)
Jaune or	gold	#ffff00	rgb(255,255,000)

Tableau A–10 Couleurs nommées de ton ORANGE - CORAIL - SAUMON

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Orange	orange	#ffa500	rgb(255,165,000)
Orange foncé	darkorange	#ff8c00	rgb(255,140,000)
Orangé	orangered	#ff4500	rgb(255,069,000)
Corail	coral	#ff7f50	rgb(255,127,080)
Corail clair	lightcoral	#f08080	rgb(240,128,128)
Saumon	salmon	#fa7872	rgb(250,120,114)
Saumon clair	lightsalmon	#ffa07a	rgb(255,160,122)
Saumon foncé	darksalmon	#e9967a	rgb(233,150,122)

Tableau A–11 Couleurs nommées de ton ROUGE

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Rouge	red	#ff0000	rgb(255,000,000)
Rouge brique	firebrick	#b22222	rgb(178,034,034)
Rouge cramoisi	crimson	#dc143c	rgb(220,020,060)
Rouge foncé	darkred	#8b0000	rgb(139,000,000)
Rouge indien	indianred	#cd5c5c	rgb(205,092,092)
Rouge tomate	tomato	#ff6347	rgb(255,099,071)

TABLEAU A–12 Couleurs nommées de ton ROSE

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Rose	pink	#ffc0cb	rgb(255,192,203)
Rose brumeux	mistyrose	#ffe4ff	rgb(255,228,255)
Rose clair	lightpink	#ffb6c1	rgb(255,182,193)
Rose passion	hotpink	#ff69b4	rgb(255,105,180)
Rose profond	deeppink	#ff1493	rgb(255,020,147)

TABLEAU A–13 Couleurs nommées de ton VIOLET - POURPRE - MAGENTA

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Violet	violet	#ee82ee	rgb(238,130,238)
Violet bourbon	cornsilk	#ff30dc	rgb(255,048,220)
Violet chardon	thistle	#d8bfd8	rgb(216,191,216)
Violet foncé	darkviolet	#9400d3	rgb(148,000,211)
Violet fuchsia	fuchsia	#ff00ff	rgb(000,206,209)
Violet moyen	mediumvioletred	#c71585	rgb(199,021,133)
Violet orchidée	orchid	#da70d6	rgb(218,112,214)
Violet orchidée foncé	darkorchid	#9932cc	rgb(153,050,204)
Violet orchidée moyen	mediumorchid	#ba55d3	rgb(186,085,211)
Violet pâle	palevioletred	#db7093	rgb(219,112,147)
Violet prune	plum	#dda0dd	rgb(221,160,221)
Pourpre	purple	#800080	rgb(128,000,128)
Pourpre moyen	mediumpurple	#9370db	rgb(147,112,219)
Magenta	magenta	#ff00ff	rgb(255,000,255)
Magenta foncé	darkmagenta	#8b008b	rgb(139,000,139)

Tableau A–14 Couleurs nommées de ton VERT

Nom en français	Nom HTML	Code hexadécimal	Code décimal
Vert	green	#008000	rgb(000,128,000)
Vert Chartreuse	chartreuse	#7fff00	rgb(127,255,000)
Vert clair	lightgreen	#90ee90	rgb(144,238,144)
Vert eau marine	aquamarine	#7fffd4	rgb(127,255,212)
Vert eau marine moyen	mediumaqua marine	#66cdaa	rgb(102,205,170)
Vert forestier	forestgreen	#228b22	rgb(034,139,034)
Vert foncé	darkgreen	#006400	rgb(000,100,000)
Vert jaune	greenyellow	#adff2f	rgb(173,255,047)
Vert jauni	yellowgreen	#9acd32	rgb(154,205,050)
Vert marin	seagreen	#2e8b57	rgb(046,139,087)
Vert marin clair	lightseagreen	#20b2aa	rgb(032,178,170)
Vert marin foncé	darkseagreen	#8fbc8f	rgb(143,188,143)
Vert marin moyen	mediumseagreen	#3cb371	rgb(060,179,113)
Vert olive	olive	#808000	rgb(128,128,000)
Vert olive grise	olivedrab	#6b8e23	rgb(107,142,035)
Vert olive foncé	darkolivegreen	#556b2f	rgb(085,107,047)
Vert pâle	palegreen	#98fb98	rgb(152,251,152)
Vert pelouse	lawngreen	#7cfc00	rgb(124,252,000)
Vert Pressibus	pressibusgreen	#99cc99	rgb(153,204,153)
Vert printanier	springgreen	#00ff7f	rgb(000,255,127)
Vert printanier moyen	mediumspring green	#00fa9a	rgb(000,250,154)
Vert sarcelle	teal	#008080	rgb(000,128,128)
Vert tilleul clair	lime	#00ff00	rgb(000,255,000)
Vert tilleul foncé	limegreen	#32cd32	rgb(050,205,050)

annexe

B

mozilla Firefox

Comportement des principaux navigateurs

Les balises XHTML et les propriétés CSS ne sont pas entièrement reconnues par les navigateurs. Voici en détail leur degré de prise en charge.

Sommaire

Même si nous parlons correctement une langue étrangère, il reste des mots qui nous échappent. Cette liste de mots incompris sera plus ou moins longue, en fonction de l'étendue de nos connaissances dans cette langue.

Eh bien, pour les navigateurs web, c'est pareil ! Globalement, il comprennent ce qu'il leur est demandé d'afficher, mais ils peuvent avoir des lacunes sur certains détails... Et quand ces incompréhensions touchent des points fondamentaux, il nous arrive de drôles de surprises !

FIGURE B–1 *Parfois, certains mots nous échappent ; de même, certaines normes sont mal comprises par les navigateurs web.*

Les tableaux qui suivent donnent les détails de la compréhension des normes par trois des principaux navigateurs utilisés actuellement : Internet Explorer versions 6 et 7, Firefox 2 et 3, Opera 9.

Ils sont le fruit de l'excellent et courageux travail de **David Hammond** (mentionnons son site : http://nanobox.chipx86.com) et peuvent être consultés à l'adresse : http://www.webdevout.net/browser_support.php. Encore ne s'agit-il que d'une synthèse, dont le détail est disponible sur le site www.webdevout.net.

Compréhension des balises HTML-XHTML

Ce premier tableau énumère les balises HTML-XHTML, avec leur niveau de compréhension par les différents navigateurs.

À part quelques soucis, notamment pour les tableaux et la balise <a>, les balises XHTML sont relativement bien comprises.

TABLEAU B–1 Compréhension des balises HTML-XHTML

Balises HTML-XHTML	IE 6	IE 7	Firefox 2	Firefox 3	Opera 9
`a`	72 %	72 %	88 %	88 %	83 %
`abbr`	50 %	88 %	97 %	97 %	84 %
`acronym`	88 %	88 %	97 %	97 %	84 %
`address`	88 %	88 %	97 %	97 %	91 %
`area`	88 %	88 %	96 %	96 %	93 %
`b`	88 %	88 %	97 %	97 %	91 %
`base`	83 %	83 %	83 %	83 %	83 %
`bdo`	88 %	88 %	97 %	97 %	91 %
`big`	88 %	88 %	97 %	97 %	91 %
`blockquote`	80 %	80 %	98 %	98 %	83 %
`body`	92 %	92 %	98 %	98 %	94 %
`br`	88 %	88 %	94 %	94 %	94 %
`button`	82 %	82 %	99 %	99 %	93 %
`caption`	88 %	88 %	97 %	97 %	91 %
`cite`	88 %	88 %	97 %	97 %	91 %
`code`	88 %	88 %	97 %	97 %	91 %
`col`	68 %	68 %	75 %	75 %	70 %
`colgroup`	68 %	68 %	75 %	75 %	70 %
`dd`	88 %	88 %	97 %	97 %	91 %
`del`	75 %	75 %	98 %	98 %	77 %
`dfn`	88 %	88 %	97 %	97 %	91 %

TABLEAU B–1 Compréhension des balises HTML-XHTML (suite)

Balises HTML-XHTML	IE 6	IE 7	Firefox 2	Firefox 3	Opera 9
`div`	88 %	88 %	97 %	97 %	91 %
`dl`	88 %	88 %	97 %	97 %	91 %
`dt`	88 %	88 %	97 %	97 %	91 %
`em`	88 %	88 %	97 %	97 %	91 %
`fieldset`	88 %	88 %	97 %	97 %	91 %
`form`	85 %	85 %	95 %	95 %	93 %
`frame`	85 %	85 %	88 %	88 %	88 %
`frameset`	96 %	96 %	98 %	98 %	92 %
`h1`	88 %	88 %	97 %	97 %	91 %
`h2`	88 %	88 %	97 %	97 %	91 %
`h3`	88 %	88 %	97 %	97 %	91 %
`h4`	88 %	88 %	97 %	97 %	91 %
`h5`	88 %	88 %	97 %	97 %	91 %
`h6`	88 %	88 %	97 %	97 %	91 %
`head`	67 %	67 %	83 %	83 %	83 %
`hr`	88 %	88 %	97 %	97 %	91 %
`html`	88 %	88 %	100 %	100 %	88 %
`i`	88 %	88 %	97 %	97 %	91 %
`iframe`	89 %	89 %	93 %	93 %	93 %
`img`	85 %	85 %	99 %	99 %	91 %
`input`	85 %	85 %	89 %	89 %	86 %
`ins`	75 %	75 %	98 %	98 %	77 %
`kbd`	88 %	88 %	97 %	97 %	91 %
`label`	75 %	81 %	86 %	86 %	95 %
`legend`	90 %	90 %	98 %	98 %	83 %
`li`	88 %	88 %	97 %	97 %	91 %
`link`	75 %	75 %	93 %	93 %	80 %

TABLEAU B–1 Compréhension des balises HTML-XHTML (suite)

Balises HTML-XHTML	IE 6	IE 7	Firefox 2	Firefox 3	Opera 9
`map`	80 %	80 %	98 %	98 %	93 %
`meta`	96 %	96 %	96 %	96 %	96 %
`noframes`	50 %	50 %	97 %	97 %	88 %
`noscript`	63 %	63 %	97 %	97 %	75 %
`object`	69 %	70 %	85 %	85 %	88 %
`ol`	88 %	88 %	97 %	97 %	91 %
`optgroup`	69 %	71 %	93 %	93 %	78 %
`option`	70 %	78 %	82 %	82 %	77 %
`p`	88 %	88 %	97 %	97 %	91 %
`param`	92 %	92 %	100 %	100 %	100 %
`pre`	88 %	88 %	97 %	97 %	91 %
`q`	70 %	70 %	98 %	98 %	83 %
`samp`	88 %	88 %	97 %	97 %	91 %
`script`	100 %	100 %	90 %	90 %	90 %
`select`	86 %	88 %	99 %	99 %	89 %
`small`	88 %	88 %	97 %	97 %	91 %
`span`	88 %	88 %	97 %	97 %	91 %
`strong`	88 %	88 %	97 %	97 %	91 %
`style`	85 %	85 %	90 %	90 %	85 %
`sub`	88 %	88 %	97 %	97 %	91 %
`sup`	88 %	88 %	97 %	97 %	91 %
`table`	90 %	90 %	94 %	94 %	92 %
`tbody`	77 %	77 %	91 %	91 %	86 %
`td`	62 %	62 %	78 %	78 %	71 %
`textarea`	90 %	90 %	99 %	99 %	97 %
`tfoot`	77 %	77 %	91 %	91 %	86 %
`th`	62 %	62 %	78 %	78 %	71 %

TABLEAU B–1 Compréhension des balises HTML-XHTML (suite)

Balises HTML-XHTML	IE 6	IE 7	Firefox 2	Firefox 3	Opera 9
`thead`	77 %	77 %	91 %	91 %	86 %
`title`	88 %	88 %	88 %	88 %	88 %
`tr`	77 %	77 %	91 %	91 %	86 %
`tt`	88 %	88 %	97 %	97 %	91 %
`ul`	88 %	88 %	97 %	97 %	91 %
`var`	88 %	88 %	97 %	97 %	91 %
alignement des cellules	37 %	37 %	68 %	68 %	68 %

Interprétation des propriétés CSS 2.1

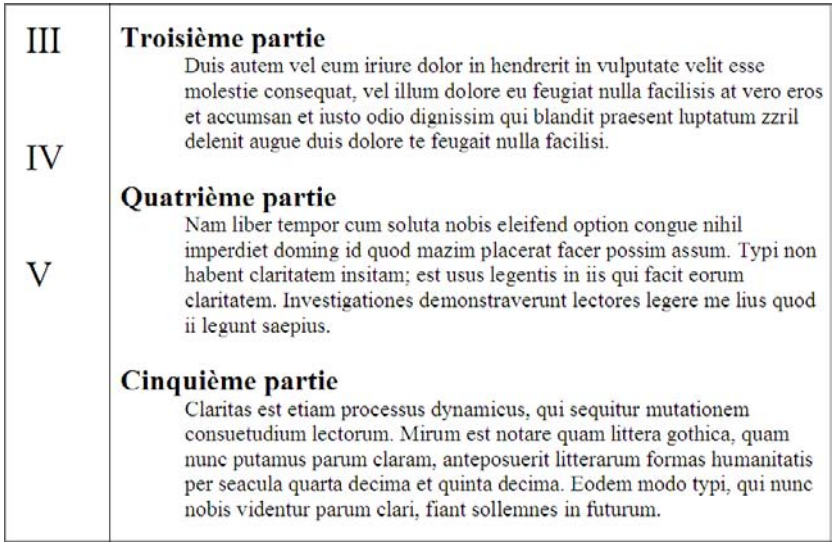


FIGURE B–2 *Certaines propriétés sont mal prises en compte, voire complètement ignorées par certains navigateurs. Il en résulte parfois un grand désordre dans la mise en page.*

Comme le montrent les tableaux qui suivent, plusieurs propriétés sont assez mal comprises, voire ignorées, par un ou plusieurs des navigateurs courants. Il est donc utile, avant d'utiliser une propriété, de savoir quelle chance elle a d'être correctement interprétée.

Unités

TABLEAU B–2 Prise en compte des unités CSS 2.1

Caractéristiques CSS 2.1	IE 6	IE 7	Firefox 2	Firefox 3	Opera 9
couleur	99 %	99 %	oui	oui	97 %
compteur	non	non	oui	oui	incomplet
entier	oui	oui	oui	oui	oui
longueur	oui	oui	oui	oui	oui
nombre	oui	oui	oui	oui	oui
pourcentage	oui	oui	oui	oui	oui
caractères	non	non	oui	oui	oui
URI	oui	oui	oui	oui	oui

Paramètre !important

TABLEAU B–3 Interprétation du paramètre !important

Caractéristiques CSS 2.1	IE 6	IE 7	Firefox 2	Firefox 3	Opera 9
`!important`	incomplet	incomplet	oui	oui	oui

Médias

TABLEAU B–4 Compréhension des différents types de média

Caractéristiques CSS 2.1	IE 6	IE 7	Firefox 2	Firefox 2	Opera 9
`@charset`	incomplet	incomplet	oui	oui	oui
`@import`	incomplet	incomplet	oui	oui	oui

TABLEAU B–4 Compréhension des différents types de média (suite)

Caractéristiques CSS 2.1	IE 6	IE 7	Firefox 2	Firefox 2	Opera 9
`@media`	incomplet	incomplet	oui	oui	oui
`@page`	non	non	non	non	oui

Sélecteurs

TABLEAU B–5 Prise en compte des sélecteurs CSS 2.1

Caractéristiques CSS 2.1	IE 6	IE 7	Firefox 2	Firefox 3	Opera 9
`*`	incomplet	incomplet	oui	oui	incomplet
`E`	incomplet	incomplet	oui	oui	oui
`E F`	incomplet	incomplet	oui	oui	oui
`E > F`	non	incomplet	oui	oui	oui
`E + F`	non	incomplet	oui	oui	incomplet
`[attr]`	non	incomplet	oui	oui	oui
`[attr="valeur"]`	non	incomplet	incomplet	incomplet	incomplet
`[attr~="valeur"]`	non	incomplet	incomplet	incomplet	incomplet
`[attr|="valeur"]`	non	incomplet	incomplet	incomplet	incomplet
`.class`	incomplet	oui	oui	oui	oui
`#id`	incomplet	oui	oui	oui	oui

Pseudo-classes

TABLEAU B–6 Interprétation des pseudo-classes CSS 2.1

Caractéristiques CSS 2.1	IE 6	IE 7	Firefox 2	Firefox 3	Opera 9
`:active`	incomplet	incomplet	oui	oui	oui
`:first-child`	non	incomplet	incomplet	oui	incomplet

TABLEAU B–6 **Interprétation des pseudo-classes CSS 2.1 (suite)**

Caractéristiques CSS 2.1	IE 6	IE 7	Firefox 2	Firefox 3	Opera 9
`:focus`	non	non	oui	oui	oui
`:hover`	incomplet	incomplet	oui	oui	oui
`:lang(C)`	non	non	oui	oui	oui
`:link`	incomplet	incomplet	oui	oui	oui
`:visited`	incomplet	incomplet	oui	oui	oui

Pseudo-éléments

TABLEAU B–7 **Compréhension des pseudo-éléments CSS 2.1**

Caractéristiques CSS 2.1	IE 6	IE 7	Firefox 2	Firefox 3	Opera 9
`:after`	non	non	incomplet	incomplet	incomplet
`:before`	non	non	incomplet	incomplet	incomplet
`:first-letter`	incomplet	incomplet	incomplet	incomplet	incomplet
`:first-line`	incomplet	incomplet	oui	oui	oui

Propriétés

TABLEAU B–8 **Prise en compte des propriétés CSS 2.1**

Caractéristiques CSS 2.1	IE 6	IE 7	Firefox 2	Firefox 3	Opera 9
`background`	56 %	58 %	oui	oui	83 %
`background-attachment`	38 %	50 %	oui	oui	88 %
`background-color`	62 %	62 %	oui	oui	87 %
`background-image`	63 %	63 %	oui	oui	88 %
`background-position`	45 %	45 %	oui	oui	95 %
`background-repeat`	75 %	75 %	oui	oui	92 %

TABLEAU B–8 Prise en compte des propriétés CSS 2.1 (suite)

Caractéristiques CSS 2.1	IE 6	IE 7	Firefox 2	Firefox 3	Opera 9
`border`	58 %	61 %	oui	oui	100 %
`border-bottom`	58 %	61 %	oui	oui	100 %
`border-bottom-color`	50 %	62 %	oui	oui	99 %
`border-bottom-style`	75 %	75 %	oui	oui	oui
`border-bottom-width`	75 %	75 %	oui	oui	oui
`border-collapse`	50 %	50 %	oui	oui	oui
`border-color`	50 %	62 %	oui	oui	99 %
`border-left`	58 %	61 %	oui	oui	100 %
`border-left-color`	50 %	62 %	oui	oui	99 %
`border-left-style`	75 %	75 %	oui	oui	oui
`border-left-width`	75 %	75 %	oui	oui	oui
`border-right`	58 %	61 %	oui	oui	100 %
`border-right-color`	50 %	62 %	oui	oui	99 %
`border-right-style`	75 %	75 %	oui	oui	oui
`border-right-width`	75 %	75 %	oui	oui	oui
`border-spacing`	non	non	oui	oui	oui
`border-style`	75 %	75 %	oui	oui	oui
`border-top`	58 %	61 %	oui	oui	100 %
`border-top-color`	50 %	62 %	oui	oui	99 %
`border-top-style`	75 %	75 %	oui	oui	oui
`border-top-width`	75 %	75 %	oui	oui	oui
`border-width`	75 %	75 %	oui	oui	oui
`bottom`	70 %	70 %	oui	oui	90 %
`caption-side`	non	non	oui	oui	oui
`clear`	75 %	50 %	oui	oui	oui
`clip`	non	non	oui	oui	oui
`color`	50 %	50 %	oui	oui	99 %
`content`	non	non	93 %	oui	80 %

TABLEAU B–8 **Prise en compte des propriétés CSS 2.1 (suite)**

Caractéristiques CSS 2.1	IE 6	IE 7	Firefox 2	Firefox 3	Opera 9
`counter-increment`	non	non	oui	oui	90 %
`counter-reset`	non	non	oui	oui	oui
`cursor`	93 %	93 %	oui	oui	93 %
`direction`	88 %	88 %	oui	oui	oui
`display`	31 %	31 %	81 %	92 %	94 %
`empty-cells`	non	non	88 %	88 %	88 %
`float`	50 %	50 %	80 %	80 %	80 %
`font`	89 %	89 %	oui	oui	oui
`font-family`	81 %	81 %	oui	oui	oui
`font-size`	88 %	88 %	oui	oui	oui
`font-style`	70 %	70 %	oui	oui	oui
`font-variant`	50 %	50 %	oui	oui	oui
`font-weight`	90 %	90 %	oui	oui	oui
`height`	50 %	50 %	oui	oui	90 %
`left`	60 %	60 %	oui	oui	oui
`letter-spacing`	63 %	63 %	oui	oui	oui
`line-height`	75 %	75 %	oui	oui	oui
`list-style`	67 %	67 %	oui	oui	oui
`list-style-image`	63 %	63 %	oui	oui	oui
`list-style-position`	63 %	63 %	oui	oui	oui
`list-style-type`	56 %	56 %	oui	oui	oui
`margin`	50 %	60 %	oui	oui	oui
`margin-bottom`	50 %	50 %	oui	oui	oui
`margin-left`	50 %	60 %	oui	oui	oui
`margin-right`	50 %	60 %	oui	oui	oui
`margin-top`	50 %	50 %	oui	oui	oui
`max-height`	non	50 %	oui	oui	oui
`max-width`	non	50 %	oui	oui	oui

TABLEAU B–8 Prise en compte des propriétés CSS 2.1 (suite)

Caractéristiques CSS 2.1	IE 6	IE 7	Firefox 2	Firefox 3	Opera 9
`min-height`	non	38 %	oui	oui	75 %
`min-width`	non	38 %	oui	oui	88 %
`outline`	non	non	oui	oui	100 %
`outline-color`	non	non	oui	oui	99 %
`outline-style`	non	non	oui	oui	oui
`outline-width`	non	non	oui	oui	oui
`overflow`	42 %	50 %	92 %	92 %	oui
`padding`	50 %	63 %	oui	oui	oui
`padding-bottom`	50 %	63 %	oui	oui	oui
`padding-left`	50 %	63 %	oui	oui	oui
`padding-right`	50 %	63 %	oui	oui	oui
`padding-top`	50 %	63 %	oui	oui	oui
`position`	42 %	50 %	oui	oui	oui
`quotes`	non	non	oui	oui	88 %
`right`	70 %	70 %	oui	oui	90 %
`table-layout`	63 %	63 %	oui	oui	oui
`text-align`	75 %	75 %	oui	oui	oui
`text-decoration`	64 %	64 %	93 %	93 %	oui
`text-indent`	63 %	63 %	oui	oui	oui
`text-transform`	58 %	58 %	oui	oui	oui
`top`	70 %	70 %	oui	oui	oui
`unicode-bidi`	70 %	70 %	oui	oui	oui
`vertical-align`	54 %	54 %	oui	oui	oui
`visibility`	70 %	70 %	oui	oui	90 %
`white-space`	29 %	29 %	oui	oui	90 %
`width`	40 %	50 %	oui	oui	90 %
`word-spacing`	63 %	63 %	oui	oui	oui
`z-index`	63 %	63 %	88 %	oui	oui

Paramètres d'impression

TABLEAU B–9 Compréhension des paramètres d'impression CSS 2.1

Caractéristiques CSS 2.1	IE 6	IE 7	Firefox 2	Firefox 2	Opera 9
`orphans`	non	non	non	non	oui
`page-break-after`	64 %	64 %	57 %	57 %	oui
`page-break-before`	64 %	64 %	57 %	57 %	oui
`page-break-inside`	non	non	non	non	oui
`widows`	non	non	non	non	oui

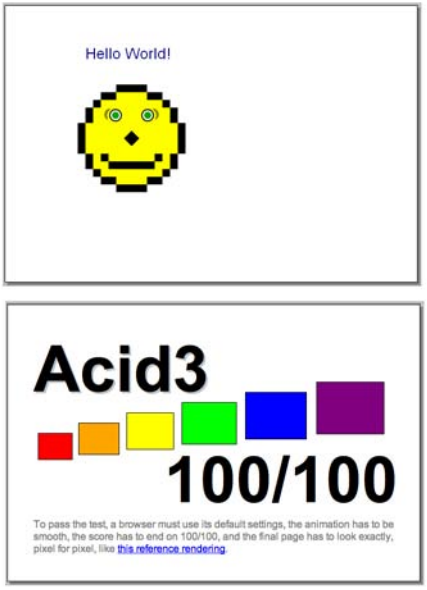


FIGURE B–3 *Créé pour inciter les développeurs à mieux respecter les normes du Web, le test Acid2 (http://acid2.acidtests.org) est une page web qui doit fournir l'image du haut ; les plus modernes des navigateurs affichent cette page correctement. L'objectif est à présent le test Acid3 (http://acid3.acidtests.org) : si le navigateur passe ce test, il affiche une animation qui se termine par l'image du bas.*

annexe

C

Résumé des propriétés CSS 2

Voici quelques pages qui pourront servir de référence, puisqu'elles résument les caractéristiques principales de chaque propriété.

Sommaire

Une fois habitué à l'utilisation des feuilles de style, vous aurez parfois besoin d'un petit rappel sur une propriété. Alors, voici un index bien pratique pour vous rafraîchir la mémoire.

Propriétés CSS 2

Ces tableaux ont été réalisés d'après une page du site www.yoyodesign.org, qui propose la traduction en français des normes du W3C, le World Wide Web Consortium : http://www.yoyodesign.org/doc/w3c/css2/propidx.html. La page web originale en anglais contenant ce tableau se trouve à l'adresse http://www.w3.org/TR/REC-CSS2/propidx.html.

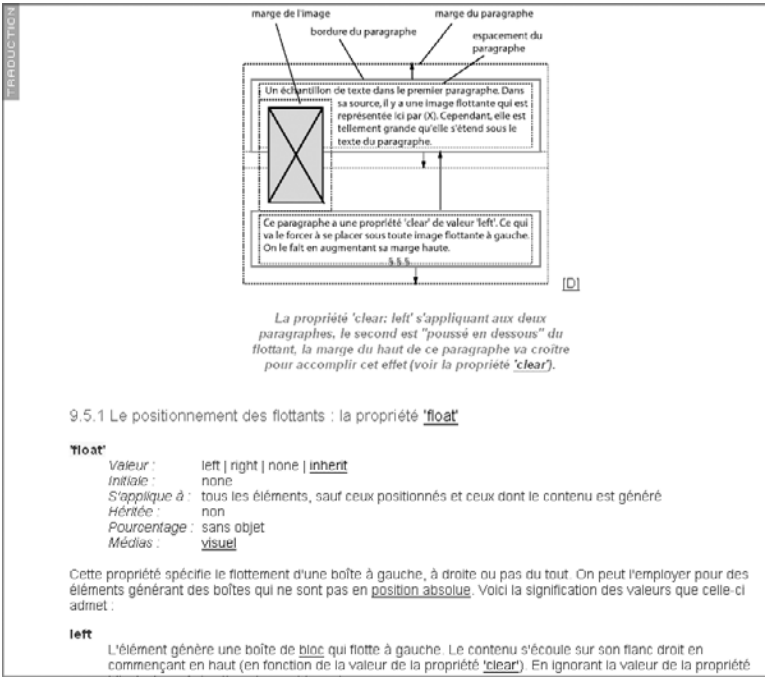


FIGURE C–1 *Pour plus de détails sur une propriété CSS, n'hésitez pas à consulter les documents du W3C, en français sur le site http://www.yoyodesign.org (table des matières à l'adresse : http://www.yoyodesign.org/doc/w3c/css2/cover.html).*

Propriétés d'affichage

TABLEAU C–1 Index des propriétés d'affichage CSS 2

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
`background`	fond	[background-color || background-image || background-repeat || background-attachment || background-position] | inherit	voir chaque propriété		admis pour background-position
`background-attachment`	défilement de l'image de fond	scroll | fixed | inherit	scroll		
`background-color`	couleur de fond	<couleur> | transparent | inherit	transparent		
`background-image`	image de fond	<uri> | none | inherit	none		
`background-position`	position de l'image de fond	[[<pourcentage> | <longueur>]{1,2} | [[top | center | bottom] || [left | center | right]]] | inherit	0 % 0 %	éléments de type bloc et éléments remplacés	% de la taille de la boîte elle-même
`background-repeat`	répétition de l'image de fond	repeat | repeat-x | repeat-y | no-repeat | inherit	repeat		
`border`	raccourci pour les bordures	[border-width || border-style || <couleur>] | inherit	voir chaque propriété		

TABLEAU C–1 Index des propriétés d'affichage CSS 2 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
`border-collapse`	fusion des bordures	collapse | separate | inherit (H)	collapse	éléments avec 'display: table' ou 'display: inline-table'	
`border-color`	couleur des bordures	<couleur>{1,4} | transparent | inherit	voir chaque propriété		
`border-spacing`	espace entre les bordures	<longueur> <longueur>? | inherit (H)	0	éléments avec 'display: table' ou 'display: inline-table'	
`border-style`	style de bordure	<bordure-style>{1,4} | inherit	voir chaque propriété		

`border-top`
`border-right`
`border-bottom`
`border-left`
	bordures sur les côtés	[border-top-width || border-style || <couleur>] | inherit	voir chaque propriété		

TABLEAU C–1 Index des propriétés d'affichage CSS 2 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)

`border-top-color`
`border-right-color`
`border-bottom-color`
`border-left-color`
	couleur des bordures sur les côtés	<couleur> | inherit	la valeur de la propriété 'color'		

`border-top-style`
`border-right-style`
`border-bottom-style`
`border-left-style`
	style de bordure sur les côtés	<bordure-style> | inherit	none		

`border-top-width`
`border-right-width`
`border-bottom-width`
`border-left-width`
	épaisseur des bordures sur les côtés	<bordure-épaisseur> | inherit	medium		

`border-width`	épaisseur des bordures	<bordure-épaisseur>{1,4} | inherit	voir chaque propriété		
`bottom`	position par rapport au bas	<longueur> | <pourcentage> | auto | inherit	auto	éléments positionnés	% de la hauteur du conteneur

TABLEAU C–1 Index des propriétés d'affichage CSS 2 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
`caption-side`	emplacement du titre (tableau)	top | bottom | left | right | inherit (H)	top	éléments avec 'display: table-caption'	
`clear`	pas de boîtes flottantes à côté	none | left | right | both | inherit	none	éléments de type bloc	
`clip`	zone visible	<forme> | auto | inherit	auto	éléments de type bloc et éléments remplacés	
`color`	couleur de police	<couleur> | inherit (H)	selon navigateur		
`content`	contenu à ajouter	[<chaîne> | <uri> | <compteur> | attr(X) | open-quote | close-quote | no-open-quote | no-close-quote]+ | inherit	chaîne vide	pseudo-éléments :before et :after	
`counter-increment`	incrémentation de compteur	[<identifiant> <entier>?]+ | none | inherit	none		
`counter-reset`	remise à zéro de compteur	[<identifiant> <entier>?]+ | none | inherit	none		

TABLEAU C–1 Index des propriétés d'affichage CSS 2 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
`cursor`	type de curseur	[[<uri> ,]* [auto | crosshair | default | pointer | move | e-resize | ne-resize | nw-resize | n-resize | se-resize | sw-resize | s-resize | w-resize| text | wait | help]] | inherit (H)	auto	(médias interactifs)	
`direction`	sens de lecture	ltr | rtl | inherit (H)	ltr	tous les éléments	
`display`	mode d'affichage d'un élément	inline | block | list-item | run-in | | inline-block | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption | none | inherit	inline	(tous médias)	
`empty-cells`	bordure des cellules vides	show | hide | inherit (H)	show	éléments avec 'display: table-cell'	

TABLEAU C–1 Index des propriétés d'affichage CSS 2 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
`float`	transformation en bloc flottant	left | right | none | inherit	none	tous les éléments, sauf ceux positionnés ou avec un contenu généré	
`font`	raccourci pour les propriétés font-...	[[font-style || font-variant || font-weight]? font-size [/ line-height]? font-family] | caption | icon | menu | message-box | small-caption | status-bar | inherit (H)	voir chaque propriété		admis pour font-size et line-height
`font-family`	police(s) de caractères	[[<famille-nom> | <famille-générique>],]* [<famille-nom> | <famille-générique>] | inherit (H)	selon l'agent utilisateur		
`font-size`	taille des caractères	<taille-absolue> | <taille-relative> | <longueur> | <pourcentage> | inherit (H)	medium		% de la taille de police du bloc parent

TABLEAU C–1 Index des propriétés d'affichage CSS 2 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
`font-style`	italique	normal | italic | oblique | inherit (H)	normal		
`font-variant`	petites majuscules	normal | small-caps | inherit (H)	normal		
`font-weight`	graissage des caractères	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit (H)	normal		
`height`	hauteur	<longueur> | <pourcentage> | auto | inherit	auto	tous les éléments, sauf en-ligne non remplacés et colonnes de tableau	voir explications
`left`	décalage à partir de la gauche	<longueur> | <pourcentage> | auto | inherit	auto	éléments positionnés	% de la largeur du bloc conteneur
`letter-spacing`	espacement entre les lettres	normal | <longueur> | inherit (H)	normal		

TABLEAU C–1 Index des propriétés d'affichage CSS 2 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
`line-height`	hauteur de ligne	normal | <nombre> | <longueur> | <pourcentage> | inherit (H)	normal		% de la taille de la police de l'élément lui-même
`list-style`	raccourci pour les propriétés list-style-...	[list-style-type || list-style-position || list-style-image] | inherit (H)	voir chaque propriété	éléments avec 'display: list-item'	
`list-style-image`	image à utiliser comme puce	<uri> | none | inherit (H)	none	éléments avec 'display: list-item'	
`list-style-position`	position de la puce	inside | outside | inherit (H)	outside	éléments avec 'display: list-item'	

TABLEAU C–1 Index des propriétés d'affichage CSS 2 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
`list-style-type`	type de puce ou de numérotation	disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-alpha | lower-latin | upper-alpha | upper-latin | hebrew | armenian | georgian | cjk-ideographic | hiragana | katakana | hiragana-iroha | katakana-iroha | none | inherit (H)	disc	éléments avec 'display: list-item'	
`margin`	raccourci pour les marges extérieures	<marge-largeur>{1,4} | inherit	voir chaque propriété		% de la largeur du bloc conteneur
`margin-top` `margin-right` `margin-bottom` `margin-left`	marges extérieures de chaque côté	<marge-largeur> | inherit	0		% de la largeur du bloc conteneur

TABLEAU C–1 Index des propriétés d'affichage CSS 2 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
`max-height`	hauteur maximum	<longueur> | <pourcentage> | none | inherit	none	tous éléments, sauf en-ligne non remplacés et éléments des tableaux	% de la largeur du bloc conteneur
`max-width`	largeur maximum	<longueur> | <pourcentage> | none | inherit	none	tous éléments, sauf en-ligne non remplacés et éléments des tableaux	% de la largeur du bloc conteneur
`min-height`	hauteur minimum	<longueur> | <pourcentage> | inherit	0	tous éléments, sauf en-ligne non remplacés et éléments des tableaux	% de la largeur du bloc conteneur

TABLEAU C–1 Index des propriétés d'affichage CSS 2 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
`min-width`	largeur minimum	<longueur> | <pourcentage> | inherit	selon le navigateur	tous éléments, sauf en-ligne non remplacés et éléments des tableaux	% de la largeur du bloc conteneur
`outline`	raccourci pour les propriétés outline-...	[outline-color || outline-style || outline-width] | inherit	voir chaque propriété		
`outline-color`	couleur du contour des boîtes	<couleur> | invert | inherit	invert		
`outline-style`	style du contour des boîtes	<bordure-style> | inherit	none		
`outline-width`	épaisseur du contour des boîtes	<bordure-épaisseur> | inherit	medium		
`overflow`	affichage des débordements	visible | hidden | scroll | auto | inherit	visible	éléments de type bloc et éléments remplacés	

TABLEAU C–1 Index des propriétés d'affichage CSS 2 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
`padding`	raccourci pour les propriétés padding-...	<espacement-largeur>{1,4} | inherit	voir chaque propriété		% de la largeur du bloc conteneur

`padding-top`
`padding-right`
`padding-bottom`
`padding-left`
	marges intérieures de chaque côté	<espacement-largeur> | inherit	0		% de la largeur du bloc conteneur

`position`	type de positionnement	static | relative | absolute | fixed | inherit	static	tous les éléments, sauf ceux avec contenu généré	
`quotes`	caractères pour guillemets	[<chaîne> <chaîne>]+ | none | inherit (H)	selon navigateur		
`right`	décalage à partir de la droite	<longueur> | <pourcentage> | auto | inherit	auto	éléments positionnés	% de la largeur du bloc conteneur

TABLEAU C–1 Index des propriétés d'affichage CSS 2 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
`table-layout`	largeur des colonnes fixe ou variable	auto | fixed | inherit	auto	éléments avec 'display: table' ou 'display: inline-table'	
`text-align`	alignement horizontal du texte	left | right | center | justify | <chaîne> | inherit (H)	selon navigateur	éléments de type bloc	
`text-decoration`	souligné-barré-clignotant-...	none | [underline || overline || line-through || blink] | inherit	none		
`text-indent`	retrait de la première ligne	<longueur> | <pourcentage> | inherit (H)	0	éléments de type bloc	% de la largeur du bloc conteneur
`text-transform`	majuscules-minuscules	capitalize | uppercase | lowercase | none | inherit (H)	none		
`top`	décalage à partir du haut	<longueur> | <pourcentage> | auto | inherit	auto	éléments positionnés	% de la largeur du bloc conteneur

TABLEAU C–1 Index des propriétés d'affichage CSS 2 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
`unicode-bidi`	gestion du texte bidirectionnel	normal | embed | bidi-override | inherit	normal		
`vertical-align`	alignement ou décalage vertical	baseline | sub | super | top | text-top | middle | bottom | text-bottom | <pourcentage> | <longueur> | inherit	baseline	éléments de type en-ligne (décalage vertical) ou avec 'display: table-cell' (alignemt)	% de la valeur de line-height de l'élément lui-même
`visibility`	affichage de l'élément	visible | hidden | collapse | inherit	inherit		
`white-space`	conservation des espaces et des sauts de ligne	normal | pre | nowrap | inherit (H)	normal	éléments de type bloc	
`width`	largeur de l'élément	<longueur> | <pourcentage> | auto | inherit	auto	tous les éléments, sauf en-ligne non remplacés et rangées de tableau	% de la largeur du bloc conteneur

TABLEAU C–1 Index des propriétés d'affichage CSS 2 (suite)

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)	Pourcentage (si utilisé)
`word-spacing`	espacement entre les mots	normal | <longueur> | inherit (H)	normal		
`z-index`	ordre de superposition	auto | <entier> | inherit	auto	éléments positionnés	

Média paginé

TABLEAU C–2 **Index des propriétés CSS 2 pour les médias paginés**

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à
`orphans`	orphelines	<entier> | inherit (H)	2	éléments de type bloc
`page`	choix de la page destination	<identifiant> | auto (H)	auto	éléments de type bloc
`page-break-after`	saut de page après	auto | always | avoid | left | right | inherit	auto	éléments de type bloc
`page-break-before`	saut de page avant	auto | always | avoid | left | right | inherit	auto	éléments de type bloc
`page-break-inside`	autorisation de saut de page	avoid | auto | inherit (H)	auto	éléments de type bloc
`size`	portrait-paysage / taille	<longueur>{1,2} | auto | portrait | landscape | inherit	auto	dans un contexte de page
`widows`	veuve	<entier> | inherit (H)	2	éléments de type bloc

Média sonore

TABLEAU C–3 Index des propriétés CSS 2 pour les médias sonores

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)
`azimuth`	angle horizontal de l'origine du son	<angle> | [[left-side | far-left | left | center-left | center | center-right | right | far-right | right-side] || behind] | leftwards | rightwards | inherit (H)	center	
`cue`	raccourci pour cue-...	[cue-before || cue-after] | inherit	voir chaque propriété	
`cue-after`	son après	<uri> | none | inherit	none	
`cue-before`	son avant	<uri> | none | inherit	none	
`elevation`	angle vertical de l'origine du son	<angle> | below | level | above | higher | lower | inherit (H)	level	
`pause`	raccourci pour pause-...	[[<durée> | <pourcentage>]{1,2}] | inherit	selon agent utilisateur	
`pause-after`	pause après	<durée> | <pourcentage> | inherit	selon agent utilisateur	

TABLEAU C–3 **Index des propriétés CSS 2 pour les médias sonores (suite)**

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)
`pause-before`	pause avant	<durée> | <pourcentage> | inherit	selon agent utilisateur	
`pitch`	fréquence moyenne	<fréquence> | x-low | low | medium | high | x-high | inherit (H)	medium	
`pitch-range`	étendue des tonalités	<nombre> | inherit (H)	50	
`play-during`	diffusion d'un son pendant	<uri> mix? repeat? | auto | none | inherit	auto	
`speak`	style de lecture	normal | none | spell-out | inherit (H)	normal	
`speak-header`	lecture des en-têtes de tableaux	once | always | inherit (H)	once	éléments contenant une information d'en-tête
`speak-numeral`	lecture des nombres	digits | continuous | inherit (H)	continuous	

TABLEAU C–3 **Index des propriétés CSS 2 pour les médias sonores (suite)**

Propriété	Fonction	Valeurs (H) si héritage	Valeur initiale	S'applique à (par défaut : tous éléments)
`speak-punctuation`	lecture des ponctuations	code | none | inherit (H)	none	
`speech-rate`	vitesse de lecture	<nombre> | x-slow | slow | medium | fast | x-fast | faster | slower | inherit (H)	medium	
`stress`	ampleur des accentuations	<nombre> | inherit (H)	50	
`richness`	clarté de la voix (portée)	<nombre> | inherit (H)	50	
`voice-family`	type de voix à utiliser	[[<voix-spécifique> | <voix-générique>],]* [<voix-spécifique> | <voix-générique>] | inherit (H)	selon agent utilisateur	
`volume`	volume moyen du son	<nombre> | <pourcentage> | silent | x-soft | soft | medium | loud | x-loud | inherit (H)	medium	

Propriétés classées par catégories

Quelles propriétés CSS peuvent être utilisées pour paragraphe, pour un tableau, pour une liste ?

L'index précédent classait les propriétés par ordre alphabétique.

Voici à présent les noms seuls des principales propriétés, avec cette fois un regroupement par catégories d'utilisation.

Caractères

```
background-color,
color,
font,
font-family,
font-size,
font-style,
font-variant,
font-weight,
text-decoration,
text-transform,
vertical-align
```

Mots, paragraphes et blocs de texte

```
background,
background-attachment, background-color,
background-image, background-position,
background-repeat,

border,
border-top, border-right, border-bottom, border-left,
border-color,
border-top-color, border-right-color,
border-bottom-color, border-left-color,
border-spacing,
border-style,
border-top-style, border-right-style,
border-bottom-style, border-left-style,
border-width,
border-top-width, border-right-width,
```

```
border-bottom-width, border-left-width,

outline,
outline-color, outline-style, outline-width,

margin,
margin-top, margin-right, margin-bottom, margin-left,

height, width,
max-height, max-width, min-height, min-width,

padding,
padding-top, padding-right,
padding-bottom, padding-left,

text-align, text-indent,
line-height, letter-spacing, word-spacing,
white-space,

content, quotes,
counter-increment, counter-reset,
direction, unicode-bidi, cursor
```

Listes à puces ou à numéros

```
list-style,
list-style-image,
list-style-position,
list-style-type
```

Tableaux

```
border-collapse,
border-spacing,
caption-side,
empty-cells,
table-layout,
text-align,
vertical-align
```

Positionnement

```
display, visibility,
float,
position,
top, bottom,
right, left,
clear, clip,
overflow,
z-index
```

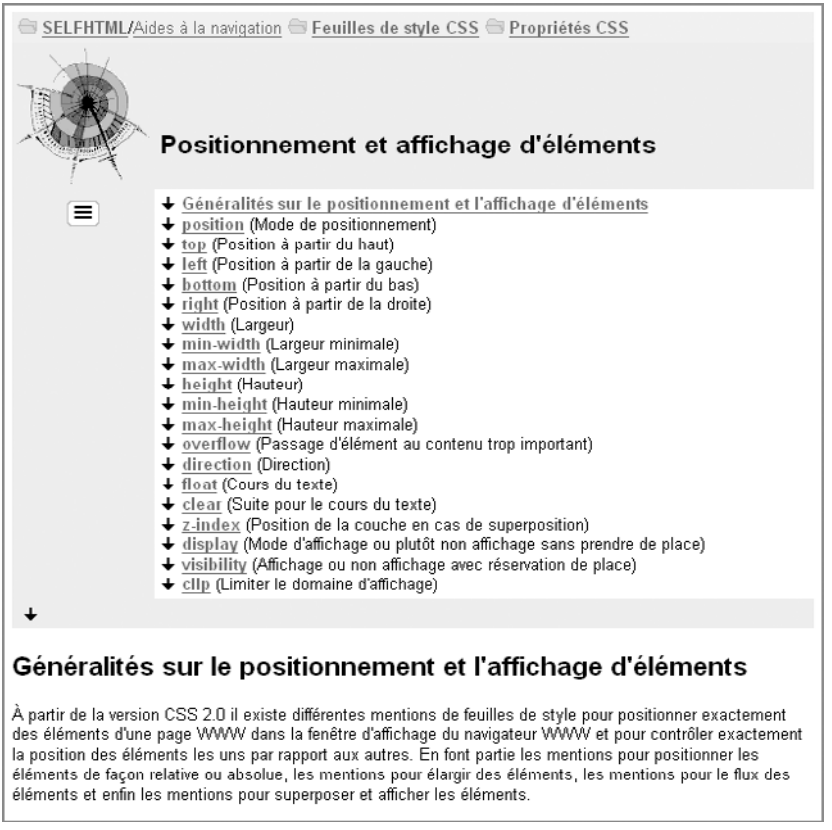


FIGURE C–2 *Extrait de la page http://fr.selfhtml.org/css/proprietes/positionnement.htm. Le site SelfHTML est une source d'information très complète pour le développeur web.*

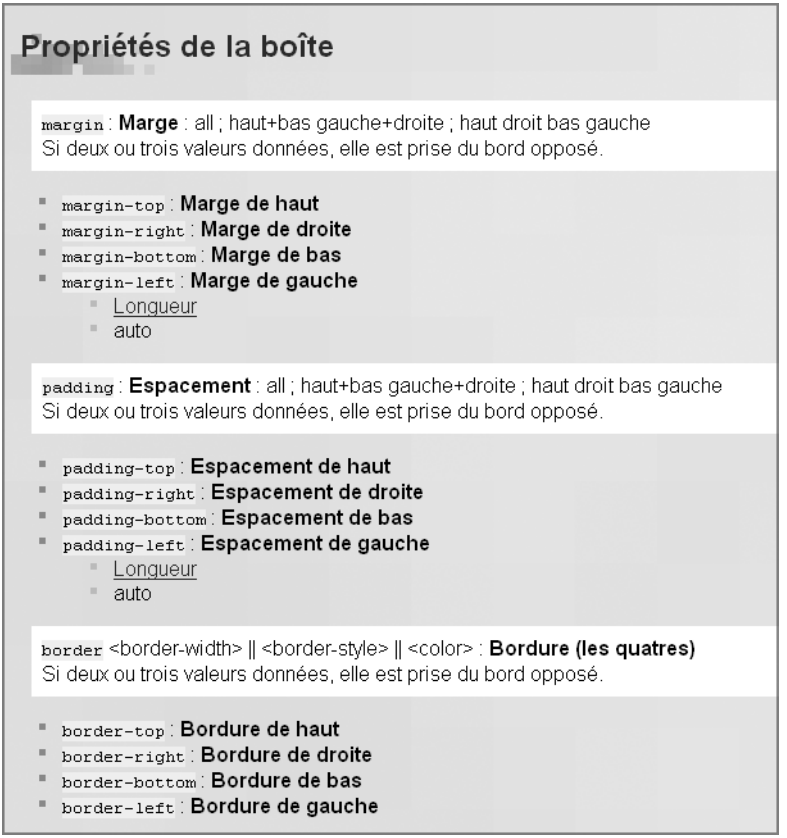


Figure C–3 *Extrait de la page http://slaout.linux62.org/html_css/doc_css.html, qui propose des tutoriels et des aide-mémoire sur le HTML et les CSS.*

annexe

D

Références bibliographiques et sites web

Voici maintenant quelques références pour compléter cet ouvrage et aller plus loin, ainsi que des sites web intéressants.

Sommaire

L'objectif de cet ouvrage était la présentation des techniques de base nécessaires pour créer des pages web, à savoir l'essentiel du XHTML, les feuilles de style CSS et leur mise en pratique. Une fois que vous aurez compris la philosophie de la programmation XHTML/CSS, il pourra continuer à vous servir d'aide-mémoire.

Pour aller plus loin, n'hésitez pas à consulter des ouvrages plus volumineux, qui vous présenteront en détail un certain nombre de cas concrets d'application.

Bibliographie

Voici une liste d'ouvrages qui sont actuellement des références en matière de développement XHTML/CSS :

- *Réussir son site web avec XHTML et CSS*, de Mathieu Nebra, éditions Eyrolles
- *CSS 2 Pratique du design web*, de Raphaël Goetter, éditions Eyrolles
- *Memento CSS*, de Raphaël Goetter, éditions Eyrolles
- *XHTML et CSS*, cours et exercices, de Jean Engels, éditions Eyrolles
- *Design web : utiliser les standards CSS et XHTML*, de Jeffrey Zeldman, éditions Eyrolles
- *CSS par Éric Meyer*, d'Éric Meyer, éditions Campus Press
- *CSS 2, Guide du designer*, de Charles Wike-Smith, éditions Campus Press
- *Des CSS au DHTML : Javascript appliqué aux feuilles de style*, de Luc Van Lancker, éditions ENI

Sites web utiles

Les quelques sites web qui suivent sont très intéressants (attention à ne pas oublier le signe « / » qui termine certaines adresses). Vous pourrez y glaner d'autres informations, applications pratiques et astuces. Cette liste n'est évidemment pas exhaustive, ce n'est qu'un aperçu des trésors de la toile...

- World Wide Web Consortium (normes web)
  http://www.w3c.org
- Spécifications officielles des CSS 2.1 en anglais, par le W3C
  http://www.w3.org/TR/CSS21/indexlist.html
- Spécifications officielles du CSS 2, traduites en français
  http://www.yoyodesign.org/doc/w3c/css2/indexlist.html
- OpenWeb
  http://openweb.eu.org/css/
- Pompage : le *web design* puisé à la source, sur
  http://pompage.net/
- CSS : On reprend tout à zéro !
  http://pompage.net/pompe/cssdezero-1/
- SelfHTML
  http://fr.selfhtml.org/
- SelfHTML : Javascript
  http://fr.selfhtml.org/javascript/index.htm
- Alsacréations
  http://www.alsacreations.com
- Forum d'Alsacréations
  http://forum.alsacreations.com
- Tutoriel HTML et CSS
  http://slaout.linux62.org/html_css/
- Aidenet
  http://www.aidenet.com
- Feuilles de style sur Aidenet
  http://www.aidenet.com/css/index.htm
- Framasoft (logiciels et documentation)
  http://www.framasoft.net
- Zen Garden (démonstration très esthétique des possibilités apportées par les feuilles de style)
  http://csszengarden.com/tr/francais

FIGURE D–1 *Extraits du site http://www.framasoft.net, réalisation de la communauté francophone du logiciel libre, qui recense et commente plus de mille programmes, propose des liens pour leur téléchargement, des tutoriels et des forums de discussion.*

Index

C

D

E

T

U

V

W

X

Z

Premiers pas en CSS et XHTML

L'informatique libre *à la portée* de **tous** !

Choisissez la simplicité et l'élégance du couple XHTML et CSS

pour créer vos sites web avec style et panache !

- → Séparez mise en forme CSS et contenu XHTML pour plus de simplicité et de souplesse ;
- → Changez sans peine la charte graphique de votre site ;
- → Apprenez à écrire une feuille de style CSS pour une présentation homogène ;
- → Comprenez la hiérarchie des éléments et des balises ;
- → Enrichissez votre texte : taille, couleur, police, interligne... ;
- → Embellissez vos tableaux : bordures, marges et arrière-plans ;
- → Positionnez vos paragraphes, images et autres éléments blocs : centrés, justifiés, flottants... ;

Cette deuxième édition tient compte de l'évolution des standards et des navigateurs. Elle rappelle l'emploi des balises XHTML et l'insertion de vidéos et d'animations Flash.

En annexes : Codage des principales couleurs • Spécificités des navigateurs Mozilla Firefox, Internet Explorer, Opéra • Aide-mémoire des principales propriétés CSS.

L'auteur

Francis Draillard est ingénieur EFREI et a travaillé dans l'industrie, la recherche, puis dans l'enseignement supérieur et la formation continue. Il enseigne en tant que professeur associé à l'EIGSI et Sup de Co La Rochelle. Concepteur web indépendant, il exerce une activité de conseil en entreprise et contribue au site collaboratif Framasoft dédié aux logiciels libres sous Windows, Linux et Mac OS X.

Conception : Nord Compo

*Illustration Ayo (**Alexis Younes**, www.73lab.com)*