

Le Campus

Maîtrise des CSS

2^e édition



Andy Budd
avec Cameron Mall et Simon Collison

PEARSON

Maîtrise des CSS

Pearson Education France a apporté le plus grand soin à la réalisation de ce livre afin de vous fournir une information complète et fiable. Cependant, Pearson Education France n'assume de responsabilités, ni pour son utilisation, ni pour les contrefaçons de brevets ou atteintes aux droits de tierces personnes qui pourraient résulter de cette utilisation.

Les exemples ou les programmes présents dans cet ouvrage sont fournis pour illustrer les descriptions théoriques. Ils ne sont en aucun cas destinés à une utilisation commerciale ou professionnelle.

Pearson Education France ne pourra en aucun cas être tenu pour responsable des préjudices ou dommages de quelque nature que ce soit pouvant résulter de l'utilisation de ces exemples ou programmes.

Tous les noms de produits ou autres marques cités dans ce livre sont des marques déposées par leurs propriétaires respectifs.

Publié par Pearson Education France
47 bis, rue des Vinaigriers
75010 PARIS
Tél. : 01 72 74 90 00
www.pearson.fr

Mise en pages : TyPAO

ISBN : 978-2-7440-4130-3

Copyright © 2010 Pearson Education France

Tous droits réservés

Titre original : *CSS Mastery, Advanced Web Standards Solutions*, second edition

Traduit par Patrick Fabre,
avec la contribution de Bruno Sebarte

ISBN original : 978-1-4302-2397-9

**Copyright © 2009 by Andy Budd, Simon Collison,
and Cameron Moll**

All rights reserved

Édition originale publiée par FriendsofEd/Apress
(www.friendsofed.com/www.apress.com)

Aucune représentation ou reproduction, même partielle, autre que celles prévues à l'article L. 122-5 2° et 3° a) du code de la propriété intellectuelle ne peut être faite sans l'autorisation expresse de Pearson Education France ou, le cas échéant, sans le respect des modalités prévues à l'article L. 122-10 dudit code.

No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

Maîtrise des CSS

Andy Budd
avec Cameron Moll et Simon Collison



Table des matières

Avant-propos	IX
À propos des auteurs	XI
À propos des réviseurs techniques	XII
Remerciements	XIII
Introduction	1
À qui ce livre est-il destiné ?	1
Structure du livre.....	2
Conventions utilisées	2
1 Les fondations	5
Structuration du code.....	5
Historique rapide du balisage	6
Types de document, commutation de DOCTYPE et modes des navigateurs	19
Validation.....	19
En résumé	22
2 Des styles qui atteignent leur cible	23
Les sélecteurs courants	23
Pseudo-classes	24
Le sélecteur universel	25
Les sélecteurs avancés	25
Sélecteur d'enfants et sélecteur de frère adjacent.....	26
Sélecteur d'attribut.....	27
Cascade et spécificité	30
Héritage	34
Planification, organisation et gestion des feuilles de styles	35
Application de styles aux documents	35
Guides de styles	40
En résumé	42

3 Vue d'ensemble du modèle de formatage visuel	43
Récapitulatif sur le modèle de boîte	43
Internet Explorer et le modèle de boîte	45
Effondrement des marges	46
Récapitulatif concernant le positionnement	48
Le modèle de formatage visuel	48
Positionnement relatif	49
Positionnement absolu.....	50
Flottement.....	52
En résumé	58
4 Utilisation des effets d'arrière-plan.....	61
Notions de base sur les images d'arrière-plan	61
Boîtes à bords arrondis.....	64
Boîtes à largeur fixe et bords arrondis.....	64
Coins masqués.....	70
Ombres portées	75
Ombres portées simples en CSS	75
Ombres portées "à la Clagnut"	77
Opacité.....	80
Opacité CSS	80
Remplacement d'images	85
Remplacement des images de Fahrner.....	86
Phark	87
Méthode IFR évolutive.....	87
En résumé	89
5 Mise en forme des liens.....	91
Mise en forme simple des liens.....	91
Soulignements amusants	93
Embellissement simple des liens	93
Soulignement de liens originaux	94
Style de liens visités.....	94
Mise en forme des cibles des liens.....	95
Différenciation des types de liens.....	96
Signalement de documents téléchargeables et de flux	98

Création de liens qui ressemblent à des boutons	99
Survol simple	100
Survol avec images	101
Survol style Pixy	102
Sprites CSS	103
Survol CSS 3	104
Info-bulles CSS	106
En résumé	108
6 Mise en forme des listes et création de barres de navigation...	109
Mise en forme de base des listes	109
Création d'une barre de navigation verticale	110
Signaler la page courante dans la barre de navigation	113
Création d'une barre de navigation horizontale simple	114
Création d'une barre de navigation graphique	116
Système de navigation élastique à onglets	118
Menus déroulants Suckerfish	120
Cartes-images CSS	123
Cartes-images à la mode Flickr	126
Survol distant	132
Note à propos des listes de définitions	136
En résumé	137
7 Mise en forme des formulaires et des tableaux de données...	139
Mise en forme des tableaux de données	139
Éléments spécifiques des tableaux	141
Balisage des tableaux de données	142
Mise en forme du tableau	143
Ajout de style visuel	144
Mise en forme simple de formulaires	146
Éléments de formulaire utiles	147
Disposition de base	148
Autres éléments	150
Embellissements	151
Mise en forme complexe de formulaire	153

Champ de date accessible	154
Cases à cocher multicolonnes.....	155
Retour de formulaire	158
En résumé	160
8 Mise en page.....	161
Planification de la mise en page	161
Les premières fondations.....	164
Centrage d'une mise en page à l'aide des marges.....	165
Mises en page flottantes.....	167
Mise en page flottante à deux colonnes	167
Mise en page flottante à trois colonnes	170
Mises en page à largeur fixe, liquide et élastique	172
Mises en page liquides	173
Mises en page élastiques.....	176
Images liquides et élastiques.....	178
Fausses colonnes	180
Colonnes de hauteurs égales.....	183
Colonnes CSS 3	187
Frameworks CSS et systèmes CSS	188
En résumé	191
9 Bogue et correctifs.....	193
La chasse aux bogues	193
Problèmes CSS courants.....	194
Les fondamentaux de la chasse aux bogues.....	198
Essayer d'éviter les bogues par avance !	200
Isoler le problème	200
Créer des cas de test minimaux.....	201
Résoudre le problème, pas le symptôme	202
Demander de l'aide	202
Avoir ou ne pas avoir un "layout"	203
Qu'est-ce que le "layout" ?	203
Quel effet résulte du layout ?.....	204
Solutions de contournement.....	206
Les commentaires conditionnels d'Internet Explorer	206

Avertissement concernant les hacks et les filtres	207
Utiliser les hacks et les filtres avec raison	208
Appliquer le filtre passe-bande pour Internet Explorer version Mac	209
Le hack de l'étoile HTML	209
Appliquer le hack du sélecteur d'enfants	210
Bogues et correctifs courants	211
Bogue de la double marge des éléments flottants	211
Bogue du décalage de texte de 3 pixels	211
Bogue des caractères dupliqués d'Internet Explorer 6	213
Bogue du coucou d'Internet Explorer 6	214
Positionnement absolu dans un conteneur relatif	215
Cesser de se référer à Internet Explorer	216
Prise en charge graduelle des navigateurs	217
En résumé	219
10 Étude de cas : Roma Italia	221
À propos de cette étude de cas	221
Les fondations	223
Un œil rivé sur le HTML 5	223
reset.css	225
La mise en page et la grille à 1 080 pixels	226
Utilisation des grilles pour la conception web	228
Fonctionnalités CSS 2 et CSS 3 avancées	230
Les sites doivent-ils être identiques dans tous les navigateurs ?	230
Sélecteur d'attribut	232
box-shadow, RGBa et text-overflow	234
Liaison des polices et typographie web améliorée	237
Définir <i>font-size</i> comme en 1999	237
Ponctuation hors justification	238
Affichage du texte sur plusieurs colonnes	240
@font-face	241
Cufón, étape intermédiaire avant @font-face	245
Interactivité avec Ajax et jQuery	247
Ajax	247
jQuery	248

Utilisation d'Ajax et de jQuery pour la fonctionnalité de recherche	249
En résumé	251
11 Étude de cas : Climb the Mountains	253
À propos de cette étude de cas	253
Organisation et conventions de la feuille de styles	255
La laborieuse feuille screen.css.....	256
Réinitialisation	257
Feuilles de styles Internet Explorer utilisant des commentaires conditionnels	258
Flexibilité de la grille.....	258
Fonctionnement de la mise en page du site CTM	259
Contrôle de la navigation avec les classes de l'élément <i>body</i>	260
Mise en valeur de la page courante	260
Mise en forme de la citation	263
Ciblage stratégique des éléments.....	264
Sélecteurs descendants profonds	264
La pseudo-classe <i>:first-child</i>	266
Sélecteurs de frères adjacents.....	268
Transparence, ombres et bords arrondis.....	269
Notre objectif	269
Bandeau translucide de légende et transparence RGBa	270
Combinaison de classes	272
<i>border-radius</i>	273
<i>box-shadow</i>	274
Positionner des listes et révéler leur contenu.....	275
Arrondir les coins	277
Le graphique d'altitude	277
En résumé	282
Index.....	285

Avant-propos

Dans notre merveilleux univers de la conception web, il existe toujours trente-six mille moyens de parvenir au même résultat. Et ce nombre ne fait qu'augmenter de jour en jour. Plutôt que d'avoir une seule manière de résoudre tel ou tel problème, on a le bonheur et le malheur à la fois de disposer d'une abondance de choix. Ce sont eux qui font le sel de ce métier, mais ce sont eux, aussi, qui le rendent si difficile. Le livre *Maîtrise des CSS* est là pour vous aider à traverser ces difficultés.

Andy Budd pratique les techniques de conception web et écrit ou parle à leur sujet depuis de nombreuses années. Ce livre offre à chacun la chance de profiter de ses techniques d'enseignement CSS essentielles sous un format facile à suivre. Il s'agit d'un catalogue de solutions, d'astuces et de trucs indispensables pour les professionnels du web.

J'ai toujours grincé des dents en lisant des livres qui affirmaient péremptoirement qu'un seul moyen simple existait pour réaliser tel ou tel objectif. Andy fait tout le contraire. Il propose plusieurs méthodes pour chaque tâche, comme la mise en forme des liens, la création d'outils de navigation à onglets, les solutions CSS 3 qui font gagner du temps, la création de mises en page fixes, fluides ou élastiques ou encore les astuces pour réparer ces satanés bogues de navigateurs qui hantent le quotidien des développeurs CSS. Armé de ces techniques populaires et élégantes, vous serez mieux préparé pour prendre des décisions avisées.

Et comme si cela ne suffisait pas, Andy a même pris les devants et s'est adjoint l'aide de deux concepteurs de haut rang pour créer un assemblage de toutes ces pièces, afin de montrer comment il était possible de combiner ces techniques essentielles. Cela fait longtemps que j'admire le travail de Cameron et de Simon. Les deux excellentes études de cas, qui présentent respectivement une maquette fluide et fiable et une solution de mise en forme flexible, sont un ajout inestimable.

Plongez-vous dans cet ouvrage et commencez les fouilles dans ces trente-six mille moyens de maîtriser les CSS.

Dan Cederholm

Auteur du best-seller *Web Standards Solutions*

À propos des auteurs

Andy Budd est l'un des fondateurs du groupe *User Experience Design Consultancy*, ou Clearleft (clearleft.com). Spécialiste en conception interactive et en utilisabilité, il est régulièrement invité à s'exprimer lors des conférences internationales comme Web Directions, An Event Apart et SXSW. Il s'occupe de dConstruct (dconstruct.org), l'une des conférences les plus populaires du Royaume-Uni en matière de conception web, et il est également responsable d'UX London (uxlondon.com), le premier événement du Royaume-Uni consacré à l'utilisabilité, à l'architecture de l'information et à l'expérience utilisateur.

Andy fut un précurseur des standards du Web au Royaume-Uni. Il a acquis une connaissance poussée de la spécification CSS et de la prise en charge des navigateurs. Comme membre actif de la communauté, il a participé au jury de plusieurs prix de conception internationaux et siège actuellement au comité consultatif du magazine web *.net*. Andy est également l'architecte de Silverbackapp (silverbackapp.com), un outil de test d'utilisabilité à bas coût pour Mac. Il est un aficionado de Twitter (@andybudd) et blogue à l'occasion sur andybudd.com.

Au comble du bonheur lorsqu'il plonge dans quelque atoll tropical lointain, Andy est professeur de plongée et pêcheur de requins à ses heures.

Cameron Moll conçoit des interfaces web signifiantes, harmonisant utilité et présentation depuis la fin des années 90. Ses travaux et son expertise ont été mis à profit par les magazines *HOW*, *Print et Communication Arts*, par Forrester Research, la National Public Radio (NPR) et bien d'autres organisations. Il intervient lors de conférences nationales et internationales sur la conception des interfaces utilisateur. Il est aussi l'auteur de *Mobile Web Design* (mobilewebbook.com).

Cameron est le fondateur et le président d'Authentic Jobs Inc. (authenticjobs.com), une destination ciblée pour les professionnels du Web et de la création et les entreprises désireuses de les embaucher. Il est également propriétaire de Cameron Moll LLC, qui propose, entre autres produits, des posters de typographie en relief (à acheter sur cameronmoll.bigcartel.com). Au beau milieu de cette folle activité, il trouve encore le temps de jouer au ballon avec chacun de ses quatre garçons.

Vous pouvez aussi retrouver Cameron en ligne sur cameronmoll.com, twitter.com/cameronmoll, flickr.com/photos/authentic et vimeo.com/cameronmoll.

Simon Collison est le cofondateur et le directeur artistique d'Erskine Design (erskinede-sign.com) où il s'intègre à une équipe talentueuse de graphistes et de développeurs aux réalisations exceptionnelles. Au cours des dix dernières années, il a travaillé sur de nombreux projets web pour des maisons de disques, des groupes de musique, des artistes, des sociétés ou pour le gouvernement – toute la panoplie semble y passer. Il travaille maintenant avec une importante liste de clients qui vont des magazines réputés jusqu'aux explorateurs polaires.

Colly tient un blog depuis longtemps (colly.com) et rédige des articles concernant le Web sur ErskineLabs (erskinelabs.com). Il est l'auteur du best-seller *Beginning CSS Web Development* chez Apress et le coauteur de *Web Standards Creativity*. Son principal bonheur ? Expérimenter les CSS et le HTML ou en parler devant un auditoire.

Dans la "vraie" vie, Colly adore grimper en montagne et se perdre dans la nature sauvage du Royaume-Uni ou d'Islande. Il conduit une voiture vieille de trente-deux ans et possède un chat extraordinairement bête qu'il a appelé Bearface (face d'ours).

À propos des réviseurs techniques

Natalie Downe est une perfectionniste par nature. Elle travaille pour Clearleft à Brighton, comme développeur web côté client. Consultante expérimentée en utilisabilité et chef de projet, ses premières amours restent le développement front-end et l'ingénierie en utilisabilité. Elle aime que les choses soient bien faites et se risque à l'occasion aux arts obscurs du Python et de sa bizarre API.

Tony White est un développeur front-end et concepteur résidant à Memphis dans le Tennessee. Pendant la semaine, il gère l'interface utilisateur des hôtels *Hilton*, où il veille à l'utilisabilité, milite en faveur des standards du Web et enflamme le HTML à l'aide de jQuery. Il dirige aussi le one-man show *Ask the CSS Guy* (askthecssguy.com), un site géré à ses heures perdues où il scrute les entrailles des techniques de conception web CSS et JavaScript.

Remerciements

Merci à tous ceux qui m'ont aidé à réaliser le livre, directement ou indirectement.

À mes amis et collègues chez Clearleft : merci de m'avoir encouragé et de m'avoir fait part de vos remarques tout au long du processus d'écriture. Merci en particulier à Natalie Downe, qui a gratifié les lecteurs de son expérience et de l'étendue de son savoir. Ton aide et tes conseils étaient inestimables et je ne comprends toujours pas comment tu trouves le temps.

Merci à Chris Mills, pour m'avoir guidé lors du processus initial d'écriture et aidé à matérialiser mes idées, ainsi qu'à tous ceux, chez Apress, qui ont travaillé sans relâche pour que ce livre soit publié à temps : votre dévouement et votre professionnalisme ont été très appréciés.

Merci aussi à tous mes collègues qui continuent à partager leurs trésors de connaissances pour faire du Web un monde meilleur. Ce livre n'aurait pas été possible sans le travail préalable des personnes suivantes, pour ne nommer que celles-là : Cameron Adams, John Allsopp, Rachel Andrew, Nathan Barley, Holly Bergevin, Mark Boulton, Douglas Bowman, The BritPack, Dan Cederholm, Tantek Çelik, Joe Clark, Andy Clarke, Simon Collison, Mike Davidson, Garrett Dimon, Derek Featherstone, Nick Fink, Patrick Griffiths, Jon Hicks, Molly E. Holzschlag, Shaun Inman, Roger Johansson, Jeremy Keith, Ian Lloyd, Ethan Marquette, Drew McLellan, Eric Meyer, Cameron Moll, Dunstan Orchard, Veerle Pieters, D. Keith Robinson, Richard Rutter, Jason Santa Maria, Dave Shea, Jeffrey Veen, Russ Weakley, Simon Willison et Jeffrey Zeldman.

Merci également à tous les lecteurs de mon blog et à tous ceux que j'ai rencontrés lors de conférences, d'ateliers et de formations au cours des deux dernières années. Vos discussions et vos idées ont aidé à alimenter le contenu de ce livre.

Enfin, merci à vous de lire ce livre. J'espère qu'il vous permettra de franchir un nouveau cap pour vos compétences CSS.

Andy Budd

Pour commencer, merci d'avoir choisi ce livre. J'espère qu'il contribuera à améliorer votre travail au quotidien. Je suis sans cesse inspiré par le potentiel de ceux qui travaillent dans l'univers du Web et vous en faites partie.

Comme Andy, je veux remercier les nombreuses personnes qui ont modelé et remodelé le Web pour l'améliorer de jour en jour. Dans quelques années, ces gens seront autant admirés que les hommes qui furent envoyés sur la Lune.

Un grand merci à Aaron Barker (aaronbarker.net) qui m'a aidé avec quelques exemples jQuery et Ajax dans mon étude de cas.

Enfin, par-dessus tout, ma gratitude va à ma merveilleuse épouse Suzanne et à mes quatre fils, Everest, Edison, Isaac et Hudson. Sans leur amour, leur soutien et leur patience, tout ce travail n'aurait pu voir le jour.

Cameron Moll

Je dois remercier mon collègue et ami Gregory Wood pour ses idées et son aide avec le concept du site Climb the Mountains. Tout ce qu'il produit m'inspire. Voilà le concepteur que je voudrais être quand je serai grand... J'aimerais aussi remercier tous mes collègues chez Erskine Design pour leur soutien et pour leur indulgence concernant ce travail enfiévré. Un grand merci à Simon Campbell, Jamie Pittock, Glen Swinfield, Phil Swan, Vicky Twycross et Angela Campbell.

Par-dessus tout, j'aimerais saisir cette occasion de remercier ma mère et ceux que j'ai perdus depuis la première édition de ce livre, mes deux grands-pères et tout particulièrement mon père. Si je fais tout ça, c'est encore pour que tu sois fier, quand bien même tu es déjà parti.

Simon Collison

Introduction

Les ressources CSS ne cessent de se multiplier et, pourtant, les mêmes questions n'en finissent pas de revenir dans les forums de discussion : Comment centrer une mise en page ? Quelle est la meilleure technique pour créer des bords arrondis ? Comment créer une structure à trois colonnes ?

Si vous suivez la communauté des développeurs CSS, il vous suffit généralement de vous souvenir du site où figure une technique ou un article particulier. Mais si vous débutez ou si vous n'avez pas le temps de lire tous les blogs, cette information peut être difficile à retrouver.

Les utilisateurs expérimentés en CSS eux-mêmes rencontrent des problèmes avec certains aspects plus obscurs des CSS, comme le modèle de positionnement ou la spécificité. C'est que la plupart d'entre eux sont des autodidactes, qui picorent des astuces dans des articles ou dans le code d'autres programmeurs sans toujours comprendre parfaitement les spécifications. Et comment pourrait-il en être autrement, alors que la spécification est complexe, souvent contradictoire, et écrite pour les fabricants de navigateurs plutôt que pour les développeurs web ?

Ensuite, il y a les navigateurs, avec lesquels il faut lutter. Leurs bogues et leurs incohérences sont l'un des plus importants problèmes pour les développeurs CSS modernes. Malheureusement, bon nombre de ces bogues sont mal documentés et leurs correctifs quelquefois hasardeux. Vous savez qu'il faut procéder d'une certaine manière pour que quelque chose fonctionne dans un navigateur ou un autre, mais vous avez du mal à vous souvenir pour quel navigateur ni comment se produit le dysfonctionnement.

Ainsi a survi l'idée d'un livre. Un livre qui regrouperait les techniques CSS les plus utiles, qui se concentrerait sur les problèmes concrets de navigateurs et qui aiderait à combler les lacunes les plus courantes en matière de CSS. Un livre qui vous aiderait à apprendre plus vite et vous amènerait à programmer les CSS comme un expert, en un rien de temps.

À qui ce livre est-il destiné ?

Maîtrise des CSS est destiné à tous ceux qui possèdent une connaissance rudimentaire du HTML et des CSS. Que vous veniez tout juste de tremper le pied dans l'univers de la conception CSS ou que vous ayez développé des sites en CSS depuis plusieurs années, vous trouverez de quoi vous alimenter dans ce livre. Vous en tirerez cependant le meilleur profit si vous avez utilisé des CSS pendant un certain temps et que vous ne vous considérez pas encore comme un expert. Ce livre est rempli d'exemples pratiques complets et d'astuces pour vous aider à maîtriser les créations CSS modernes.

Structure du livre

Cet ouvrage démarre en douceur, avec trois chapitres sur les concepts de base et les meilleures pratiques des CSS. Vous verrez comment structurer le code et le commenter, vous découvrirez tout sur le modèle de positionnement des CSS et comment fonctionnent véritablement le flottement et le dégagement des éléments. Vous en savez peut-être déjà pas mal sur ces sujets, mais il est probable que vous découvrirez certaines choses que vous avez manquées ou mal comprises. Les trois premiers chapitres sont à ce titre une excellente introduction aux CSS, mais aussi un bon récapitulatif pour des connaissances déjà acquises.

Une fois les bases traitées, les cinq chapitres suivants couvrent les techniques CSS essentielles, comme la manipulation des images, des liens et des listes, la conception de formulaires et de tableaux de données et la mise en page CSS. Chacun d'eux commence simplement puis avance progressivement vers des exemples plus compliqués. Vous verrez comment créer des boîtes à bords arrondis, des images avec des ombres portées transparentes, des barres de navigation à onglets et des boutons interactifs. Parmi d'autres techniques, vous commencerez par découvrir la procédure classique avant de voir comment parvenir au même effet avec des CSS 3. Si vous souhaitez suivre les exemples du livre, vous pouvez télécharger tout le code sur le site www.pearson.fr.

Les bogues de navigateurs sont la croix et la bannière pour bien des développeurs CSS. Tous les exemples de ce livre s'attachent donc à des techniques qui fonctionnent dans les différents navigateurs. En outre, un chapitre complet est consacré aux bogues et à leur résolution. Vous y apprendrez tout sur les techniques de chasse aux bogues et sur la manière d'identifier et de réparer les bogues courants avant qu'ils ne posent problème. Vous découvrirez même les causes précises de nombreux bogues CSS apparemment aléatoires d'Internet Explorer.

Les deux derniers chapitres forment la pièce de résistance de l'ouvrage. Simon Collison et Cameron Moll, deux des meilleurs concepteurs CSS existants, ont combiné toutes ces techniques dans deux formidables études de cas. Vous apprendrez non seulement comment elles fonctionnent mais également comment les appliquer en pratique dans un projet web réel.

Ce livre peut être lu de bout en bout ou conservé à proximité de votre ordinateur pour servir de référence sur les astuces et techniques CSS. À vous de choisir.

Conventions utilisées

Ce livre utilise quelques conventions qu'il vaut la peine de noter :

- HTML fait tout aussi bien référence au HTML qu'au XHTML.
- Sauf indication contraire, la mention CSS renvoie en fait à la spécification CSS 2.1.
- La formule "Internet Explorer 6 (IE 6) et ses versions antérieures sous Windows" fait référence aux versions Internet Explorer 5.0 à 6.0 sous Windows.
- Les "navigateurs web modernes" font référence aux dernières versions de Firefox, Safari et Opera, ainsi qu'à Internet Explorer 7 et au-delà.

- Tous les exemples HTML de ce livre sont censés s'insérer dans la balise `<body>` d'un document valide, les CSS étant contenues dans une feuille de styles externe. À l'occasion, les codes HTML et CSS peuvent être placés dans un même exemple, par souci de concision. Dans un véritable document, ces éléments devraient cependant rejoindre leurs emplacements respectifs pour fonctionner correctement.

Enfin, pour les exemples HTML qui contiennent des données répétées, nous avons utilisé le signe trois points (...) pour signaler que le code se poursuit, au lieu d'écrire toutes les lignes.

Les formalités sont faites : passons à l'action !

1

Les fondations

L'espèce humaine est curieuse de nature. Rien ne nous amuse plus que de trifouiller dans tout ce qui nous tombe sous la main. J'ai moi-même récemment acheté un iMac et il n'a pas fallu plus de quelques secondes pour que je le mette en pièces et l'étudie sous toutes ses coutures. Avant même d'avoir lu le manuel. On ne se refait pas. Comme tout le monde, j'adore bricoler, voir comment les choses fonctionnent, rêver à d'improbables modélisations. Ce n'est qu'une fois que la pagaille est complète, qu'une panne fatale s'est produite ou que je tombe sur quelque chose de parfaitement inexplicable que je finis par ouvrir la documentation.

L'un des meilleurs moyens d'apprendre les CSS (*Cascading Style Sheets* ou feuilles de styles en cascade) consiste à se jeter d'emblée à l'eau et à bricoler sans attendre. Je parie même que c'est ainsi que la plupart d'entre vous ont appris à programmer – en glanant quelques informations dans un blog, en examinant le code source de telle page pour comprendre comment elle s'affiche et en testant le tout dans vos propres sites. Il est peu probable que vous ayez commencé par lire l'ensemble de la spécification CSS. Peu d'individus y survivraient, du reste.

Le bricolage est le meilleur moyen de commencer, mais si vous n'y prêtez attention, vous risquez de mal comprendre certains concepts essentiels ou d'accumuler des problèmes qui ressurgiront par la suite. Je sais de quoi je parle : cela m'est arrivé plusieurs fois. Je me propose donc de passer en revue quelques concepts élémentaires mais souvent mal compris et de vous montrer la manière de procéder pour que votre code HTML et CSS reste clair et bien structuré.

Au cours de ce chapitre, vous découvrirez :

- comment structurer le code ;
- l'importance d'une documentation facile à comprendre ;
- les conventions de nommage ;
- à quels moments utiliser des ID et des noms de classes ;
- les microformats ;
- les différentes versions du HTML et des CSS ;
- les types de document, la commutation de DOCTYPE et les modes des navigateurs.

Structuration du code

En général, les gens ne se posent pas de questions sur les fondations des immeubles. Et pourtant, sans de bonnes fondations, bien robustes, la plupart des bâtiments ne tiendraient pas debout. Si ce livre traite avant tout des techniques CSS avancées, une grande partie de

ce que vous allez apprendre ne peut se faire (ou alors très difficilement) sans que vous ayez un document HTML valide et bien structuré.

À cette section, vous allez voir pourquoi le HTML clair et bien structuré est essentiel pour le développement de sites compatibles avec les standards. Vous verrez également comment intégrer d'autres informations à vos documents et, ce faisant, comment faciliter le développement.

Historique rapide du balisage

À ses débuts, le Web n'était guère plus qu'un amas de documents de recherche, reliés les uns aux autres par du code HTML servant à les mettre en forme et à les structurer simplement. Cependant, quand sa popularité s'est accrue, il a été utilisé par les développeurs à des fins de présentation. Au lieu de se servir des éléments de titre pour les titres des pages, ils ont voulu recourir à des combinaisons de balises de police et de style pour créer différentes sortes d'effets visuels. Les tableaux, d'abord employés pour l'affichage des données, sont rapidement devenus des outils de mise en page tandis que les instructions `blockquote` ont servi à insérer des espaces blancs au lieu de signaler la présence de citations. Très vite, le Web a perdu tout son sens et s'est transformé en un véritable salmigondis de balises de polices et de tableaux. Les concepteurs web ont même forgé une expression pour désigner ce type de balisage : la "soupe à balises" (voir Figure 1.1).

Figure 1.1

Le balisage de l'article à la une d'**abcnews.com**, le 14 août 2000, utilise des tableaux pour la mise en forme et du texte en gras et très grand pour les titres. Le code manque de structure et est difficile à comprendre.



```

<----- 11 MAIN CONTENT----->
<td colspan="2" width="425" valign="top" bgcolor="#eeeeee">
<table width="417" cellspacing="0" cellpadding="4" border=0>
<tr><td width="417" valign="top">
<a href="/sections/politics/DailyNews/DEMCVN_open000813.html" >
</a>
<font face="geneva,arial,helvetica size=5><b>
<a href="/sections/politics/DailyNews/DEMCVN_open000813.html" >
Painting the Torch
<b></b>
</b></font><br>
<font face="geneva,arial,helvetica size=2>Bill Clinton gave a spirited
defense of his eight years in office and touted the qualifications of his
vice president, Al Gore, who wants to take Clinton's place in the
White House. Get full coverage and <a href
="http://abcnews.go.com/sections/politics/DailyNews/DEMCVN_trans_clinton0008
a transcript</a></font>

```

À force d'être orienté sur la présentation, le code des pages web est devenu de plus en plus difficile à comprendre et à gérer. Les éditeurs WYSIWYG (*What You See Is What You Get*, littéralement : "Ce que vous voyez, c'est ce que vous obtenez") offrent aux auteurs une échappatoire en mettant à leur disposition des outils de mise en page visuels. Mais au lieu de simplifier les choses, ces outils ont contribué à les compliquer encore, en produisant leur propre tambouille de balises. Les éditeurs comme FrontPage ou Dreamweaver permirent de créer des tableaux de mise en page complexes en un clic, mais en truffant le code de tableaux imbriqués et de GIF d'espacement (voir Figure 1.2). Ces mises en page étaient extrêmement fragiles et menaçaient de dysfonctionner au moindre mouvement. Comme le balisage était bourré de code insensé, on se retrouvait fréquemment à effacer des balises par inadvertance. À la manière d'un mur dont on retirerait la brique du bas, c'est alors toute la mise en page qui s'effondrait. En outre, avec un code aussi complexe, la recherche des bogues était presque impossible. Il était souvent plus simple de reprogrammer complètement la page que

de chercher à la rapiécer. Les choses se compliquaient encore avec les sites de grande taille : leur présentation étant verrouillée dans des pages individuelles, il fallait élaborer de très complexes routines de recherche et de remplacement pour répercuter la plus infime modification dans l'ensemble du site. J'ai moi-même détérioré plusieurs sites en mon temps avec des routines de ce genre. Les modèles de pages se désynchronisaient très facilement et un simple ajustement pouvait nécessiter d'intervenir patiemment sur chacune des pages du site.

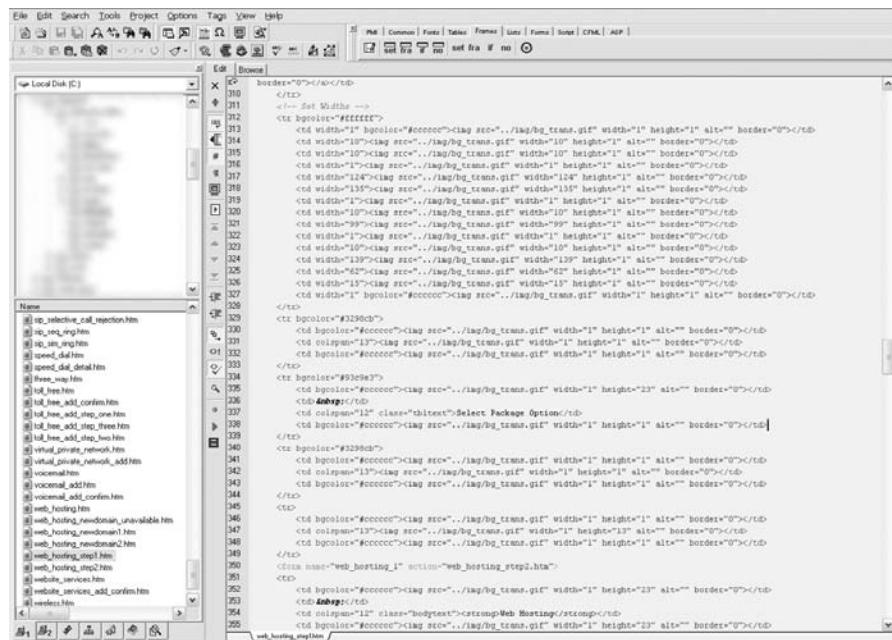
Comme les tableaux n'ont jamais été conçus pour la mise en page, il a fallu que l'ingénieux David Siegel invente un hack pour les faire fonctionner. Afin de les empêcher de s'effondrer en hauteur ou en largeur, il a imaginé d'utiliser une petite image GIF transparente d'un pixel. En l'insérant dans les cellules des tableaux, puis en la redimensionnant verticalement et horizontalement, il parvint à contraindre artificiellement la hauteur et la largeur des cellules, afin de préserver les proportions de la mise en page. Ces "user de shim.gif", comme les nommait Dreamweaver, étaient monnaie courante dans les anciennes maquettes tabulaires. Fort heureusement, cette pratique est maintenant révolue et vous ne verrez plus ces petits éléments de présentation contaminer votre code.

Au lieu d'être considéré comme un simple langage de balisage, le HTML a acquis la réputation d'un langage compliqué, déroutant et très facilement sujet aux erreurs. De nombreuses personnes craignaient de toucher au code. Cette tendance a favorisé une dépendance excessive vis-à-vis des éditeurs visuels et contribué à forger une génération de concepteurs peu aguerris aux techniques de programmation.

À l'orée du nouveau millénaire, le secteur de la conception web se trouvait dans un état de délabrement avancé. Il fallait faire quelque chose.

Figure 1.2

Cette capture d'écran du logiciel Homesite présente une maquette complexe, réalisée avec des tableaux et des GIF d'espacement (merci à Jeff L.).

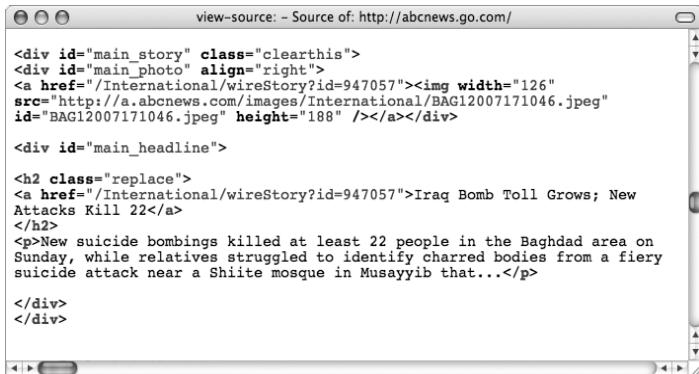


C'est dans cet état de marasme complet que sont arrivées les feuilles de styles CSS. Elles offraient aux utilisateurs la possibilité de contrôler l'apparence des pages avec des feuilles de styles externes et donc de séparer le contenu de la présentation. Des changements portant sur tout le site pouvaient ainsi s'effectuer à un seul et unique endroit et se trouver propagés dans tout le système. Les balises de présentation comme les balises de police furent abandonnées et la mise en page fut contrôlée avec des CSS et non des tableaux. Le balisage put alors retrouver sa simplicité d'origine et l'on commença à retrouver de l'intérêt au code sous-jacent.

La signification des éléments a pu naturellement retrouver sa place au sein des documents. Les styles par défaut des navigateurs pouvant être redéfinis, il était possible de baliser un élément afin d'indiquer qu'il s'agissait d'un titre, sans nécessairement avoir à le rendre gros, gras et très moche. On put créer des listes sans se retrouver avec des puces et des balises de citation et être instantanément punis par leur habituel effet de mise en forme. Les développeurs commencèrent alors à utiliser les éléments HTML pour ce qu'ils signifiaient et non pour l'apparence qu'ils permettaient d'obtenir (voir Figure 1.3).

Figure 1.3

Le balisage de l'article à la une sur [abcnews.com](http://abcnews.go.com) tel qu'on pouvait le voir cette année est bien structuré et facile à comprendre. S'il contient encore quelques balises de présentation, l'amélioration est considérable par rapport au code de la Figure 1.1.



```
<div id="main_story" class="clearthis">
<div id="main_photo" align="right">
<a href="/International/wireStory?id=947057"></a></div>
<div id="main_headline">
<h2 class="replace">
<a href="/International/wireStory?id=947057">Iraq Bomb Toll Grows; New
Attack Kill 22</a>
</h2>
<p>New suicide bombings killed at least 22 people in the Baghdad area on
Sunday, while relatives struggled to identify charred bodies from a fiery
suicide attack near a Shiite mosque in Musayyib that...</p>
</div>
</div>
```

Importance de la signification

Un balisage qui a du sens, voilà qui offre aux développeurs plusieurs avantages importants. Les pages dites "sémantiques" sont plus faciles à gérer que les simples pages de présentation. Supposons ainsi que vous deviez modifier une citation dans une page. Si cette citation est correctement balisée, vous pouvez aisément parcourir le code jusqu'au premier élément `blockquote`. Si la citation est un élément de paragraphe comme les autres, elle est en revanche bien plus difficile à trouver. Considérons un exemple plus complexe, mais pas moins réaliste : vous devez ajouter une colonne à votre page d'accueil. Avec les CSS, vous n'aurez qu'à déposer le contenu au bon endroit puis à mettre à jour les largeurs des colonnes dans le fichier CSS. Pour obtenir le même résultat dans une maquette tabulaire, il faut ajouter une colonne au tableau, modifier les valeurs `colspan` des précédentes colonnes, altérer les largeurs de toutes les cellules et modifier les largeurs de toutes les images GIF d'espace-ment. Toute la structure de la page doit être modifiée pour s'adapter à ce simple changement.

En plus d'être simple à comprendre par les hommes, le balisage sémantique peut être compris par les programmes ou d'autres machines. Les moteurs de recherche, par exemple, peuvent

reconnaître les titres parce qu'ils sont entourés de balises `h1` et leur conférer l'importance qu'ils méritent. Les utilisateurs qui se servent de lecteurs d'écran peuvent s'appuyer sur les titres pour faciliter la navigation dans les pages.

Ce qui nous intéresse au premier chef dans ce livre, c'est que le balisage sémantique offre un moyen simple de cibler les éléments à mettre en forme. Il donne une structure au document et crée un cadre sous-jacent à partir duquel les autres éléments peuvent être construits. Il permet ainsi de mettre en forme des éléments sans avoir à ajouter d'autres identifiants et donc à surcharger votre code. Le HTML inclut une grande variété d'éléments signifiants, comme :

- `h1, h2, ... ;`
- `ul, ol et dl ;`
- `strong et em ;`
- `blockquote et cite ;`
- `abbr, acronym et code ;`
- `fieldset, legend et label ;`
- `caption, thead, tbody et tfoot.`

Chaque fois qu'un élément signifiant existe et qu'on peut l'utiliser, il est toujours judicieux de le faire.

La querelle qui oppose les CSS aux tableaux semble refleurir périodiquement sur les blogs, les listes de discussion ou les forums de développeurs. Ce débat ressurgit chaque fois qu'un développeur habitué aux tableaux rechigne à l'idée de devoir assimiler de nouvelles connaissances. Cette réaction est assez naturelle et je conçois qu'une mise en page en CSS puisse paraître difficile de prime abord, surtout quand tout semble bien fonctionner avec les méthodes actuelles. Les avantages des CSS ont cependant été détaillés un nombre incalculable de fois : code moins surchargé, téléchargement plus rapide, maintenance plus facile – la liste entière serait longue. La plupart des développeurs professionnels ont constaté l'intérêt de s'appuyer sur les standards du Web et il est rare de voir une agence, quels que soient sa taille ou son niveau de sérieux, travailler avec l'ancienne méthode. Si vous utilisez toujours des maquettes tabulaires, vous aurez donc énormément de mal à travailler avec des agences. Ces vieilles habitudes se meurent heureusement et une nouvelle génération de développeurs apparaît qui n'a jamais eu à souffrir des désagréments liés à ces méthodes de travail.

ID et noms de classes

Les éléments sémantiques constituent une excellente base, mais la liste des éléments disponibles n'est pas exhaustive. Le HTML 4 a été créé pour servir de langage de balisage de document simple plutôt que comme un langage d'interface. Du coup, les éléments dédiés à certains usages spécifiques, comme les zones de contenu ou les barres de navigation, n'existent pas. Vous pouvez créer vos propres éléments en XML, mais pour un tas de raisons qu'il est trop long de détailler ici, cette méthode n'est pas très commode.

Le HTML 5 se promet de résoudre ces problèmes en proposant aux développeurs un jeu d'éléments plus riche. Parmi ceux-ci, des éléments de structure comme `header`, `nav`, `article`, `section` et `footer`, ainsi que de nouvelles fonctionnalités d'interface utilisateur comme des éléments de saisie de données et de menu. En prévision du HTML 5, de nombreux développeurs ont déjà commencé à adopter cette nomenclature dans les conventions de nom pour leurs ID et leurs noms de classes.

Le mieux, en attendant, est d'utiliser les éléments existants et de leur donner une signification supplémentaire avec un ID ou un nom de classe. On ajoute ainsi une structure au document, qui fournit des connecteurs utiles pour les styles. En ajoutant l'ID `nav` à une simple liste de liens, vous pouvez créer votre propre élément de navigation personnalisé.

```
<ul id="nav">
  <li><a href="/home/">Home</a></li>
  <li><a href="/about/">About Us</a></li>
  <li><a href="/contact/">Contact</a></li>
</ul>
```

L'ID sert à identifier un élément spécifique, comme la barre de navigation d'un site, et doit être unique dans la page. Il est utile pour identifier des éléments structurels fixes comme les zones de navigation ou de contenu principales. Il est aussi utile pour identifier des éléments uniques – un lien ou un élément de formulaire particuliers, par exemple.

S'il n'est possible d'appliquer qu'un seul nom d'ID à un élément dans une page, on peut en revanche appliquer le même nom de classe à n'importe quel nombre d'éléments. C'est tout l'intérêt des classes. Elles sont utiles pour identifier des types de contenu ou des éléments similaires. Par exemple, votre page peut contenir de nombreux articles de news.

```
<div id="story-id-1">
  <h2>Salter Cane win Best British Newcomer award</h2>
  <p>In a surprise turn of events, alt folk group, Salter Cane, won Best
    ↪ British Newcomer and the Grammys this week...</p>
</div>

<div id="story-id-2">
  <h2>Comic Sans: The Movie wins best documentary at the BAFTAs </h2>
  <p>The story of this beloved typeface one the best documentary category.
    ↪ Director Richard Rutter was reported to be speechless...</p>
</div>
```

Au lieu d'attribuer un ID séparé à chaque article, vous pouvez associer la classe `news` à chacun d'entre eux.

```
<div class="news">
  <h2>Salter Cane win Best British Newcomer award</h2>
  <p>In a surprise turn of events, alt folk group, Salter Cane, won Best
    ↪ British Newcomer and the Grammys this week...</p>
</div>

<div class="news">
  <h2>"Comic Sans: The Movie" wins best documentary at the BAFTAs </h2>
  <p>The story of this beloved typeface one the best documentary category.
    ↪ Director Richard Rutter was reported to be speechless...</p>
</div>
```

Nommer les éléments

Lorsque vous nommez les ID et les classes, il est important que les noms choisis soient aussi indépendants que possible de la présentation. Par exemple, si vous souhaitez que vos messages de notification de formulaire s'affichent en rouge, vous pouvez certes leur attribuer une classe nommée rouge. Ce choix convient tant qu'il n'y a pas d'autre élément rouge dans la page, mais si vous souhaitez que les étiquettes du formulaire soient rouges également, vous allez devoir deviner à quel élément associer cette classe et cela commencera à devenir confus. Imaginez à quel point la situation deviendrait déroutante si vous utilisiez des éléments de présentation dans tout votre site ! Les choses se compliqueraient encore si vous décidiez de changer l'apparence des notifications de formulaire pour les afficher en jaune et non en rouge. Vous devriez alors modifier tous les noms de classes ou vous contenter d'utiliser une classe appelée rouge pour afficher les éléments en jaune...

Il est en fait préférable de nommer les éléments en fonction de ce qu'ils sont plutôt que de l'apparence qu'ils possèdent. Le code devient ainsi plus compréhensible et ne risque jamais de se désynchroniser avec la mise en page. Dans l'exemple précédent, au lieu d'attribuer la classe rouge aux notifications, vous pouvez leur donner un nom plus descriptif, comme `.avertissement` ou `.notification` (voir Figure 1.4). L'aspect le plus intéressant des noms descriptifs tient à ce que vous pouvez les utiliser dans tout le site. Par exemple, vous pouvez aussi utiliser la classe `.notification` avec d'autres types de messages et leur attribuer un style complètement différent en fonction de l'endroit où ils se trouvent dans le document.

Figure 1.4

Bons et mauvais noms d'ID.

Mauvais noms	Bons noms
rouge colonneGauche navHaut premierPara	erreur contenuSecondaire navPrincipale intro

Lorsque vous écrivez des noms de classe et d'ID, soyez attentif à la casse, car les navigateurs considèrent que `.nimportequi` est une autre classe que `.nimporteQui`. Le meilleur moyen de gérer ce problème consiste à respecter systématiquement la même convention de nom. Pour ma part, j'écris tous mes noms d'ID et de classes en minuscules, avec des tirets pour séparer les mots. C'est une écriture plus lisible. `nimporte-qui` est ainsi plus lisible que `nimporteQui`.

ID ou classes ?

Il est parfois difficile de savoir si un élément doit posséder un ID ou un nom de classe. En règle générale, les classes doivent s'appliquer aux éléments de nature similaires, qui peuvent apparaître à plusieurs endroits dans la même page, alors que les ID doivent s'appliquer aux éléments uniques. Reste dès lors à savoir quels éléments sont par nature similaires et lesquels sont uniques.

Par exemple, imaginez que votre site contienne un système de navigation principal dans l'en-tête, un système de navigation paginé en bas de la page des résultats de recherche et un troisième système de navigation dans le pied de page. Allez-vous donner à chacun de ces éléments un ID distinct, comme `main-nav`, `page-nav` et `footer-nav` ou allez-vous attribuer à chacun la classe `nav` et les mettre en forme en fonction de leur position dans le document ? Auparavant, j'avais plutôt tendance à préférer la première méthode, qui me paraissait offrir un ciblage plus défini. Malheureusement, elle présente aussi des désavantages. Que se passe-t-il si vous décidez finalement d'afficher le système de navigation paginé à la fois en haut et en bas des résultats de la recherche ou que vous souhaitez créer un système de navigation à deux niveaux dans le pied de page ?

Si vous utilisez beaucoup d'ID, vous serez rapidement à court de noms et vous finirez par développer des conventions de noms extrêmement longues et compliquées. C'est pour cette raison que je privilégie aujourd'hui les noms de classes et n'emploie un ID que lorsque je cible un élément très spécifique et que je ne risque jamais de me resservir de ce nom pour autre chose autre part dans le site. Vous ne devez en fait utiliser un ID que lorsque vous avez la certitude que l'élément n'apparaîtra qu'une seule fois. Si vous envisagez d'employer des éléments similaires à l'avenir, préférez plutôt une classe. En vous tenant à des conventions de noms générales et en utilisant des classes, vous éviterez de créer de longues chaînes de sélecteurs d'ID qui renvoient toutes à des styles similaires.

```
#andy, #rich, #jeremy, #james-box, #cennydd, #paul, #natalie, #sophie {  
    font-size: 1.6em;  
    font-weight: bold;  
    border: 1px solid #ccc;  
}  
Il suffit alors de créer une classe générique pour l'ensemble.  
.staff {  
    font-size: 1.6em;  
    font-weight: bold;  
    border: 1px solid #ccc;  
}
```

Grâce à leur flexibilité, les classes peuvent être d'une redoutable efficacité. En contrepartie, on peut aussi facilement en abuser. Bien souvent, les débutants en CSS en ajoutent à tout ce qu'ils rencontrent, afin de pouvoir contrôler aussi précisément que possible leurs styles. Les premiers éditeurs WYSIWYG avaient également tendance à ajouter des classes chaque fois qu'un style était appliqué. De nombreux développeurs ont contracté cette mauvaise habitude, parce qu'ils ont appris les CSS en observant le code généré par les logiciels. C'est ce qu'on appelle une "classite aiguë" ; c'est presque aussi mauvais que la manie de tableaux, car cela a pour effet d'ajouter du code dépourvu de signification au document.

```
<h2 class="news-head">Andy wins an Oscar for his cameo in Iron Man</h2>  
<p class="news-text">  
    Andy Budd wins the Oscar for best supporting actor in Iron Man      after his  
    ➔ surprise cameo sets Hollywood a twitter with speculation.  
</p>  
<p class="news-text"><a href="news.php" class="news-tink">More</a></p>
```

Dans cet exemple, on identifie chaque élément comme partie d'un article de news en utilisant un nom de classe lié aux news. Le but ici est de faire en sorte que les titres et le texte

possèdent un style différent du reste de la page, mais il n'est pas nécessaire d'utiliser toutes ces classes supplémentaires pour cibler chaque élément individuel. Vous pouvez identifier l'ensemble du bloc comme un élément de news en le plaçant dans une div (du code) portant le nom de classe news. Ensuite, les titres ou le texte peuvent être ciblés grâce aux propriétés de cascade des feuilles de styles.

```
<div class="news">
  <h2>Andy wins an Oscar for his cameo in Iron Man </h2>
  <p>Andy Budd wins the Oscar for best supporting actor in Iron Man after
  ↪ his surprise cameo sets Hollywood a twitter with speculation.</p>
  <p><a href="news.php">More</a></p>
</div>
```

Chaque fois que vous vous retrouvez à répéter des mots dans les noms de classes (comme news-head et news-link ou section-head et section-foot), demandez-vous si ces éléments ne peuvent pas être décomposés en leurs parties constituantes. Votre code serait alors mieux décomposé et donc plus flexible.

En supprimant les classes superflues, vous simplifieriez le code et réduirez le poids de vos pages. Je reviendrai plus tard sur les sélecteurs CSS et le ciblage des styles, mais il n'est presque jamais nécessaire de dépendre à ce point des noms de classes. Si vous ajoutez un grand nombre de classes, c'est sans doute le signe que votre document HTML est mal structuré.

div et span

L'élément div est l'un des premiers qui puisse contribuer à donner une structure aux documents. De nombreuses personnes pensent – à tort – que l'élément div ne possède pas de signification. La balise div marque pourtant une division et fournit un moyen de décomposer les documents en zones bien définies. En plaçant la zone de contenu principal dans une div et en lui attribuant la classe content, vous structurez votre document de manière sémantique.

Pour limiter le plus possible le balisage, vous ne devez utiliser d'élément div que lorsqu'il n'existe pas déjà un élément distinctif. Par exemple, si vous vous servez d'une liste pour votre navigation principale, il n'est pas nécessaire de l'envelopper dans une div.

```
<div class="nav">
  <ul>
    <li><a href="/home/">Home</a></li>
    <li><a href="/about/">About Us</a></li>
    <li><a href="/contact/">Contact</a></li>
  </ul>
</div>
```

Vous pouvez supprimer la div et appliquer votre classe directement à la liste :

```
<ul class="nav">
  <li><a href="/home/">Home</a></li>
  <li><a href="/about/">About Us</a></li>
  <li><a href="/contact/">Contact</a></li>
</ul>
```

La "divite aiguë" (l'utilisation abusive de balises `div`) est souvent le symptôme d'un code mal structuré et excessivement compliqué. Certains débutants en CSS tentent, en fait, de dupliquer leur ancienne structure tabulaire avec des `div`. Malheureusement, cela revient à remplacer un jeu de balises superflues par un autre. Au lieu de cela, les `div` doivent être utilisées pour regrouper des éléments liés par leur signification ou leur fonction plutôt que par leur présentation ou leur disposition.

Si les `div` peuvent être utilisées pour regrouper des éléments de niveau bloc, les `span` servent à regrouper ou à identifier des éléments incorporés dans les lignes :

```
<h2>Andy wins an Oscar for his cameo in Iron Man </h2>
<p>Published on <span class="date">February 22nd, 2009</span>
by <span class="author">Harry Knowles</span></p>
```

Le but est de conserver un code aussi épuré et structuré que possible ; néanmoins, il est parfois impossible d'éviter d'y ajouter quelques éléments `div` ou `span` non sémantiques pour obtenir l'affichage souhaité. Si c'est le cas, ne vous inquiétez pas outre mesure. La période que nous traversons correspond encore à une phase de transition et la norme CSS 3 promet de nous offrir un bien meilleur contrôle des documents. En attendant, il faut souvent faire quelques concessions et enfreindre la théorie. Le principal est de savoir quand faire des compromis et de les faire pour de bonnes raisons.

Microformats

Comme le HTML est assez pauvre en éléments, le marquage de certains types d'informations, comme les personnes, les lieux ou les dates, est souvent difficile. Pour remédier à cela, un groupe de développeurs a décidé de créer un ensemble de conventions de noms et de systèmes de balisage standardisés pour représenter ces données. Ces conventions de noms, qui s'appuient sur des formats de données existants comme vCard et iCalendar, se sont fait connaître sous l'appellation de "microformats". À titre d'exemple, voici mes informations de contact, balisées au format hCard :

```
<div class="vcard">
<p><a class="url fn" href="http://andybudd.com/">Andy Budd</a>
  <span class="org">Clearleft Ltd</span>
  <a class="email" href="mailto:info@andybudd.com">info@andybudd.com</a>
</p>
<p class="adr">
  <span class="locality">Brighton</span>,
  <span class="country-name">England</span>
</p>
</div>
```

Les microformats permettent de coder des données de manière à les rendre accessibles à d'autres programmes et services. Certaines personnes ont écrit des scripts capables d'extraire des informations d'événements écrites au format hCalendar et de les importer directement dans une application de calendrier (voir Figures 1.5 et 1.6).

The screenshot shows a web browser displaying the UX London Programme website. The page is titled 'Day 2 — Workshops'. The schedule is organized into three columns for 'Wed 19th May', 'Thu 20th May', and 'Fri 21st May'. Each day has a header and a list of workshops with descriptions and speakers. The workshops are as follows:

- Wednesday 19th May:**
 - 9:00 – 12:30: **Get Visible** by **Liz Danzico**. Description: In this workshop, you'll learn strategies you can apply to your work, so you can get past designer's block and communicate ideas with visible clarity.
- Thursday 20th May:**
 - 9:00 – 12:30: **How To Think With Pretty Pictures (Demystifying Concept Models)** by **Stephen P. Anderson**. Description: Whether you're trying to explain something to a client or make sense of it for yourself, you'll find the techniques taught in this workshop invaluable.
 - 12:30 – 13:30: **Lunch**
 - 13:30 – 17:00: **Real-World Agile User Experience Design** by **Jeff Patton**. Description: This workshop is about real-world user experience design practices that have been emerging from agile and agile-like contexts over the last decade.
 - 13:30 – 17:00: **Information Architecture with Maps** by **Peter Morville**. Description: In this workshop, we'll explore how maps can improve the process and product of classic and cross-media information architecture and user experience design.
 - 13:30 – 17:00: **Good Design Faster** by **Leah Buley**. Description: Highly recommended for anyone doing agile, iterative product development (and basically anyone who's tasked with doing more in less time)
- Friday 21st May:**
 - 9:00 – 12:30: **Knowledge Games: Design practices for systems thinking and co-creation** by **Dave Gray**. Description: You will learn the ten essentials of knowledge games, a basic toolkit for knowledge-game designers. And together we will design and experience a knowledge game.
 - 12:30 – 13:30: **Lunch**
 - 13:30 – 17:00: **Content Strategy: The Missing Piece of the UX Puzzle** by **Designing for the Usage Lifecycle**. Description: This workshop is about real-world user experience design practices that have been emerging from agile and agile-like contexts over the last decade.
 - 13:30 – 17:00: **DRY UX** by **Guerilla Usability Testing**. Description: This workshop is about real-world user experience design practices that have been emerging from agile and agile-like contexts over the last decade.

At the bottom of the page, there is a link to 'Add the schedule to your calendar' with a 'Subscribe' button.

Figure 1.5

Le calendrier de la conférence UX London est balisé au format hCalendar.

The screenshot shows a detailed hCalendar view for the week of May 17-21, 2010. The calendar grid shows the days of the week (17 to 21) and the hours of the day (07:00 to 15:00). Each cell in the grid represents a specific time slot. The content of these cells is as follows:

Jour	Sémaine	Mois	0 Afficher la semaine de travail	1 Afficher la semaine entière	Rechercher UX London: Programme
17	17 - 21 mai 2010	18	19	20	21
17		mardi	mercredi	jeudi	vendredi
17			UX London: Cumberland Hotel, London		
07:00					
08:00					
09:00			Design for Engagement		
10:00			Search Patterns: The Future of Discovery		
11:00			Break		
12:00			Metrics-driven Design		
13:00			Designing for Improvisation		
14:00			Lunch		
15:00			To Be Announced		
			Experiencing Comics		
			Break		
Tâches			Afficher les tâches pour : Échéance		

The 'Afficher la semaine entière' checkbox is checked. The 'Rechercher UX London: Programme' search bar is present at the top right of the calendar grid.

Figure 1.6

Cela signifie que les visiteurs peuvent ajouter tout l'emploi du temps à leur application de calendrier en un clic.

D'autres personnes ont écrit des plug-ins qui permettent à Firefox d'extraire des informations de contact au format hCard et de les envoyer vers un téléphone portable *via* Bluetooth (voir Figures 1.7 et 1.8).

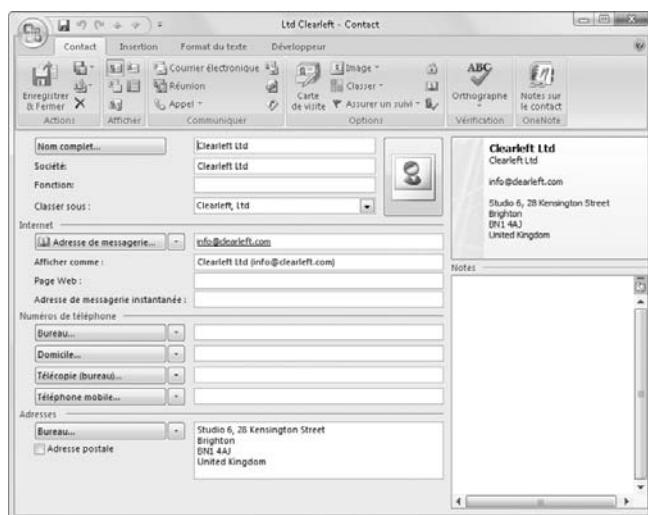
Figure 1.7

Les informations de contact dans le site web Clearleft sont également balisées au format hCard.



Figure 1.8

Avec Outlook (ci-dessus), Operator ou l'ancien complément Tails de Firefox, vous pouvez importer ces informations de contact directement dans votre carnet d'adresses.



Il existe neuf microformats officiels à cette date et quatorze formats à l'état de brouillon, dont :

- **hCalendar** pour les dates, les calendriers et les événements ;
- **hCard** pour les personnes et les organisations ;
- **XFN** pour les relations entre personnes ;
- **hProduct** pour les descriptions de produits (brouillon) ;
- **hRecipe** pour les ingrédients et les recettes (brouillon) ;
- **hReview** pour les critiques de produits et d'événements (brouillon) ;
- **hAtom** pour le contenu épisodique, comme les articles de blogs (brouillon).

Beaucoup de sites web importants prennent déjà en charge les microformats. C'est le cas de Google Maps, qui utilise le format hCard pour les informations d'adresse dans ses résultats de recherche. Yahoo! prend également en charge les microformats sur un certain nombre de propriétés du site de partage de photos Flickr. Yahoo! a d'ailleurs lâché pas moins de vingt-six millions de microformats en pleine nature, en utilisant le format hListing pour le moteur de comparaisons de prix Kelkoo. Il est extrêmement facile d'ajouter des données microformatées à un site web. Je vous recommande donc de le faire chaque fois que c'est possible.

Il ne s'agit là que d'un aperçu très superficiel des possibilités offertes par les microformats. Si vous souhaitez en apprendre plus à ce sujet, vous pouvez lire l'article anglais "Microformats: Empowering Your Mark-up for Web 2.0" de John Allsopp. Sinon, rendez-vous sur <http://microformats.org> pour consulter les spécifications officielles.

Les différentes versions du HTML et des CSS

Les CSS existent en plusieurs versions (ou niveaux). Il est donc important de savoir laquelle utiliser. La spécification CSS 1, devenue une recommandation à la fin de 1996, incluait des propriétés très rudimentaires comme les polices, les couleurs et les marges. La spécification CSS 2, sortie en 1998, intégrait des concepts avancés tels le flottement et le positionnement, ainsi que de nouveaux sélecteurs comme les sélecteurs d'enfants, les sélecteurs de frères adjacents et les sélecteurs universels.

Le temps s'écoule à son propre rythme dans l'univers du W3C (*World Wide Web Consortium*), si bien qu'en dépit du fait que le travail sur la spécification CSS 3 a débuté avant l'an 2000 sa promulgation officielle n'est pas encore à l'ordre du jour. Pour accélérer le développement et l'implémentation dans les navigateurs, la spécification CSS 3 a donc été décomposée en modules qui peuvent être rendus officiels et implémentés indépendamment. Elle inclut de passionnantes ajouts, dont un module de mise en page avancé, de toutes nouvelles propriétés d'arrière-plan et une flopée de nouveaux sélecteurs. Quelques-uns de ces modules sont planifiés pour la seconde moitié de 2009. Malheureusement, il s'agit de stades que nous avons déjà connus et il est arrivé que des modules, sur le point d'être publiés, soient finalement ramenés à l'état de "working drafts" (brouillons). Il est donc difficile de savoir combien d'entre eux vont vraiment franchir ce cap. On peut espérer que certains passeront au statut de recommandation officielle d'ici 2011. La mauvaise nouvelle, c'est que certains semblent ne pas avoir été seulement commencés, tandis que d'autres n'ont pas été mis à jour depuis plusieurs années. À un tel rythme, on a peu de raisons de croire que la spécification CSS 3 parviendra à être totalement complète un jour.

La bonne nouvelle, c'est que, malgré les retards, beaucoup d'éditeurs de navigateur ont déjà implémenté certaines des parties les plus intéressantes de la spécification CSS 3. Il est donc déjà possible de commencer à utiliser un grand nombre de ces sélecteurs.

En raison du long délai prévu entre la sortie officielle de la CSS 2 et celle de la CSS 3, un travail a démarré en 2002 sur la CSS 2.1. Cette révision de la CSS 2 a pour but de corriger certaines erreurs et de proposer une présentation plus précise de l'implémentation des CSS dans les navigateurs. La CSS 2.1 parvient lentement mais sûrement à maturité et correspond donc à la version CSS aujourd'hui recommandée.

Le HTML 4.01 est passé au stade de recommandation à la fin de 1999. C'est la version que la plupart des personnes utilisent. En janvier 2000, le W3C a créé une version XML du HTML 4.01 et l'a nommée XHTML 1.0. La principale différence entre le XHTML 1.0 et le HTML 4.01 tient au fait que le premier respecte les conventions de programmation XML. Contrairement au HTML standard, tous les attributs XHTML doivent contenir des guillemets et tous les éléments doivent être fermés. Le code suivant, qui correspond à du HTML valide, est incorrect pour le XHTML :

```
<h2>Peru Celebrates Guinea Pig festival
<p><img src=pigonastick.jpg alt=Roast Guinea Pig>
<p>Guinea pigs can be fried, roasted, or served in a casserole.
```

En XHTML 1.0, ce code s'écrirait de la manière suivante :

```
<h2>Peru Celebrates Guinea Pig festival</h2>
<p></p>
<p>Guinea pigs can be fried, roasted, or served in a casserole.</p>
```

Le XHTML 1.1 avait pour objectif de rapprocher encore le XHTML du XML. Il existait en pratique très peu de différences entre les deux langages, à l'exception d'une, mais de taille : s'il était toujours acceptable de servir une page XHTML 1.0 sous forme de document HTML, les pages XHTML 1.1 étaient supposées être envoyées aux navigateurs comme s'il s'agissait de code XML. Du coup, si elles contenaient la moindre erreur, comme un simple caractère accentué mal encodé, les navigateurs web n'étaient pas censés les afficher. C'était évidemment tout sauf idéal pour la plupart des propriétaires de sites web, si bien que le XHTML 1.1 n'a jamais vraiment décollé.

On débat encore aujourd'hui pour savoir s'il faut ou non servir les pages XHTML 1.0 comme du HTML ou s'il faut s'en tenir au HTML 4.01. Il est clair toutefois qu'il ne faut pas utiliser du XHTML 1.1, à moins de choisir le type MIME approprié et de ne pas voir d'inconvénient au fait que la page ne s'affiche pas si elle contient la moindre erreur.

Le HTML 5 est assez nouveau. Comme sa spécification possède encore le statut de brouillon, il continue de changer fréquemment. Il connaît pourtant déjà un vrai succès et plusieurs navigateurs importants commencent à le prendre en charge. Il est né de la frustration que les développeurs avaient accumulée avec le développement lent et archaïque du XHTML 2. Une poignée d'entre eux a donc décidé d'esquisser leur propre spécification. Le succès fut tel que le HTML 5 est devenu un projet officiel du W3C et que le développement du XHTML 2 a été suspendu.

Le but du HTML 5 est de produire un langage de balisage moderne qui corresponde mieux au type d'informations publiées sur le Web. Il introduit donc de nouveaux éléments structurels comme `header`, `nav`, `article`, `sections` et `footer`. Il contient également un ensemble de nouveaux éléments de formulaire qui devraient faciliter considérablement le développement d'applications web.

Types de document, commutation de DOCTYPE et modes des navigateurs

Une DTD, ou définition de type de document, est un ensemble de règles, lisibles par ordinateur, qui définissent ce qui est, ou non, autorisé dans une version donnée du HTML ou du XML. Les navigateurs sont censés les utiliser lorsqu'ils analysent les pages web pour vérifier leur validité et pour les traiter. Pour savoir quelle DTD employer, et donc connaître votre version du HTML, les navigateurs analysent la déclaration DOCTYPE de la page.

Il s'agit d'un bloc de une ou deux lignes de code, situé au début du document HTML et qui décrit la DTD utilisée. Dans cet exemple, la DTD est celle du XHTML 1.0 Strict :

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

Les déclarations DOCTYPE contiennent généralement (mais pas toujours) une URL vers le fichier de DTD spécifié. Le HTML 5, par exemple, ne requiert pas d'URL. Le plus souvent, les navigateurs ne lisent pas ces fichiers et préfèrent reconnaître les déclarations DOCTYPE courantes.

Les DOCTYPE sont proposés en deux modes : strict et transitionnel. Transitionnels, ils sont destinés aux personnes qui opèrent une transition depuis d'anciennes versions du langage. Les versions transitionnelles du HTML 4.01 et du XHTML 1.0 permettent toujours d'utiliser des éléments déconseillés, comme l'élément `font`. Les versions strictes de ces langages interdisent au contraire l'utilisation des éléments déconseillés, afin de séparer clairement le contenu de la présentation.

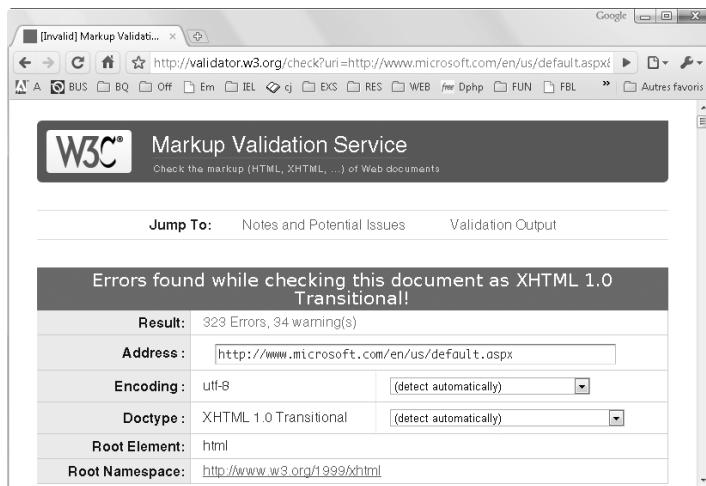
Validation

En plus de disposer d'un balisage sémantique, le document HTML doit être écrit avec du code valide. Si ce n'est pas le cas, les navigateurs tentent d'interpréter le code par eux-mêmes et commettent parfois des erreurs. Pire : si un document XHTML est envoyé avec le type MIME approprié, les navigateurs qui comprennent le XML ne l'affichent tout simplement pas quand la page est invalide. Comme les navigateurs doivent savoir quelle DTD utiliser pour traiter correctement la page, il faut une déclaration DOCTYPE pour valider la page.

Pour vérifier que le HTML est valide, vous pouvez utiliser le validateur du W3C, un bookmarklet de validateur ou un plug-in comme l'extension Web Developer de Firefox. De nombreux éditeurs HTML disposent maintenant de validateurs intégrés et vous pouvez même installer sur votre ordinateur une copie locale du validateur du W3C. Le validateur indique si la page est valide et, si elle ne l'est pas, il en explique les raisons (voir Figure 1.9).

Figure 1.9

La page d'accueil de **microsoft.com** contient 323 erreurs HTML et 34 erreurs CSS.



La validation est importante car elle facilite le repérage des bogues dans le code. Il est donc judicieux de prendre l'habitude de la faire fréquemment et assez tôt. Elle n'est cependant pas une fin en soi, et bien des pages, qui se trouvent être par ailleurs très satisfaisantes, échouent à la validation à cause de petites erreurs comme des signes d'espérillette non encodés ou du contenu hérité. Si la validation est importante, il faut parfois aussi céder au bon sens pratique.

Il existe plusieurs sortes d'outils de validation. Vous pouvez vous rendre à l'adresse <http://validator.w3.org/>, où vous tapez votre URL. Si vous devez faire des validations fréquemment (ce qui serait en principe judicieux), il sera peut-être fastidieux de saisir chaque fois votre URL. Pour ma part, j'utilise un bookmarklet (ou favelet) de validation – un petit programme JavaScript stocké dans le dossier Favoris du navigateur. Quand je clique dessus, cela déclenche l'action JavaScript correspondante. Dans le cas du bookmarklet de validation, la page actuellement consultée est transmise au validateur du W3C et affiche le résultat. Vous trouverez le bookmarklet de validation et d'autres bookmarklets pratiques pour le développement web à l'adresse <http://favelets.com/>.

Si vous utilisez Firefox, vous pouvez télécharger et installer un grand nombre de plug-ins. Parmi ceux-ci, mon favori est l'extension Web Developer. En plus de permettre de valider le code HTML et CSS, elle offre la possibilité de réaliser une série d'autres tâches pratiques comme la démarcation graphique des différents éléments HTML, la désactivation des feuilles de styles et même l'édition des styles dans le navigateur. Elle peut être téléchargée à l'adresse <http://chrispederick.com/work/web-developer/>. Elle est indispensable à tous les développeurs CSS qui utilisent Firefox. L'autre excellent outil est l'extension Validator, téléchargeable à l'adresse <http://users.skynet.be/mgueury/mozilla/>.

Il existe aussi une barre d'outils de développeur pour Internet Explorer 6 et 7, qui peut être téléchargée à l'adresse <http://tinyurl.com/7mnyh>. Elle n'est pas aussi riche en fonctionnalités que la barre d'outils Firefox, mais tout de même très utile. Internet Explorer 8 inclut son propre jeu d'outils pour les développeurs, tout comme Safari 4.

Les déclarations DOCTYPE, si elles sont importantes pour la validation, sont aussi utilisées par les navigateurs à d'autres fins.

Modes des navigateurs

Quand les éditeurs de navigateurs ont commencé à créer des navigateurs compatibles avec les standards du Web, ils ont aussi souhaité assurer une compatibilité descendante. C'est la raison pour laquelle ils ont créé deux modes de rendu : le mode Standards et le mode Quirks (littéralement, "caprices"). En mode Standards, le navigateur reproduit la page conformément aux spécifications en vigueur. En mode Quirks, il l'affiche en prenant plus de liberté avec les spécifications mais en adaptant le rendu pour une meilleure compatibilité arrière. Le mode Quirks émule généralement le comportement des anciens navigateurs comme Internet Explorer 4 et Netscape Navigator 4 afin d'afficher correctement les anciens sites.

L'exemple le plus évident de la différence entre ces modes concerne Internet Explorer et le modèle de boîte propriétaire de Microsoft. À partir de sa version 6, Internet Explorer a utilisé le modèle de boîte universel en mode Standards et s'est mis à reproduire son ancien modèle de boîte propriétaire en mode Quirks. Pour assurer la compatibilité descendante avec les sites construits pour Internet Explorer 5 et ses versions antérieures, d'autres navigateurs, comme Opera 7 et ses versions ultérieures, ont également adopté, en mode Quirks, le modèle de boîte fautif d'Internet Explorer.

Il existe d'autres différences de rendu plus subtiles et propres à certains navigateurs. Ainsi, certains d'entre eux ne requièrent par exemple pas le symbole # pour exprimer les valeurs hexadécimales de couleur, presupposent que les longueurs sans unités sont calculées en pixels ou permettent de modifier les tailles des polices en spécifiant des mots-clés.

Mozilla et Safari possèdent un troisième mode appelé Almost Standards Mode (littéralement, "presque le mode Standards"). Il est identique au mode Standards, à quelques différences subtiles près, relatives à la gestion des tableaux.

L'extension Web Developer de Firefox permet de savoir dans quel mode une page s'affiche. Une marque de coche verte apparaît dans la barre d'outils si le site s'affiche en mode Standards. En mode Quirks, c'est une croix rouge qui apparaît. Les outils de développement d'Internet Explorer 8 signalent aussi le mode utilisé pour le rendu dans le navigateur.

Commutation de DOCTYPE

Le navigateur choisit la méthode de rendu à utiliser en fonction de l'existence d'une déclaration DOCTYPE et de la DTD utilisée. Si un document XHTML contient un DOCTYPE entièrement formé, il est normalement reproduit en mode Standards. Pour un document HTML 4.01, un DOCTYPE contenant une DTD stricte amènera généralement la page à s'afficher en mode Standards. Ce sera également le cas pour un DOCTYPE contenant une DTD transitionnelle et un URI, alors qu'une DTD transitionnelle sans URI la fera s'afficher en mode Quirks. Un DOCTYPE mal formé ou absent entraîne l'affichage des documents HTML et XHTML en mode Quirks.

Cet ajustement du mode de rendu en fonction de l'existence d'un DOCTYPE est connu sous le nom de "DOCTYPE switching" (commutation de DOCTYPE) ou "DOCTYPE sniffing" (reniflement du DOCTYPE). Ces règles, que tous les navigateurs ne respectent pas à la lettre, montrent bien comment fonctionne la commutation de DOCTYPE. Pour une liste

plus complète, consultez le tableau de la page <http://hsivonen.iki.fi/doctype/> qui montre les modes de rendu utilisés par les différents navigateurs selon la déclaration DOCTYPE choisie.

La commutation de DOCTYPE est un hack employé par les navigateurs pour distinguer les documents hérités des documents conformes aux standards. Même en écrivant du code CSS valide, si vous choisissez le mauvais DOCTYPE, vos pages s'affichent en mode Quirks et se comportent de manière imprévisible. Il est donc important d'inclure une déclaration DOCTYPE entièrement formée dans chaque page de votre site et de choisir une DTD stricte lorsque vous utilisez du HTML.

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

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

<!DOCTYPE html>
```

De nombreux éditeurs HTML ajoutent automatiquement les déclarations DOCTYPE aux documents. Si vous créez un document XHTML, certains logiciels plus anciens peuvent également ajouter une déclaration XML avant la déclaration DOCTYPE :

```
<?xml version="1.0" encoding="utf-8"?>
```

La déclaration XML est une déclaration facultative utilisée par les fichiers XML pour définir différents paramétrages comme la version du XML et le type d'encodage des caractères. Malheureusement, Internet Explorer 6 repasse automatiquement au mode Quirks si la déclaration DOCTYPE n'est pas le premier élément dans la page. Ce problème a été réparé dans Internet Explorer 7, mais à moins que vous ne serviez vos pages sous forme de documents XML, mieux vaut éviter d'utiliser une déclaration XML.

En résumé

Vous avez vu comment les conventions de nommage et le balisage sémantique pouvaient rendre le code plus facile à lire et à gérer. Vous avez également appris la différence entre les ID et les noms de classes et vu comment les utiliser. Vous avez découvert les différentes versions des langages CSS et HTML et la manière dont les navigateurs s'y prennent pour gérer ces différences.

Au chapitre suivant, vous allez passer en revue certains des sélecteurs CSS de base et découvrir tout un ensemble de nouveaux sélecteurs CSS 3. Vous apprendrez tout sur les concepts de spécificité et de cascade et vous verrez comment organiser et planifier les feuilles de styles pour une maintenance simplifiée.

2

Des styles qui atteignent leur cible

Un document valide et bien structuré fournit les fondations sur lesquelles les styles peuvent s'appliquer. Pour pouvoir mettre en forme un élément HTML donné avec des CSS, il vous faut un moyen de le cibler. Dans les CSS, la partie de la règle de style qui réalise cette opération est appelée le *sélecteur*.

- Au cours de ce chapitre, vous découvrirez :
- les sélecteurs courants ;
- les sélecteurs avancés ;
- les nouveaux sélecteurs CSS 3 ;
- les merveilleux concepts de spécificité et de cascade ;
- comment planifier et gérer les feuilles de styles ;
- comment commenter le code.

Les sélecteurs courants

Les sélecteurs les plus courants sont les sélecteurs de type et les sélecteurs descendants. Les premiers sont utilisés pour cibler un type d'élément particulier, comme un élément de paragraphe ou de titre. Vous devez simplement spécifier le nom de l'élément que vous souhaitez mettre en forme. Les sélecteurs de type sont parfois aussi appelés sélecteurs d'élément ou sélecteurs simples.

```
p {color: black;}  
h1 {font-weight: bold;}
```

Les sélecteurs descendants permettent de cibler les descendants d'un élément ou d'un groupe d'éléments donnés. Ils sont indiqués par un espace entre deux autres sélecteurs. Dans l'exemple suivant, seuls les éléments de paragraphe qui sont des descendants d'un élément `blockquote` se voient appliquer un retrait. Tous les autres paragraphes restent ne sont pas affectés.

```
blockquote p {padding-left: 2em;}
```

Ces deux types de sélecteurs sont particulièrement intéressants pour appliquer des styles génériques qui se retrouvent dans l'ensemble d'un site. Pour être plus spécifique et cibler des éléments particuliers, vous pouvez utiliser des sélecteurs d'ID et de classe, qui ciblent les éléments possédant l'ID ou le nom de classe correspondant. Les sélecteurs d'ID s'identifient à l'aide d'un caractère dièse, ceux de classe, à l'aide d'un point. Dans l'exemple suivant, la première règle passe en gras le texte du paragraphe d'introduction et la seconde passe la date en gris :

```
#intro {font-weight: bold;}  
.date-posted {color: #ccc;}
```

```
<p id="intro">Happy Birthday Andy</p>
<p class="date-posted">24/3/2009</p>
```

Nombreux sont les programmeurs CSS qui ont tendance à s'appuyer excessivement sur les sélecteurs de classe et d'ID. S'ils souhaitent mettre en forme les titres d'une manière dans la zone de contenu principale et d'une autre dans la zone de contenu secondaire, ils créent souvent deux classes et appliquent l'une à un titre et l'autre à l'autre titre. Il est pourtant bien plus simple d'utiliser une combinaison de sélecteurs de type, de sélecteurs descendants, de sélecteurs d'ID ou de sélecteurs de classe :

```
#main-content h2 {font-size: 1.8em;}
#secondaryContent h2 {font-size: 1.2em;}

<div id="main-content">
  <h2>Articles</h2>
  ...
</div>
<div id="secondary-content">
  <h2>Latest news</h2>
  ...
</div>
```

Cet exemple est très simple et facilement compréhensible, mais vous seriez surpris du nombre d'éléments que vous pouvez cibler à l'aide des quatre seuls sélecteurs précédents. Si vous vous retrouvez à ajouter beaucoup de classes superflues à un document, c'est probablement le signe qu'il est mal structuré. Au lieu de cela, réfléchissez à la manière dont ces éléments diffèrent les uns des autres. Vous constaterez souvent que la seule véritable différence tient à l'endroit où ils apparaissent dans la page. Au lieu de leur attribuer différentes classes, pensez à appliquer une classe ou un ID à l'un de leurs parents, puis ciblez-les à l'aide d'un sélecteur descendant.

Pseudo-classes

Il peut arriver que vous souhaitiez appliquer un style à un élément en fonction d'un autre critère que celui de la structure du document (par exemple, l'état d'un lien ou d'un élément de formulaire). Cela est possible en utilisant un sélecteur de pseudo-classe.

```
/* affiche tous les liens non visités en bleu */
a:link {color:blue;}

/* affiche tous les liens visités en vert */
a:visited {color:green;}

/* affiche les liens en rouge quand on les survole ou les active.
l'état focus est ajouté pour la prise en charge du clavier */
a:hover, a:focus, a:active {color:red;}

/* affiche en rouge les lignes de tableau survolées */
tr:hover {background-color: red;}

/* affiche les éléments de saisie en jaune s'ils sont activés */
input:focus {background-color:yellow;}
```

`:link` et `:visited` sont ce qu'on appelle des pseudo-classes de liens. Elles ne peuvent être appliquées qu'aux éléments d'ancre. `:hover`, `:active` et `:focus` sont des pseudo-classes dynamiques, qui peuvent théoriquement s'appliquer à n'importe quel élément. La plupart des navigateurs modernes prennent en charge cette fonctionnalité. On ne s'étonnera pas, en revanche, qu'Internet Explorer 6 ne prête attention aux pseudo-classes `:active` et `:hover` que lorsqu'elles s'appliquent à un lien d'ancre et ignore complètement `:focus`. Internet Explorer 7 prend en charge `:hover` sur n'importe quel élément, mais ignore `:active` et `:focus`.

Enfin, il vaut la peine de noter que les pseudo-classes peuvent être adjointes les unes aux autres afin de créer des comportements plus complexes, par exemple pour un effet de survol différent sur les liens déjà consultés et ceux qui ne l'ont pas été.

```
/* affiche tous les liens visités survolés en vert olive */
a:visited:hover {color:olive;}
```

Le sélecteur universel

Le sélecteur universel est sans doute le plus efficace et le moins utilisé de tous les sélecteurs. Il agit à la manière d'un joker, puisqu'il désigne tous les éléments disponibles. Comme les jokers dans d'autres langages, il est identifié par un astérisque. Il est souvent utilisé pour appliquer des mises en forme à tous les éléments d'une page. Par exemple, pour supprimer le remplissage et les marges par défaut du navigateur sur tous les éléments, vous pouvez utiliser la règle suivante :

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

Combiné avec d'autres sélecteurs, le sélecteur universel peut servir à mettre en forme tous les descendants d'un élément particulier ou à ignorer un niveau de descendants. Vous verrez comment procéder en pratique à une autre section de ce chapitre.

Les sélecteurs avancés

Les CSS 2.1 et CSS 3 possèdent un certain nombre d'autres sélecteurs utiles. Malheureusement, si la plupart des navigateurs modernes les prennent en charge, ce n'est pas le cas des plus anciens, comme Internet Explorer 6. Les CSS ont cependant été conçues en tenant compte des problèmes de compatibilité descendante. Si un navigateur ne comprend pas un sélecteur, il ignore tout simplement la règle. Il est ainsi possible d'appliquer des embellissements stylistiques et fonctionnels dans les navigateurs récents sans craindre les problèmes qui pourraient en découler dans les navigateurs anciens. Évitez simplement d'utiliser ces sélecteurs avancés pour des fonctionnalités essentielles ou des aspects très importants de la mise en page du site.

Sélecteur d'enfants et sélecteur de frère adjacent

Le premier des sélecteurs avancés est le sélecteur d'enfants. Si un sélecteur descendant sélectionne tous les descendants d'un élément, le sélecteur d'enfants ne cible que les descendants immédiats de l'élément, que l'on appelle ses enfants. Dans l'exemple suivant, les éléments de la liste externe se voient attribuer une icône personnalisée alors que ceux de la liste imbriquée restent inchangés (voir Figure 2.1) :

```
#nav>li {
    background: url(folder.png) no-repeat left top;
    padding-left: 20px;
}

<ul id="nav">
    <li><a href="/">Accueil</a></li>
    <li><a href="/services/">Services</a>
        <ul>
            <li><a href="/services/design/">Conception</a></li>
            <li><a href="/services/development/">Développement</a></li>
            <li><a href="/services/consultancy/">Consulting</a></li>
        </ul>
    </li>
    <li><a href="/contact/">Nous contacter</a></li>
</ul>
```

Figure 2.1

Le sélecteur d'enfants met en forme les enfants de la liste, mais pas ses petits-enfants.



Le sélecteur d'enfants est pris en charge par Internet Explorer 7 et ses versions ultérieures. Un petit bogue de la version 7 provoque cependant des problèmes si des commentaires HTML sont insérés entre le parent et l'enfant.

Il est possible de simuler un sélecteur d'enfants qui fonctionne avec Internet Explorer 6 et les versions antérieures à l'aide du sélecteur universel. Pour cela, vous devez d'abord appliquer à tous les descendants le style que vous souhaitez attribuer aux enfants. Ensuite, utilisez le sélecteur universel pour redéfinir ces styles avec les descendants des enfants. Pour simuler le précédent exemple de sélecteur d'enfants, vous écririez ainsi :

```
#nav li {
    background: url(folder.png) no-repeat left top;
    padding-left: 20px;
}

#nav li * {
    background-image: none;
    padding-left: 0;
}
```

Parfois, il arrivera que vous souhaitiez mettre en forme un élément en raison de sa proximité avec un autre élément. Le sélecteur de frère adjacent permet de cibler un élément précédé par un autre élément qui possède le même parent. Grâce à lui, vous pouvez mettre en forme le premier paragraphe qui suit un titre de premier niveau, en le passant en gras, en gris ou en l'affichant légèrement plus gros que les paragraphes suivants (voir Figure 2.2) :

```
h2 + p {  
    font-size: 1.4em;  
    font-weight: bold;  
    color: #777;  
}  
<h2>Peru Celebrates Guinea Pig festival</h2>  
<p>The guinea pig festival in Peru is a one day event to celebrate  
these cute local animals. The festival included a fashion show where  
animals are dressed up in various amusing costumes.</p>  
<p>Guinea pigs can be fried, roasted, or served in a casserole. Around  
65 million guinea pigs are eaten in Peru each year.</p>
```

Figure 2.2

Le sélecteur de frère adjacent peut être utilisé pour mettre en forme le premier paragraphe après un titre et éviter d'utiliser des classes superflues.

Peru Celebrates Guinea Pig festival

The guinea pig festival in Peru is a one day event to celebrate these cute local animals. The festival included a fashion show where animals are dressed up in various amusing costumes.

Guinea pigs can be fried, roasted or served in a casserole. Around 65 million guinea pigs are eaten in Peru each year.

Comme avec le sélecteur d'enfants, le processus échoue dans Internet Explorer 7 si des commentaires sont logés entre les éléments ciblés.

Sélecteur d'attribut

Le sélecteur d'attribut permet de cibler un élément en fonction de l'existence ou de la valeur d'un attribut. Ce mode de sélection est particulièrement intéressant et efficace.

Par exemple, lorsque vous survolez un élément doté d'un attribut `title`, la plupart des navigateurs affichent une info-bulle. Vous pouvez, par exemple, exploiter ce mécanisme pour faire apparaître la définition d'acronymes ou d'abréviations :

```
<p>The term <acronym title="self-contained underwater breathing  
apparatus">SCUBA</acronym> is an acronym rather than an abbreviation as it is  
pronounced as a word.</p>
```

Il n'existe cependant aucun moyen de savoir que ces informations supplémentaires existent si l'on ne survole pas l'élément. Pour résoudre ce problème, vous pouvez utiliser le sélecteur d'attribut afin d'appliquer une mise en forme différente aux éléments d'acronyme qui possèdent des titres et à ceux qui n'en ont pas – ici, en les soulignant avec une bordure en pointillé. Vous pouvez fournir d'autres informations contextuelles en transformant le curseur en point d'interrogation pendant le survol de l'élément, afin d'indiquer que celui-ci a quelque chose de particulier par rapport aux autres.

```
acronym[title] {  
    border-bottom: 1px dotted #999;  
}  
  
acronym[title]:hover, acronym[title]:focus {  
    cursor: help;  
}
```

En plus de mettre en forme un élément en fonction de l'existence d'un attribut, vous pouvez appliquer des styles en fonction d'une valeur particulière. Par exemple, les sites qui sont liés à l'utilisation d'un attribut `rel` avec la valeur `nofollow` n'offrent aucun avantage en termes de classement sur Google. La règle suivante affiche une image à côté de ces liens, par exemple pour signaler que le site cible est déconseillé :

```
a[rel="nofollow"] {  
    background: url(nofollow.gif) no-repeat right center;  
    padding-right: 20px;  
}
```

Tous les navigateurs web modernes, Internet Explorer 7 y compris, prennent en charge ces sélecteurs. Comme Internet Explorer 6 ne les reconnaît pas, vous pouvez les utiliser justement pour appliquer un style à Internet Explorer 6 et un autre aux navigateurs mieux lotis en termes de fonctionnalités. Par exemple, Andy Clarke utilise cette technique en proposant une version en noir et blanc de son site aux utilisateurs d'Internet Explorer 6 (voir Figure 2.3) et une version en couleur pour tous les autres navigateurs (voir Figure 2.4).

```
#header {  
    background: url(branding-bw.png) repeat-y left top;  
}  
  
[id="header"] {  
    background: url(branding-color.png) repeat-y left top;  
}
```

Certains attributs peuvent posséder plusieurs valeurs séparées par des espaces. Le sélecteur d'attribut permet de cibler un élément en fonction de l'une de ces valeurs. Par exemple, le microformat XFN permet de définir la relation avec un site en ajoutant des mots-clés à l'attribut `rel` du lien d'ancre. On peut ainsi faire savoir qu'un site appartient à l'un de ses collègues de travail en ajoutant le mot-clé `co-worker` aux liens de sa liste de blogs.

Figure 2.3

Andy Clarke propose une version en noir et blanc de son site aux utilisateurs d'Internet Explorer 6 en utilisant, notamment, des sélecteurs d'attribut.

**Figure 2.4**

Les navigateurs plus récents disposent d'une version en couleur.



Cette information peut ensuite être signalée grâce à une icône particulière affichée à côté du nom du collègue.

```
.blogroll a[rel="co-worker"] {  
    background: url(co-worker.gif) no-repeat left center;  
}  
  
<ul class="blogroll">  
    <li>  
        <a href="http://adactio.com/" rel="friend met colleague co-worker">Jeremy  
Keith</a>  
    </li>  
    <li>  
        <a href="http://clagnut.com/" rel="friend met colleague co-worker">Richard  
Rutter</a>  
    </li>  
    <li>  
        <a href="http://hicksdesign.com/" rel="friend met colleague">John Hicks</a>  
    </li>  
    <li>  
        <a href="http://stuffandnonsense.co.uk/" rel="friend met colleague">Andy  
Clarke</a>  
    </li>  
</ul>
```

Cascade et spécificité

Avec une feuille de styles même moyennement compliquée, il peut arriver que deux règles ciblent le même élément. Les CSS gèrent ces conflits par le biais d'un processus appelé *cascade*. La cascade attribue une importance à chaque règle. Les feuilles de styles auteur sont celles écrites par les développeurs de site et sont considérées comme les plus importantes. Les utilisateurs peuvent appliquer leurs propres styles *via* le navigateur et ces feuilles viennent en second rang. Pour finir, les feuilles de styles par défaut utilisées par le navigateur ou l'agent utilisateur sont considérées comme les moins importantes, de sorte qu'il est toujours possible de les redéfinir. Pour mieux contrôler l'affichage, les utilisateurs peuvent redéfinir des règles en les marquant avec `!important`. Ces règles redéfinissent même celles marquées comme `!important` par l'auteur. Cette liberté est offerte pour des besoins spécifiques en termes d'accessibilité, comme des feuilles de styles à contraste élevé pour les problèmes de dyslexie.

La cascade respecte donc l'ordre de priorité suivant :

- les styles utilisateur marqués comme `!important` ;
- les styles auteur marqués comme `!important` ;
- les styles auteur ;
- les styles utilisateur ;
- les styles appliqués par le navigateur/l'agent utilisateur.

Les règles sont ensuite ordonnées en fonction de la spécificité du sélecteur. Les règles aux sélecteurs plus spécifiques redéfinissent celles dont les sélecteurs sont moins spécifiques.

Si deux règles possèdent la même spécificité, c'est celle qui est définie en dernier qui prend le dessus.

Spécificité

Par calculer le niveau de spécificité d'une règle, chaque sélecteur reçoit une valeur numérique. La spécificité d'une règle est ensuite évaluée par addition des valeur de tous ses sélecteurs. Malheureusement, la spécificité n'est pas calculée en base 10 mais avec un numéro de base élevé et non spécifié. Ce procédé est destiné à garantir qu'un sélecteur très spécifique, comme un sélecteur d'ID, ne soit jamais redéfini par un grand nombre de sélecteurs moins spécifiques, comme des sélecteurs de type. Si vos sélecteurs comptent moins de 10 sélecteurs, vous pouvez cependant vous contenter, pour simplifier, de calculer la spécificité en base 10.

La spécificité d'un sélecteur se décompose en quatre niveaux constituants : a, b, c et d.

- Si le style est un style incorporé, a vaut 1.
- b vaut le nombre total de sélecteurs d'ID.
- c vaut le nombre de sélecteurs de classe, de pseudo-classe et d'attribut.
- d vaut le nombre de sélecteurs de type et de sélecteurs de pseudo-élément.

Avec ces règles, il est possible de calculer la spécificité de n'importe quel sélecteur CSS. Le Tableau 2.1 présente une série de sélecteurs avec leur spécificité correspondante.

Tableau 2.1 : Exemples de spécificité

Sélecteur	Spécificité	Spécificité en base 10
Style=""	1,0,0,0	1000
#wrapper #content {}	0,2,0,0	200
#content .datePosted {}	0,1,1,0	110
div#content {}	0,1,0,1	101
#content {}	0,1,0,0	100
p.comment .dateposted {}	0,0,2,1	21
p.comment{}	0,0,1,1	11
div p {}	0,0,0,2	2
p {}	0,0,0,1	1

Au premier abord, toutes ces explications concernant la spécificité et les nombres en base élevée mais non spécifiée peuvent sembler déroutantes. Voici donc ce qu'il faut en retenir. Fondamentalement, une règle écrite dans un attribut de style sera toujours plus spécifique que n'importe quelle autre. Une règle avec un ID sera plus spécifique qu'une règle sans et une règle avec un sélecteur de classe le sera plus qu'une règle avec de simples sélecteurs de type. Enfin, si deux règles possèdent la même spécificité, celle définie en dernier prend le dessus.

La spécificité peut être extrêmement importante pour la correction des bogues, car vous devez savoir quelle règle a priorité et pourquoi. Par exemple, supposons que vous ayez créé le jeu de règles suivant. D'après vous, de quelle couleur seront les titres ?

```
#content div#main-content h2 {  
    color: gray;  
}  
#content #main-content>h2 {  
    color: blue;  
}  
body #content div[id="main-content"] h2 {  
    color: green;  
}  
#main-content div.news-story h2 {  
    color: orange;  
}  
  
#main-content [class="news-story"] h2 {  
    color: yellow;  
}  
div#main-content div.news-story h2.first {  
    color: red;  
}  
  
<div id="content">  
    <div id="main-content">  
        <h2>Strange Times</h2>  
        <p>Here you can read bizarre news stories from around the globe.</p>  
        <div class="news-story">  
            <h2 class="first">Bog Snorkeling Champion Announced Today</h2>  
            <p>The 2008 Bog Snorkeling Championship was won by Conor Murphy  
            ➔ with an impressive time of 1 minute 38 seconds.</p>  
        </div>  
    </div>  
</div>
```

Bizarrement, la réponse est que les deux titres seront gris. Le premier sélecteur possède la plus haute spécificité parce qu'il est composé de deux sélecteurs d'ID. Certains des derniers sélecteurs peuvent avoir l'air plus complexes, mais comme ils ne contiennent qu'un seul ID, ils perdent toujours face aux sélecteurs plus spécifiques.

Si vous rencontrez des règles CSS qui ne semblent pas fonctionner, c'est sans doute qu'il existe un conflit de spécificité. Essayez de rendre vos sélecteurs plus spécifiques en ajoutant l'ID de l'un de leurs parents. Si cet ajout règle le problème, vous constaterez sans doute qu'il existe une règle plus spécifique quelque part dans votre feuille de styles qui redéfinit celle que vous essayiez d'appliquer. Dans ce cas, n'hésitez pas à parcourir votre code à nouveau afin de résoudre les conflits de spécificité et que le code soit aussi léger que possible.

Utiliser la spécificité dans les feuilles de styles

La spécificité est très utile pour concevoir des CSS, car elle permet de définir des styles généraux pour les éléments communs puis de redéfinir ces styles pour les éléments plus spécifiques. Par exemple, supposons que vous souhaitiez que la plus grande partie du texte

de votre site s'affiche en noir, à l'exception du texte d'introduction, que vous voulez gris. Vous pouvez alors procéder de la manière suivante :

```
p {color: black;}  
p.intro {color: grey;}
```

Cette approche convient pour les petits sites mais, avec ceux de grande taille, vous rencontrerez un nombre toujours plus important d'exceptions. Le texte d'introduction de vos articles de news devra s'afficher en bleu, le texte d'introduction de votre page d'accueil s'affichera sur un arrière-plan gris, etc. Chaque fois que vous créerez un style plus spécifique, vous aurez besoin de redéfinir certaines des règles générales. Rapidement, la quantité de code pourra devenir considérable. Tout peut aussi sérieusement se compliquer parce qu'un élément peut récupérer des styles à différents emplacements.

Pour éviter la confusion, essayez de vous assurer que vos styles généraux sont très généraux et que les styles spécifiques le sont autant que possible et n'ont jamais besoin d'être eux-mêmes redéfinis. Si vous constatez que vous devez redéfinir des styles généraux plusieurs fois, mieux vaut supprimer la déclaration à redéfinir dans les règles générales et l'appliquer explicitement à chaque élément qui la requiert.

Ajouter une classe ou un ID à la balise *body*

L'un des moyens intéressants d'exploiter la spécificité consiste à appliquer une classe ou un ID à la balise *body*. Vous pouvez ensuite redéfinir des styles page par page ou même pour tout le site. Par exemple, si vous souhaitez que toutes vos nouvelles pages possèdent une disposition particulière, vous pouvez ajouter un nom de classe à l'élément *body* et l'utiliser pour cibler vos styles :

```
body.news {  
    /* faire quelque chose */  
}  
  
<body class="news">  
    <p>My, what a lovely body you have.</p>  
</body>
```

Vous aurez parfois besoin de redéfinir ces styles dans une page particulière, comme votre page d'archive des news. Dans ce cas, vous pouvez ajouter un ID à la balise *body* afin de cibler cette page.

```
body.news {  
    /* faire quelque chose */  
}  
  
body#archive {  
    /* faire autre chose */  
}  
  
<body id="archive" class="news">  
    <p>My, what a lovely body you have.</p>  
</body>
```

En utilisant une classe pour le type de page et un ID pour la page spécifique, vous profitez d'un contrôle précis sur l'apparence et la mise en page de votre site. C'est l'une de mes techniques favorites pour concevoir des sites faciles à gérer.

Héritage

Les gens confondent souvent l'héritage et la cascade. Si, au premier abord, ces deux notions s'apparentent, elles sont en réalité assez différentes. L'héritage est plus facile à comprendre. Certaines propriétés, comme la couleur ou la taille des polices, sont héritées par les descendants des éléments auxquels elles sont appliquées. Par exemple, si vous attribuez à l'élément `body` une couleur de texte noire, le texte de tous les descendants de cet élément apparaîtra également en noir. Il en va de même pour les tailles de police. Si vous donnez à l'élément `body` une taille de police de 1.4 em (cadratin), tous les éléments de la page doivent hériter cette taille de police. Je dis "doivent", car Internet Explorer sous Windows et Netscape ont quelques problèmes avec l'héritage des tailles de police dans les tableaux. Pour contourner ce problème, vous devez soit indiquer explicitement que les tableaux doivent hériter des tailles de police, soit définir séparément leurs tailles de police.

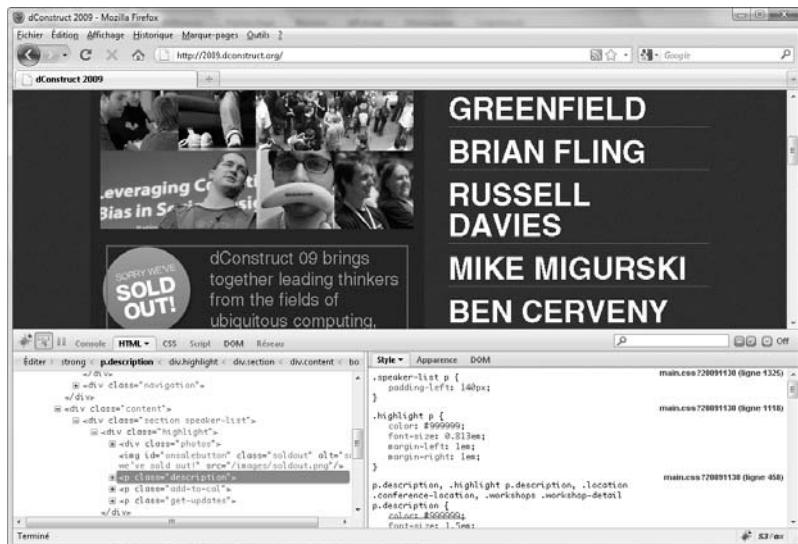


Figure 2.5

Firebug est un plug-in pratique de Firefox qui permet d'analyser différents éléments et de voir d'où proviennent les styles qui leur sont appliqués.

Si vous définissez la taille de police de l'élément `body`, vous remarquerez que ce style n'est pas répercuté sur les titres de la page. Vous pourriez en déduire que les titres n'héritent pas la propriété de taille du texte mais, en réalité, c'est la feuille de styles par défaut du navigateur qui définit la taille des titres. Tous les styles qui sont appliqués directement à un élément redéfinissent un style hérité. Les styles hérités possèdent en effet une spécificité nulle.

L'héritage est très utile car il évite d'ajouter le même style à chaque descendant d'un élément. Si la propriété que vous essayez de définir est une propriété héritée, vous pouvez également l'appliquer à l'élément parent. Après tout, quelle utilité y a-t-il à écrire ceci :

```
p, div, h1, h2, h3, ul, ol, dl, li {color: black;}
```

alors qu'on peut écrire ceci ?

```
body {color: black;}
```

De la même manière que la cascade peut aider à simplifier les CSS, l'héritage bien utilisé peut aider à réduire le nombre et la complexité des sélecteurs dans le code. Si de nombreux éléments héritent différents styles, il peut cependant être difficile de déterminer d'où proviennent les styles.

Planification, organisation et gestion des feuilles de styles

Plus les sites grossissent, se compliquent et se sophistiquent sur le plan graphique, plus les CSS deviennent difficiles à gérer. À cette section, nous allons voir comment gérer le code, notamment en regroupant les styles dans des sections logiques et en ajoutant des commentaires pour faciliter la lecture du code.

Application de styles aux documents

Vous pouvez ajouter des styles directement dans l'en-tête d'un document en les plaçant entre des balises `style`, mais cette méthode n'est pas conseillée. Si vous créez une autre page qui utilise les mêmes styles, vous êtes alors contraint de les dupliquer dans celle-ci. Si vous souhaitez ensuite changer un style, il faut le changer à deux endroits au lieu d'un. Les CSS permettent heureusement de conserver tous les styles dans une ou plusieurs feuilles de styles externes. Il existe deux moyens d'attacher des feuilles de styles externes à une page web : les lier ou les importer.

```
<link href="/css/basic.css" rel="stylesheet" type="text/css" />
<style type="text/css">
<!--
@import url("/css/advanced.css");
-->
</style>
```

L'importation ne se restreint pas aux limites du document HTML, puisque vous pouvez également importer une feuille de styles à partir d'une autre feuille de styles. Il est ainsi possible de lier une feuille de styles de base dans la page HTML puis d'y importer les styles plus compliqués.

```
@import url(/css/layout.css);
@import url(/css/typography.css);
@import url(/css/color.css);
```

Cette manière de décomposer les CSS en plusieurs feuilles de styles était une procédure courante, que je recommandais, il y a peu encore. Les récents bancs d'essai des navigateurs semblent cependant suggérer que l'importation des feuilles de styles est moins rapide que leur liaison.

Il existe deux autres problèmes liés à la vitesse dans le cas de l'utilisation de plusieurs fichiers CSS. Pour commencer, le fait d'utiliser plusieurs fichiers conduit à envoyer plus de paquets depuis le serveur, et c'est leur nombre (plutôt que leur contenu) qui affecte le temps de téléchargement. En outre, les navigateurs ne peuvent télécharger simultanément qu'un nombre limité de fichiers du même domaine. Pour les anciens navigateurs, cette limite se réduisait à deux pauvres fichiers, et les navigateurs modernes ont monté la barre à huit fichiers. Dans un ancien navigateur, avec trois feuilles de styles, il fallait attendre que les deux premiers fichiers se téléchargent pour que la transmission du troisième commence. Voilà pourquoi un unique fichier CSS bien structuré peut aider à améliorer considérablement les vitesses de téléchargement.

Le recours à un unique fichier CSS permet aussi de conserver tout le code au même endroit. Il m'est arrivé de recommander de décomposer le code pour en faciliter la maintenance. Malheureusement, c'était souvent difficile de déterminer si une déclaration était liée à la mise en page ou à la typographie du site. Parfois, elle pouvait ressortir aux deux et il fallait trancher de manière arbitraire. Cela nécessitait, en outre, de conserver plusieurs feuilles de styles ouvertes et de basculer constamment des unes aux autres. Grâce aux fonctionnalités de réduction des lignes de code proposées par la plupart des éditeurs CSS, il est maintenant plus facile d'éditer un unique fichier. Il semble donc préférable d'utiliser un unique fichier CSS plutôt que plusieurs petits. Tout dépend cependant du site concerné ; aucune règle ne vaut uniformément pour tous les sites en la matière.

Lorsque vous écrirez vos propres feuilles de styles, vous aurez une idée plutôt claire de la manière dont elles sont structurées, des problèmes que vous avez rencontrés et des raisons qui vous ont poussé à faire les choses d'une manière et pas d'une autre. Si vous rouvrez vos feuilles de styles six mois plus tard, il est fort probable que vous aurez oublié bien des choses à ce sujet. Il arrivera peut-être aussi que vous deviez confier vos CSS à un autre programmeur chargé de l'éditer. Il est donc préférable de commenter votre code.

Les commentaires CSS s'ajoutent très facilement. Ils commencent par `/*` et se terminent par `*/`. Ce sont des commentaires dits "de style C", car il s'agit du type de commentaire utilisé dans le langage de programmation C. Ils peuvent être d'une ligne ou multilignes et peuvent apparaître n'importe où dans le code.

```
/* Styles du corps de page */
body {
    font-size: 67.5%; /* Définit la taille de police */
}
```

Quand les fichiers CSS s'allongent, il peut devenir difficile de retrouver certains styles. L'un des moyens d'accélérer cette tâche consiste à ajouter un drapeau à chacun des en-têtes de commentaire. Il s'agit d'un mot supplémentaire qui précède le texte d'en-tête et n'apparaît pas, naturellement, dans les fichiers CSS. Une recherche portant sur ce drapeau suivi des deux premières lettres de l'en-tête de commentaire conduira ainsi directement à la partie du

fichier recherché. Dans l'exemple suivant, la recherche "@group typ" amène à la section de typographie de la feuille de styles :

```
/* @group typographie */
```

Si vous utilisez l'éditeur OS X CSS Edit, cette convention est utilisée comme moyen simple mais efficace de naviguer dans les feuilles de styles.

Structuration du code

Il est judicieux de décomposer les feuilles de styles en blocs cohérents afin d'en faciliter la maintenance. Chez Clearleft, nous commençons généralement par les styles les plus généraux, comme ceux qui sont appliqués à la balise body et qui doivent être hérités par tous les éléments du site. Ensuite, viennent les redéfinitions globales, puis les liens, les titres et les autres éléments.

Une fois les styles généraux établis, nous devenons un peu plus précis et traitons les styles utilitaires. Il s'agit de classes générales utilisées dans tout le site, par exemple pour les formulaires et les messages d'erreur. Nous passons ensuite aux éléments structurels, comme la mise en page et la navigation.

À mesure que nous avançons dans la feuille de styles, nous étageons progressivement les styles les uns au-dessus des autres, en gagnant chaque fois en spécificité. Quand les éléments d'architecture de la page sont traités, nous passons aux composants propres à chaque page. Pour finir, nous incluons toutes les redéfinitions et les exceptions en bas du document, lequel présente finalement la structure suivante :

- Styles généraux
 - Styles body
 - Redéfinitions
 - Liens
 - Titres
 - Autres éléments
- Styles utilitaires
 - Formulaires
 - Notifications et erreurs
 - Éléments répétitifs
- Structure de page
 - Titres, pieds de page et navigation
 - Mise en page
 - Autres blocs de page

- Composants de page
 - Pages individuelles
- Redéfinitions

J'utilise un grand bloc de commentaire mis en forme pour séparer visuellement chaque section.

```
/* @group styles généraux
-----*/
```



```
/* @group styles utilitaires
-----*/
```



```
/* @group structure de page
-----*/
```



```
/* @group composants de page
-----*/
```



```
/* @group redéfinitions
-----*/
```

Bien sûr, tout ne vient pas se ranger dans un bloc bien défini. Parfois, il faut se poser quelques questions. Gardez à l'esprit que, plus vous pourrez décomposer et rationaliser votre code, plus il sera facile à comprendre et plus vous pourrez rapidement retrouver les règles que vous cherchez.

Comme mes feuilles de styles possèdent souvent une structure analogue, je gagne du temps en créant mes propres modèles CSS précommentés pour tous mes projets. Vous pouvez encore gagner du temps en ajoutant quelques règles courantes à utiliser dans tous vos sites, de manière à créer une sorte de fichier CSS prototype. Inutile de réinventer la roue chaque fois que vous lancez un nouveau projet. Vous trouverez un fichier CSS prototype dans le code à télécharger pour ce livre sur le site www.pearson.fr.

Note à soi-même

Dans le cadre des projets complexes et volumineux, il est souvent utile d'annoter les fichiers CSS avec des commentaires temporaires qui facilitent le développement. Il peut s'agir de simples rappels des tâches à effectuer avant la phase de lancement ou de tableaux de référence pour les valeurs courantes, comme les largeurs de colonne.

Si vous utilisez beaucoup de couleurs, vous risquez d'effectuer de constants allers-retours entre votre application graphique et votre éditeur texte pour vérifier les valeurs hexadécimales. C'est vite agaçant et c'est précisément pourquoi certains développeurs ont suggéré de créer des variables CSS. L'idée est intéressante, mais elle conduirait à rapprocher encore les CSS d'un langage de programmation, avec le risque de décourager les non-programmeurs. J'ai préféré adopter une méthode plus simple. Je me contente d'ajouter une petite table de référence des couleurs en haut de ma feuille de styles afin de pouvoir m'y référer constamment pendant le développement. Une fois que j'ai fini de développer la page, je la supprime généralement.

```
/* Variables de couleur

@colordef #434343; gris sombre
@colordef #f2f6e4; vert clair
@colordef #90b11f; vert foncé
@colordef #369; bleu foncé
*/
```

Pour rendre vos commentaires plus signifiants, vous pouvez utiliser des mots-clés qui distinguent les commentaires importants. Pour ma part, j'utilise `@todo` pour indiquer que quelque chose doit être modifié, corrigé ou revu plus tard, `@bugfix` pour signaler un problème avec le code ou un navigateur particulier et `@workaround` pour évoquer une solution de contournement :

```
/* :@todo Penser à supprimer cette règle avant de publier le site */
/* @workaround: J'ai réussi à résoudre le problème dans IE en définissant
une petite marge négative, mais ce n'est pas esthétique */
/* @bugfix: Cette règle ne fonctionne pas sous IE 5.2 Mac */
```

En programmation, ces mots-clés sont appelés des *gotchas* ; ils peuvent être très utiles au cours des phases ultérieures de développement. En fait, tous ces termes font partie d'un projet appelé CSSDoc (<http://cssdoc.net>) qui vise à développer une syntaxe standardisée pour les commentaires des feuilles de styles.

Supprimer les commentaires et optimiser les feuilles de styles

Les commentaires peuvent considérablement augmenter la taille des fichiers CSS. Il peut donc être utile de les en faire disparaître. La plupart des éditeurs HTML/CSS ou des éditeurs de texte possèdent une fonctionnalité de recherche et remplacement qui facilite cette opération. Vous pouvez sinon utiliser l'un des multiples optimiseurs CSS en ligne comme celui que propose www.cssoptimiser.com. L'optimiseur supprime non seulement les commentaires, mais également les espaces blancs, afin de vous aider à retirer encore quelques octets supplémentaires de votre code. Si vous choisissez de supprimer les commentaires de vos feuilles de styles, n'oubliez pas d'en conserver une version avec commentaires pour votre environnement de production. Le meilleur moyen de gérer ce processus consiste à créer un script de déploiement qui supprime automatiquement les commentaires lorsque vous publiez vos modifications sur le Web. Comme il s'agit d'une technique avancée, mieux vaut toutefois la réservier aux sites de grande envergure ou sophistiqués.

Autrement, le meilleur moyen de réduire la taille des fichiers est d'activer la compression côté serveur. Si vous utilisez un serveur Apache, contactez votre hébergeur concernant l'installation de `mod_gzip` ou `mod_deflate`. Tous les navigateurs web modernes peuvent gérer ces fichiers compressés avec GZIP et les décompresser à la volée. Ces modules Apache détectent si le navigateur peut gérer les fichiers de ce type et, si c'est le cas, ils transmettent une version compressée. La compression côté serveur peut réduire la taille des fichiers HTML et CSS de 80 % environ, ce qui libère la bande passante et accélère le téléchargement des pages. Si vous n'avez pas accès à ces modules Apache, vous pouvez compresser vos fichiers en suivant le didacticiel (en anglais) à l'adresse <http://tinyurl.com/8w9rp>.

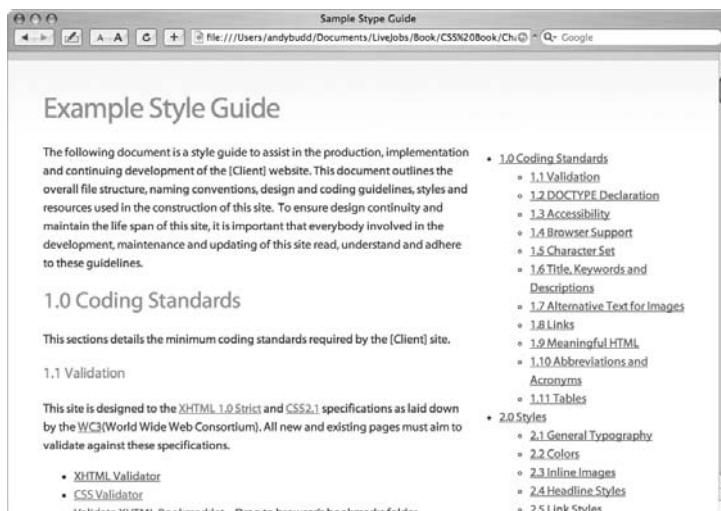
Guides de styles

La gestion de la plupart des sites web occupe généralement plusieurs personnes et parfois même, pour les sites d'envergure, plusieurs équipes chargées de différents aspects du site. Les programmeurs, les gestionnaires de contenu et les autres développeurs d'interface peuvent avoir besoin de comprendre le fonctionnement des éléments de votre code et de votre mise en page. Il est donc judicieux de créer un guide de styles.

Le guide de styles est un document, une page web ou un microsite, qui explique comment se coordonnent le code et la conception graphique d'un site. Au départ, le guide de styles doit définir les directives de conception générales comme le traitement approprié pour les titres et les autres éléments typographiques, indiquer comment fonctionne la structure de grille et préciser la palette de couleurs à utiliser. Les guides de styles de qualité détaillent aussi le traitement de la répétition des éléments comme les articles, les news et les notifications, afin d'indiquer comment les implémenter. Les guides de styles plus détaillés peuvent même inclure des informations concernant les standards en matière de programmation comme la version XHTML/CSS utilisée, le niveau d'accessibilité désiré, les détails concernant la prise en charge des navigateurs et des bonnes pratiques de programmation générales (voir Figure 2.6).

Figure 2.6

Exemple de guide de styles.

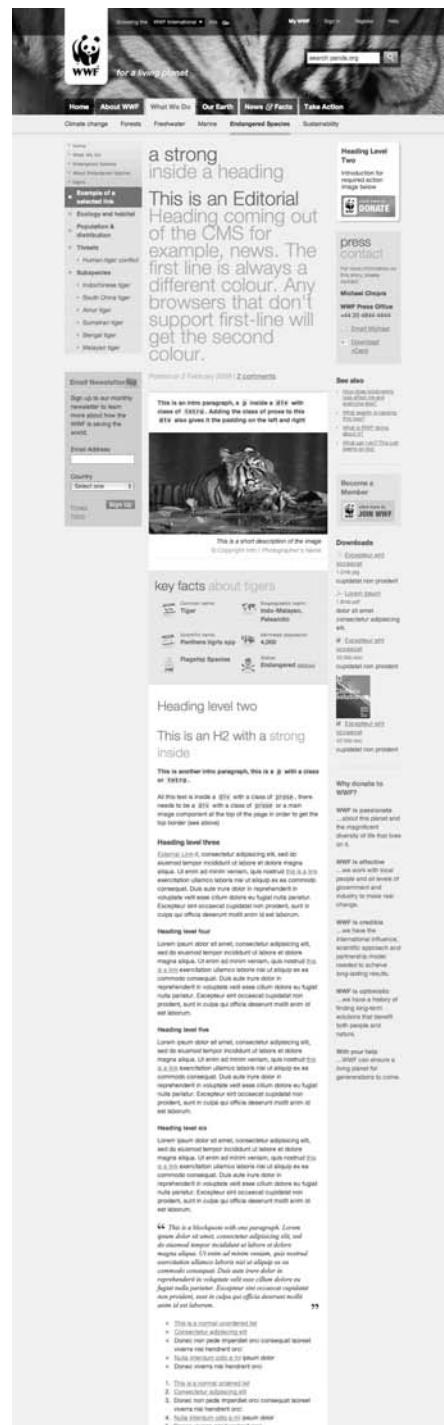


Les guides de styles sont un excellent moyen de transmettre un projet à ceux qui doivent en assurer la maintenance ou se charger de son implémentation. En consignant quelques directives simples, vous les aiderez à contrôler le développement du site et à minimiser la fragmentation des styles au fil du temps. Les guides de styles sont en outre une excellente introduction pour les nouveaux employés ou les contractuels à un système qui ne leur est pas familier et peut d'abord paraître compliqué.

Malheureusement, le guide de styles risque de se désynchroniser rapidement avec le site en production et sa mise à jour requiert quelques efforts. C'est la raison pour laquelle je préfère utiliser une sorte de guide de styles actif que j'ai appelé un "portfolio de motifs" (voir Figure 2.7).

Figure 2.7

Extrait du portfolio de motifs du site WWF International. La page effective fait environ cinq fois cette longueur et contient toutes les permutations typographiques ou structurelles autorisées sur le site.



Un portfolio de motifs est une page, ou une série de pages, qui utilise les feuilles de styles réelles du site pour afficher chaque permutation et combinaison de styles d'un site, des niveaux de titre et des styles de texte jusqu'au contenu spécifique et aux types de mises en page. Ces pages peuvent fournir aux développeurs back-end et front-end des ressources très précieuses, en leur donnant la possibilité de construire des combinaisons de pages qui n'ont pas encore été conçues. En exploitant ainsi les styles réellement utilisés, ils peuvent aussi opérer des tests régressifs, afin de vérifier que tous les changements CSS n'ont pas provoqué de problème gênant.

En résumé

Vous vous êtes rafraîchi la mémoire avec les sélecteurs CSS 2.1 courants et vous avez pu découvrir certains des puissants nouveaux sélecteurs CSS 3. Vous comprenez maintenant mieux le principe de spécificité et vous savez comment utiliser la cascade pour structurer vos règles CSS et mieux cibler vos éléments. Vous avez également appris à commenter et structurer vos feuilles de styles pour en faciliter la maintenance.

Au chapitre suivant, vous en apprendrez plus sur le modèle de boîte CSS, vous verrez comment et pourquoi les marges s'effondrent et découvrirez comment fonctionnent le flottement et le positionnement.

3

Vue d'ensemble du modèle de formatage visuel

Les trois concepts CSS les plus importants à comprendre sont le flottement, le positionnement et le modèle de boîte. Ils déterminent la disposition des éléments de la page et forment ainsi la base des mises en page CSS. Si vous êtes habitué à créer des maquettes tabulaires, ces notions pourront paraître étranges au premier abord. En fait, la plupart des développeurs doivent passer un certain temps à créer des sites en CSS avant de bien comprendre les tenants et les aboutissants du modèle de boîte, la différence entre le positionnement absolu et le positionnement relatif, et le fonctionnement du flottement et de la propriété `clear`. Le développement des sites CSS sera plus facile lorsque vous aurez vraiment assimilé ces concepts.

Au cours de ce chapitre, vous découvrirez :

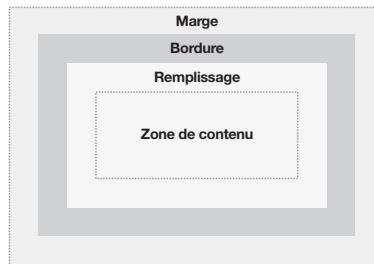
- les complexités et les particularités du modèle de boîte ;
- comment et pourquoi les marges s'effondrent ;
- la différence entre positionnement absolu et positionnement relatif ;
- le fonctionnement des propriétés `float` et `clear`.

Récapitulatif sur le modèle de boîte

Le modèle de boîte est l'une des pierres d'angle des CSS. Il définit la manière dont les éléments s'affichent et, dans une certaine mesure, comment ils interagissent les uns avec les autres. Chaque élément dans la page est considéré comme une boîte rectangulaire composée du contenu de l'élément, d'un remplissage, d'une bordure et de marges (voir Figure 3.1).

Figure 3.1

Illustration du modèle de boîte.



Le remplissage s'applique autour de la zone de contenu. Quand on ajoute un arrière-plan à un élément, il est appliqué à la zone formée par le contenu et le remplissage. C'est la raison pour laquelle le remplissage est souvent utilisé pour créer une gouttière autour du contenu afin de donner l'impression qu'il ne déborde pas sur les côtés de l'arrière-plan. La bordure affiche une ligne à l'extérieur de la zone remplie. Différents styles de ligne sont proposés : continue, en pointillé ou à tirets. À l'extérieur de la bordure figure une marge. Les marges sont transparentes et ne se voient pas. Elles sont généralement utilisées pour contrôler l'espace entre les éléments.

La CSS 2.1 contient aussi la propriété `outline` (contour). À la différence des bordures, les contours sont dessinés par-dessus la boîte de l'élément et n'affectent ni sa taille, ni son positionnement. Ils peuvent donc être utiles pour corriger des bogues, car ils n'altèrent pas la mise en page. Les contours sont pris en charge par la plupart des navigateurs modernes, dont Internet Explorer 8, mais ils ne sont pas reconnus par Internet Explorer 7 et ses versions antérieures.

Le remplissage, les bordures et les marges sont optionnels et valent zéro par défaut. De nombreux éléments se voient cependant attribuer des marges et un remplissage par défaut par la feuille de styles de l'agent utilisateur. Vous pouvez redéfinir ces styles de navigateur en ramenant les propriétés `margin` ou `padding` de l'élément à zéro. Vous pouvez le faire au cas par cas ou pour tous les éléments, à l'aide du sélecteur universel :

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

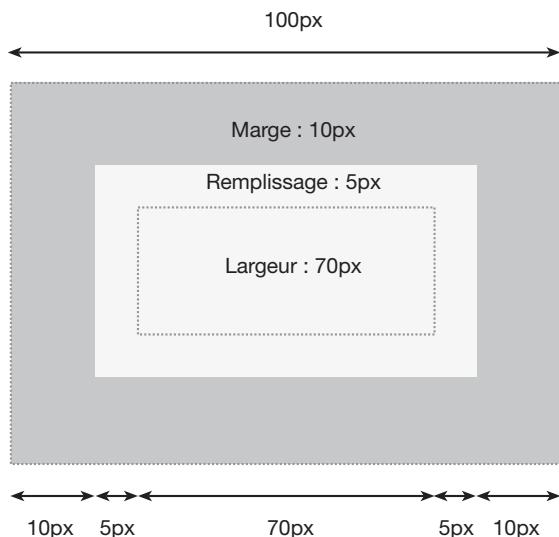
Rappelez-vous simplement que cette technique agit sans discernement et peut donc avoir des effets indésirables sur certains éléments, comme `option`. Il est préférable de ramener ces propriétés à zéro de manière explicite, en utilisant une réinitialisation globale.

En CSS, `width` et `height` font référence à la largeur et à la hauteur de la zone de contenu. L'ajout d'un remplissage, de bordures et de marges ne modifie pas la taille de la zone de contenu mais augmente la taille totale de la boîte de l'élément. Si vous souhaitez une boîte avec une marge de 10 pixels et un remplissage de 5 pixels de chaque côté pour atteindre 100 pixels de large, vous devez fixer la largeur du contenu à 70 pixels (voir Figure 3.2) :

```
#myBox {
  margin: 10px;
  padding: 5px;
  width: 70px;
}
```

Figure 3.2

Le modèle de boîte correct.



Le remplissage, les bordures et les marges peuvent être appliqués sur tous les côtés d'un élément ou sélectivement sur certains d'entre eux. Les marges peuvent également avoir des valeurs négatives et être utilisées pour une variété de techniques.

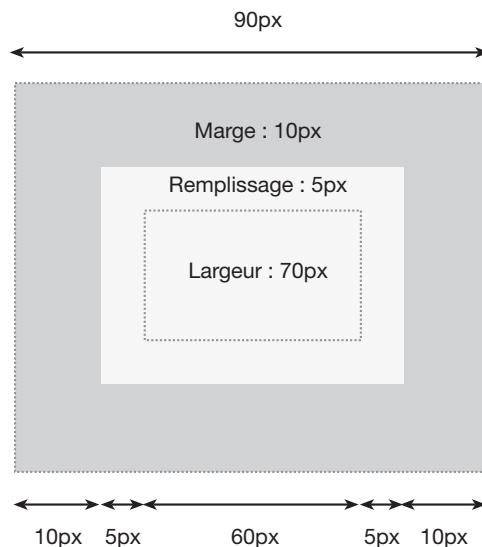
Internet Explorer et le modèle de boîte

Malheureusement, les anciennes versions d'Internet Explorer ainsi qu'Internet Explorer 6 en mode Quirks utilisent leur propre modèle de boîte non standard. Au lieu de mesurer simplement la largeur du contenu, ces navigateurs considèrent la propriété `width` comme la somme de la largeur du contenu, du remplissage et des bordures. C'est cohérent, puisque, en pratique, les boîtes possèdent une taille fixe et que le remplissage s'insère à l'intérieur. Plus vous ajoutez de remplissage, moins il y a de place pour le contenu. Il n'en reste pas moins que la spécification ne reprend pas ce raisonnement, et que le fait que ces versions d'Internet Explorer n'en tiennent pas compte pose de vrais problèmes. Dans le précédent exemple, la largeur totale de la boîte serait ainsi de 90 pixels dans Internet Explorer 5.x. Ce dernier considère en effet que les 5 pixels de remplissage de chaque côté font partie de la largeur de 70 pixels au lieu de s'y ajouter (voir Figure 3.3).

La propriété CSS 3 `box-sizing` permet de définir le modèle de boîte à utiliser ; elle a cependant peu de chance d'être largement utilisée sauf dans certaines circonstances bien particulières.

Figure 3.3

Le modèle de boîte propriétaire d'Internet Explorer peut rendre des éléments plus petits que prévu.



Par chance, il existe plusieurs moyens de résoudre ce problème, comme vous le verrez au Chapitre 9. La meilleure solution, et de loin, consiste cependant à éviter tout simplement le problème. Pour cela, n'ajoutez pas de remplissage aux éléments qui possèdent une largeur définie. Tâchez d'ajouter du remplissage ou des marges au parent ou à l'enfant de l'élément.

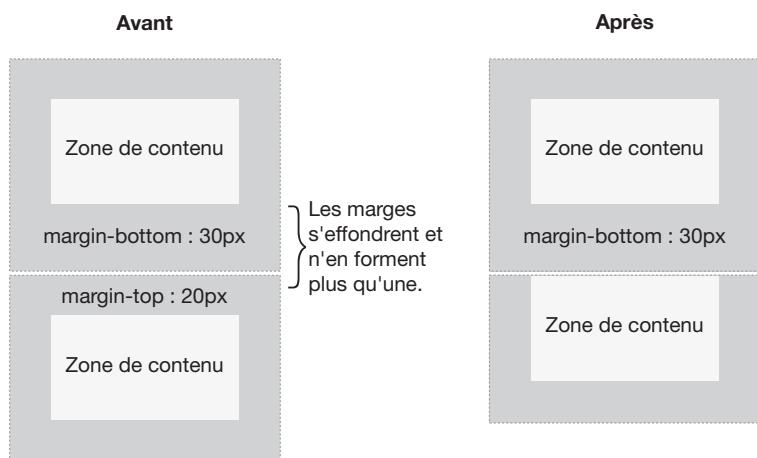
Effondrement des marges

L'effondrement des marges est un phénomène assez simple à comprendre. Il peut cependant être très déroutant lorsque vous travaillez à la structuration des mises en page. Lorsque deux marges verticales se rencontrent, elles s'effondrent l'une sur l'autre pour n'en former plus qu'une. La hauteur de cette marge fusionnée est égale à la hauteur de la plus grande des deux.

Lorsque deux éléments se trouvent l'un sur l'autre, la marge inférieure du premier élément fusionne avec la marge supérieure du second (voir Figure 3.4).

Figure 3.4

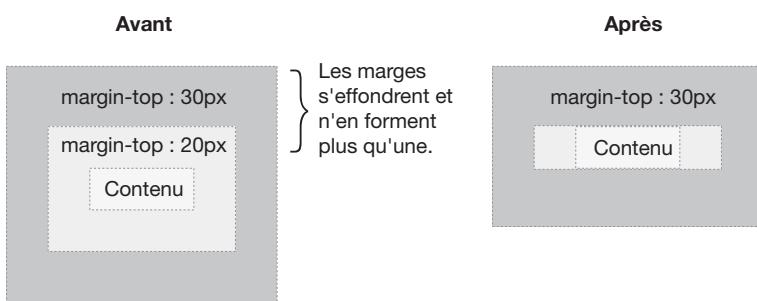
Exemple d'élément dont la marge supérieure se fusionne avec celle de l'élément précédent.



Lorsqu'un élément est contenu dans un autre élément et qu'il n'existe pas de remplissage ou de bordure séparant les marges, les marges du haut ou du bas sont fusionnées également (voir Figure 3.5).

Figure 3.5

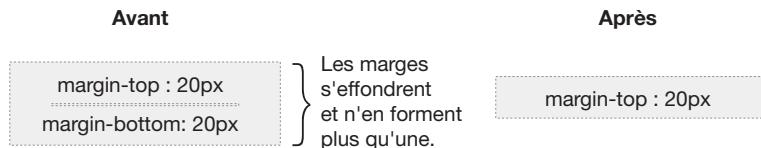
Exemple d'élément dont la marge supérieure fusionne avec la marge supérieure de son élément parent.



Aussi étrange que cela puisse paraître, les marges peuvent s'effondrer sur elles-mêmes. Si un élément vide possède une marge mais ni bordure ni remplissage, la marge supérieure touche la marge du bas et les deux fusionnent (voir Figure 3.6).

Figure 3.6

Exemple d'élément dont la marge supérieure se fusionne avec la marge inférieure.



Si cette marge touche la marge d'un autre élément, elle s'effondre elle-même à nouveau (voir Figure 3.7).

Figure 3.7

Exemple de marge effondrée d'un élément vide qui fusionne avec les marges d'un autre élément vide.

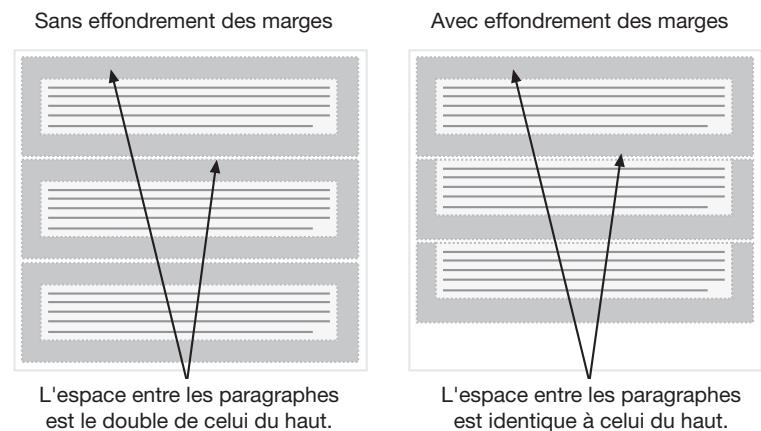


C'est pour cette raison que les séries d'éléments de paragraphe vides ne prennent que très peu d'espace, car toutes les marges s'effondrent et fusionnent pour n'en former plus qu'une.

L'effondrement des marges peut dérouter, mais il est en réalité parfaitement cohérent. Considérez une page classique composée de plusieurs paragraphes (voir Figure 3.8). L'espace au-dessus du premier paragraphe est égal à la marge supérieure. Sans l'effondrement des marges, l'espace entre tous les paragraphes subséquents correspondrait à la somme de leurs marges adjacentes. Il serait alors le double de celui du haut de la page. Grâce à l'effondrement des marges, les marges du haut et du bas de chaque paragraphe fusionnent et produisent un espacement égal partout.

Figure 3.8

Les marges s'effondrent afin que l'espace reste partout le même entre les éléments.



L'effondrement des marges ne se produit qu'avec les marges verticales des boîtes de bloc dans le flot normal du document. Les marges entre les boîtes incorporées dans les lignes, flottantes ou positionnées de manière absolue, ne s'effondrent jamais.

Récapitulatif concernant le positionnement

Maintenant que vous êtes familiarisé avec le modèle de boîte, considérons le formatage visuel et les modèles de positionnement. Il est absolument essentiel de comprendre les nuances de ces deux modèles, car ils contrôlent ensemble la manière dont chacun des éléments se dispose dans la page.

Le modèle de formatage visuel

On fait souvent référence aux éléments `p`, `h1` ou `div` comme à des éléments de niveau bloc. Cela signifie qu'ils s'affichent comme des blocs de contenu ou des boîtes de bloc. À l'inverse, les éléments `strong` ou `span` sont des éléments incorporés, car leur contenu s'affiche à l'intérieur des lignes, sous forme de boîtes incorporées dans la ligne.

Le type des boîtes peut être modifié à l'aide de la propriété `display`. Vous pouvez ainsi amener un élément incorporé, comme une ancre, à se comporter comme un élément de niveau bloc, en attribuant la valeur `block` à sa propriété `display`. Vous pouvez également faire en sorte qu'un élément ne génère pas de boîte du tout en attribuant la valeur `none` à sa propriété `display`. La boîte, et donc tout son contenu, n'est plus affichée et ne prend aucun espace dans le document.

Il existe trois mécanismes de positionnement de base en CSS : le flux normal qui joue sur les marges et remplissage, la flottaison, avec les éléments flottants, et la position, par la propriété `position`. À moins que vous ne spécifiez un autre réglage, les boîtes commencent toutes par se positionner dans le flux de base. La position de la boîte d'un élément dans le flux de base est déterminée par la position de cet élément dans le code HTML.

Les boîtes de niveau bloc apparaissent l'une après l'autre verticalement. La distance verticale entre elles est calculée en fonction de leurs marges verticales.

Les boîtes incorporées sont disposées horizontalement dans une ligne. Leur espacement horizontal peut être ajusté avec le remplissage, les bordures et les marges de gauche et de droite (voir Figure 3.9). Le remplissage, les bordures et les marges verticales n'ont en revanche aucun effet sur la hauteur de la boîte incorporée. De la même manière, le fait de définir une hauteur ou une largeur explicite sur une boîte incorporée n'a aucun effet. La boîte horizontale formée par une ligne est appelée boîte de ligne. Elle est toujours assez grande pour toutes les boîtes de ligne qu'elle contient.

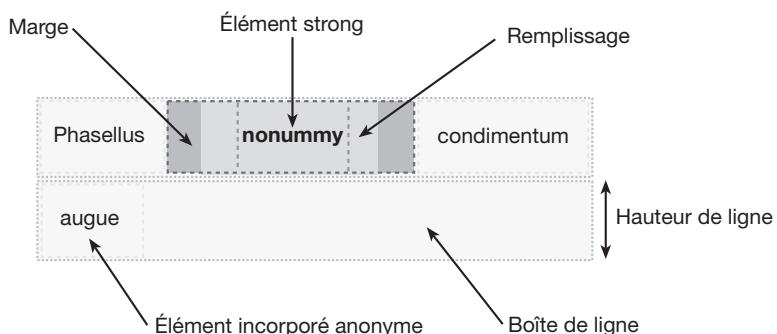
Il existe un autre piège, cependant : le fait de définir la hauteur de ligne peut augmenter la hauteur de la boîte. Voilà pourquoi le seul moyen de modifier les dimensions d'une boîte incorporée consiste à modifier la hauteur de ligne ou les bordures, le remplissage ou les marges de gauche et de droite.

Les CSS 2.1 permettent de définir la propriété `display` d'un élément en lui attribuant la valeur `inline-block`. Cette déclaration intègre l'élément dans la ligne, comme s'il s'agissait d'un élément incorporé, mais amène le contenu de la boîte à se comporter comme si la boîte était un élément de niveau bloc, ce qui permet de définir explicitement les largeurs, les hauteurs, les marges verticales et le remplissage. Dans l'ensemble, cette propriété a été mal prise en charge, ce qui explique qu'elle reste méconnue. Elle est maintenant prise en charge

par Firefox 3.0 et ses versions ultérieures. Internet Explorer 8 et les dernières versions de Safari et Opera la reconnaissent aussi. Le mode `inline-block` a donc toutes les chances d'être souvent utilisé à l'avenir.

Figure 3.9

Éléments incorporés à l'intérieur d'une boîte de ligne.



Comme les éléments HTML qui peuvent être imbriqués, les boîtes peuvent contenir d'autres boîtes. La plupart sont formées à partir d'éléments explicitement définis, mais il existe un cas où un élément de niveau bloc est créé même s'il n'a pas été explicitement défini : lorsque vous ajoutez du texte au début d'un élément de niveau bloc, comme une `div`. Même si vous n'avez pas défini le texte comme un élément de niveau bloc, il est traité comme tel :

```
<div>
  Du texte
  <p>Encore du texte</p>
</div>
```

Dans ce cas, la boîte est appelée une boîte de bloc anonyme, car elle n'est associée à aucun élément explicitement défini.

Un phénomène similaire se produit avec les lignes de texte à l'intérieur des éléments de niveau bloc. Imaginez qu'un paragraphe contienne trois lignes de texte. Chacune forme une boîte de ligne anonyme. Vous ne pouvez pas créer de bloc anonyme de style ou des boîtes de ligne directement, sauf en utilisant le pseudo-élément `:first-line`, dont l'usage est bien évidemment limité. Toutefois, il est utile de comprendre que tout ce que vous voyez à l'écran crée une boîte d'une forme ou d'une autre.

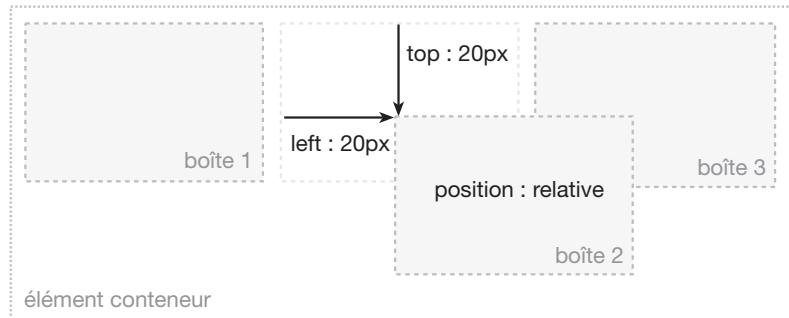
Positionnement relatif

Le positionnement relatif est assez simple à comprendre. Un élément positionné ainsi conserve exactement son emplacement dans le flux. Vous pouvez ensuite le déplacer par rapport à son point d'origine en définissant une position verticale ou horizontale. Si vous fixez la position du haut à 20 pixels, la boîte apparaît 20 pixels au-dessous de sa position normale. Si vous fixez la position gauche à 20 pixels, vous créez un espace de 20 pixels à gauche de l'élément et déplacez ce dernier vers la droite (voir Figure 3.10).

```
#myBox {
  position: relative;
  left: 20px;
  top: 20px;
}
```

Figure 3.10

Positionnement relatif d'un élément.



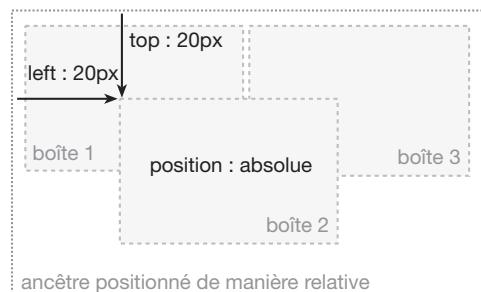
Avec le positionnement relatif, l'élément continue d'occuper l'espace d'origine, qu'il soit décalé ou non. Le décalage peut dès lors le conduire à chevaucher d'autres boîtes.

Positionnement absolu

Le positionnement relatif est considéré faire partie du modèle de positionnement standard dans le flux, car la position de l'élément est alors relative à la position normale dans le flux. À la différence, le positionnement absolu extrait l'élément du flux du document. Il n'y occupe donc plus d'espace, et les autres éléments dans le flux se comportent alors comme si l'élément positionné de manière absolue n'existant pas (voir Figure 3.11).

Figure 3.11

Positionnement absolu d'un élément.



Un élément positionné de manière absolue est positionné par rapport l'emplacement de son ancêtre le plus proche. S'il n'en possède pas, il est positionné par rapport au bloc parent initial. Selon l'agent utilisateur, il peut s'agir du support du document ou de l'élément HTML.

Comme les boîtes positionnées de manière relative, une boîte positionnée de manière absolue peut être décalée par rapport au haut, au bas, à la gauche ou à la droite de son bloc conteneur. Vous disposez ainsi d'une grande flexibilité de positionnement, qui permet de placer l'élément presque n'importe où dans la page.

Le principal problème que les utilisateurs rencontrent avec le positionnement consiste à se rappeler à quoi correspond chaque option. Le positionnement relatif est "relatif" à la position initiale de l'élément dans le flot du document, alors que le positionnement absolu est "relatif" au plus proche ancêtre positionné ou, s'il n'existe pas, au bloc conteneur initial.

Comme les boîtes positionnées de manière absolue sortent du flux du document, elles peuvent chevaucher d'autres éléments dans la page. Vous pouvez en contrôler l'ordre d'empilement en paramétrant une propriété appelée `z-index`. Plus le rang `z-index` est élevé, plus la boîte apparaît haut dans la pile.

Le positionnement absolu d'un élément par rapport à son plus proche ancêtre positionné offre d'intéressantes options pour la mise en page. Supposons ainsi que vous souhaitez aligner un paragraphe de texte en bas à droite d'une grande boîte. Vous pouvez donner à la boîte conteneur une position relative, puis positionner de manière absolue le paragraphe par rapport à cette boîte :

```
#branding {  
    width: 70em;  
    height: 10em;  
    position: relative;  
}  
  
#branding .tel {  
    position: absolute;  
    right: 1em;  
    bottom: 1em;  
    text-align: right;  
}  
  
<div id="branding">  
<p class="tel">Tel: 0845 838 6163</p>  
</div>
```

Le positionnement absolu d'une boîte par rapport à un ancêtre au positionnement relatif fonctionne bien dans la plupart des navigateurs modernes. Internet Explorer 5.5 et 6 sous Windows sont cependant victimes d'un bogue. Si vous tentez de positionner absolument la boîte par rapport à la droite ou au bas de son ancêtre positionné de manière relative, vous devez vous assurer que les dimensions de l'ancêtre sont bien spécifiées. Sans cela, Internet Explorer positionne, à tort, la boîte par rapport au support du document. Nous reviendrons sur les correctifs à ce bogue au Chapitre 9. La solution la plus simple consiste à définir la largeur et la hauteur de la boîte relative pour éviter ce problème.

Le positionnement absolu peut être utile à la structuration des pages, notamment s'il s'opère avec des ancêtres positionnés de manière relative. Il est parfaitement possible de créer une structure de page en n'utilisant que des positionnements absolus. Pour que cela fonctionne, il faut que ces éléments aient des dimensions fixes, afin qu'on puisse les positionner où on le souhaite sans qu'ils se chevauchent.

Parce qu'ils sortent du flux normal du document, les éléments positionnés de manière absolue n'ont pas d'effet sur les boîtes dans le flux normal. Si vous agrandissez une boîte positionnée de manière absolue (par exemple, en augmentant la taille des polices), les boîtes situées autour ne se réorganisent pas pour en tenir compte. La moindre modification de taille peut ainsi détruire les subtiles proportions de vos maquettes en provoquant le chevauchement des boîtes positionnées de manière absolue.

Positionnement fixe

Le positionnement fixe est une sous-catégorie du positionnement absolu. La seule différence est que, dans ce cas, le bloc conteneur de l'élément fixe n'est autre que la fenêtre d'affichage elle-même. Le positionnement fixe permet ainsi de créer des éléments flottants qui restent toujours au même endroit dans la fenêtre. L'ancien site web **snook.ca** (voir Figure 3.12) en propose un exemple. Le formulaire pour les commentaires est positionné de manière fixe afin de rester ancré au même endroit dans l'écran quand la page défile. Cette astuce améliore l'utilisabilité du site, et évite à l'utilisateur de faire défiler la page pour laisser son commentaire.



Figure 3.12

Dans l'ancien site web **snook.ca**, le formulaire pour les commentaires situé dans la marge à droite de l'écran utilisait une position fixe pour rester au même endroit dans la fenêtre d'affichage

Malheureusement, Internet Explorer 6 et ses versions antérieures ne prennent pas en charge le positionnement fixe. Internet Explorer 7 le reconnaît partiellement, mais avec quelques bogues en pratique. Pour contourner ces problèmes, Jonathan Snook a ici utilisé du code JavaScript pour répliquer l'effet dans Internet Explorer.

Flottement

Le dernier modèle de mise en forme visuelle est le modèle de flottement. Une boîte flottante peut être calée à gauche ou à droite, de manière que son bord externe touche le bord de sa

boîte conteneur ou d'une autre boîte flottante. Les boîtes flottantes ne s'insèrent pas dans le flux normal du document, si bien que les boîtes de bloc qui sont dans le flot se comportent comme si les boîtes flottantes n'existaient pas.

Comme le montre la Figure 3.13, la boîte 1, qui flotte à droite, sort du flux du document et se décale à droite jusqu'à ce que son bord droit touche le bord droit du bloc conteneur.



Figure 3.13

Exemple d'élément flottant à droite.

À la Figure 3.14, la boîte 1, qui flotte à gauche, sort du flot du document et se décale à gauche jusqu'à ce que son bord gauche touche le bord gauche du bloc conteneur. Comme elle n'est plus dans le flot, elle ne prend aucun espace et s'affiche par-dessus la boîte 2 qu'elle masque complètement. Si vous faites flotter les trois boîtes à gauche, la boîte 1 est décalée à gauche jusqu'à toucher sa boîte conteneur, et les deux autres boîtes sont décalées à gauche jusqu'à toucher la boîte flottante précédente.



Figure 3.14

Exemple d'éléments flottant à gauche.

Si le bloc conteneur est trop étroit pour que tous les éléments flottants tiennent à l'horizontale, le reste des éléments flottants est renvoyé plus bas, jusqu'à ce qu'il y ait assez de place (voir Figure 3.15). Si les éléments flottants possèdent des hauteurs différentes, des éléments flottants viennent buter sur d'autres éléments flottants en se décalant vers le bas.

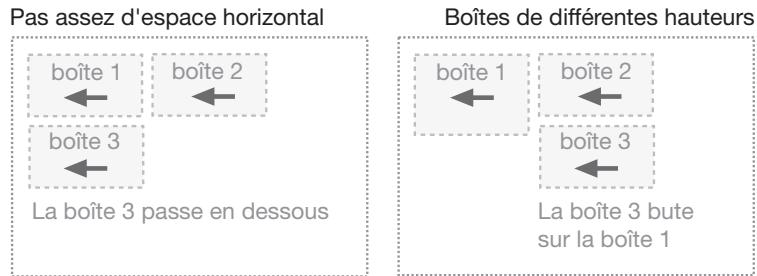


Figure 3.15

S'il n'y a pas assez d'espace sur le plan horizontal, les éléments flottants se décalent vers le bas jusqu'à trouver de la place.

Boîtes de ligne et dégagement

À la précédente section, vous avez vu que le flottement d'un élément le sortait du flot du document et qu'il n'y exerçait donc plus d'effet sur les éléments non flottants. En réalité, ce n'est pas exactement vrai. Si un élément flottant est suivi par un élément dans le flot du document, sa boîte se comporte comme si le flottement n'existant pas. Le contenu textuel de la boîte conserve néanmoins en mémoire la présence de l'élément flottant et se décale afin de lui laisser de la place. En termes techniques, une boîte de ligne adjacente à un élément flottant est raccourcie afin de faire place à l'élément flottant et d'habiller ainsi la boîte flottante. En fait, les éléments flottants ont été créés pour permettre au texte d'habiller les images en les bordant (voir Figure 3.16).

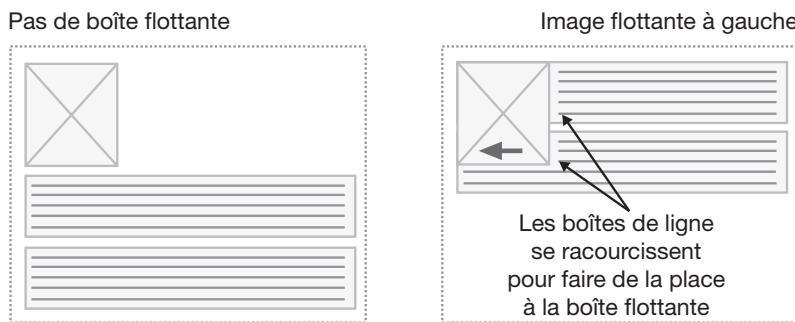
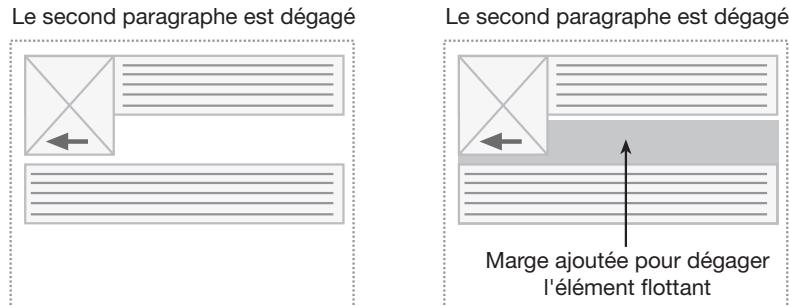


Figure 3.16

Les boîtes de ligne se raccourcissent à côté des éléments flottants.

Pour empêcher les boîtes de ligne de courir le long des boîtes flottantes, vous devez appliquer une propriété `clear` à l'élément qui contient ces boîtes de ligne. La propriété `clear` peut valoir `left`, `right`, `both` ou `none`. Elle indique le côté de la boîte qui ne doit pas se trouver à côté d'une boîte flottante. J'ai toujours considéré cette propriété comme une sorte de drapeau magique qui annulait automatiquement le flottement précédent. En réalité, c'est encore plus intéressant. Lorsque vous dégagéz un élément avec la propriété `clear`, le navigateur ajoute suffisamment de marge au-dessus pour décaler sa bordure supérieure vers le bas, au-delà de l'élément flottant (voir Figure 3.17).

**Figure 3.17**

Le dégagement de la marge supérieure d'un élément crée suffisamment d'espace vertical pour l'élément flottant précédent.

Comme vous pouvez le voir, les éléments flottants sortent du flux du document et n'ont pas d'effet sur les éléments qui les entourent. Toutefois, le dégagement d'un élément dégage en fait un espace vertical pour tous les éléments flottants précédents.

Ce mécanisme peut être utile pour la mise en page car il permet aux éléments alentour de faire de la place aux éléments flottants. Il permet également de résoudre le problème signalé précédemment avec le positionnement absolu, où les modifications de hauteur n'affectent pas les éléments alentour et peuvent rompre les équilibres de la mise en page.

Considérons un peu plus précisément cette question des éléments flottants et de la propriété `clear`. Supposons que vous ayez une image et que vous souhaitez la faire flotter à gauche d'un bloc de texte. Vous voulez que cette image et le texte soient contenus dans un autre élément avec une couleur d'arrière-plan et une bordure. D'emblée, vous tenterez sans doute de procéder comme ceci :

```
.news {
  background-color: gray;
  border: solid 1px black;
}

.news img {
  float: left;
}

.news p {
  float: right;
}

<div class="news">

<p>Du texte</p>
</div>
```

Pourtant, comme les éléments flottants sont sortis du flux du document, la div qui les enveloppe ne prend aucun espace. Dans ce cas, comment faire pour que l'enveloppe englobe

l'élément flottant ? Vous devez appliquer une propriété `clear` quelque part dans cet élément (voir Figure 3.18). Comme il n'existe pas d'élément à dégager dans cet exemple, vous pouvez ajouter un élément vide sous le dernier paragraphe et le dégager.

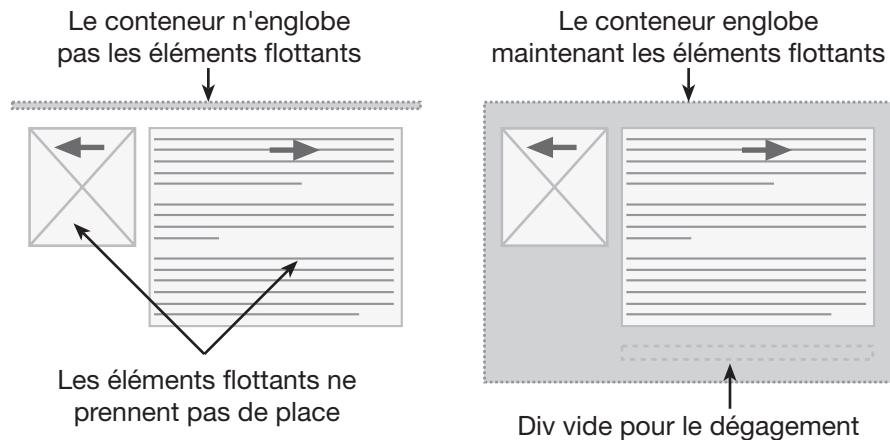


Figure 3.18

Comme les éléments flottants ne prennent pas d'espace, ils ne sont pas englobés par les éléments conteneurs. L'ajout d'un élément vide permet d'appliquer le dégagement et force l'élément conteneur à englober les éléments flottants.

```
.news {
  background-color: gray;
  border: solid 1px black;
}

.news img {
  float: left;
}

.news p {
  float: right;
}

.clear {
  clear: both;
}

<div class="news">

<p>Du texte</p>
<br class="clear" />
</div>
```

On obtient ainsi le résultat souhaité, mais en ajoutant du code superflu. Souvent, on trouve un élément présent auquel appliquer la propriété `clear`, mais il faut parfois manger son chapeau et ajouter du code sans pertinence sémantique, aux seules fins de la mise en forme.

Au lieu de dégager le texte et l'image qui flottent, vous pouvez choisir de faire flotter la div conteneur également :

```
.news {  
    background-color: gray;  
    border: solid 1px black;  
    float: left;  
}  
  
.news img {  
    float: left;  
}  
  
.news p {  
    float: right;  
}  
  
<div class="news">  
      
    <p>Du texte</p>  
</div>
```

Vous obtenez alors l'effet souhaité mais, du coup, c'est l'élément suivant qui est maintenant affecté par le flottement. Pour résoudre ce problème, certains développeurs choisissent de faire flotter presque tous les éléments de leur maquette, puis dégagent ces éléments flottants avec un élément qui a sa raison d'être sémantique, comme le pied de page du site. Cette approche réduit ou élimine les balises superflues, mais le flottement peut être compliqué et certains navigateurs plus anciens ont de la peine avec les mises en page flottantes complexes. C'est la raison pour laquelle bien des concepteurs se résignent à utiliser quelques balises superflues.

La propriété `overflow` définit la manière dont un élément est censé se comporter si le contenu englobé est trop grand pour les dimensions spécifiées. Par défaut, le contenu déborde de la boîte, en chevauchant l'espace adjacent. Mais l'un des effets secondaires liés à l'application des valeurs `hidden` ou `auto` à la propriété `overflow` tient à ce que se trouvent ainsi dégagés tous les éléments flottants qui y sont contenus. C'est ainsi un moyen utile de dégager un élément sans ajouter de balise superflue. Cette méthode n'est pas toujours la plus appropriée, car la définition de la propriété `overflow` affecte son comportement. En particulier, elle peut forcer l'affichage de barres de défilement ou détourer du contenu dans certains cas.

Certaines personnes ont adopté une méthode pour dégager les éléments flottants, qui consiste à utiliser du contenu généré par le code CSS ou par un script JavaScript. Le concept de base est alors le même. Au lieu d'ajouter un élément qui permet d'effectuer le dégagement directement dans le balisage, vous l'ajoutez dynamiquement à la page. Dans les deux cas, vous devez indiquer où vient se placer l'élément à dégager, ce qui se fait généralement avec l'ajout d'un nom de classe :

```
<div class="news clear">  
      
    <p>Du texte</p>  
</div>
```

Avec la méthode CSS, vous utilisez la pseudo-classe :`after` en combinaison avec la déclaration de contenu pour ajouter du nouveau contenu à la fin du contenu existant spécifié. Dans le cas présent, j'ajoute un point, car c'est un caractère petit et discret. Il ne faut pas que le nouveau contenu prenne de l'espace vertical ou soit affiché dans la page. La propriété `height` doit donc prendre la valeur `0` et la propriété `visibility` la valeur `hidden`. Comme ces éléments dégagés récupèrent de l'espace pour leur marge supérieure, il faut que la propriété `display` du contenu généré possède la valeur `block`. Une fois cela fait, vous pouvez dégager votre contenu généré :

```
.clear:after {  
    content: ".";  
    height: 0;  
    visibility: hidden;  
    display: block;  
    clear: both;  
}
```

Cette méthode fonctionne dans la plupart des navigateurs web, mais elle échoue avec Internet Explorer 6 et ses versions antérieures. Il existe plusieurs solutions de contournement, dont un grand nombre sont présentées à l'adresse www.positioniseverything.net/easyclearing.html. La plus courante nécessite d'utiliser le Saint Hack (voir Chapitre 8) pour piéger Internet Explorer 5 et 6 et leur faire appliquer une "structuration" (voir Chapitre 9) qui dégage les éléments flottants.

```
.clear {  
    display: inline-block;  
}  
/* Le Saint Hack ne cible qu'Internet Explorer sous Windows */  
* html .clear {height: 1%;}  
.clear {display: block;}  
/* Fin du Saint Hack */
```

Cette méthode, parce qu'elle est assez complexe, ne convient pas nécessairement à tout le monde. Nous ne l'avons finalement incluse que par souci d'exhaustivité.

La méthode JavaScript sort du cadre de ce livre, mais on peut la mentionner rapidement. À la différence de la précédente, elle fonctionne avec tous les navigateurs principaux pourvu que JavaScript n'y soit pas désactivé. Si vous l'utilisez, vous devez cependant vous assurer que le contenu ne reste pas lisible dans les navigateurs où les fonctionnalités JavaScript sont désactivées.

En résumé

Vous avez découvert le modèle de boîte et vu comment le remplissage, les marges, la largeur et la hauteur changent les dimensions d'une boîte. Vous avez également découvert le principe d'effondrement des marges et vu comment il peut influer sur vos mises en page. Vous avez découvert les trois modèles de mise en forme CSS : le flux normal, le positionnement et le flottement. Vous avez appris la différence entre les boîtes incorporées et les boîtes de bloc, vous avez vu comment positionner un élément de manière absolue par rapport

à un ancêtre positionné de manière relative et comment fonctionne le dégagement avec la propriété `clear`.

Il est temps de mettre ces connaissances en pratique. Lors des prochains chapitres, vous découvrirez un certain nombre de concepts CSS essentiels et vous verrez comment ils peuvent être utilisés de manière utile et pratique. Ouvrez votre éditeur de texte favori et commençons à programmer !

4

Utilisation des effets d'arrière-plan

Maintenant que vous êtes armé sur le plan théorique, passons à la pratique. Le Web est aujourd’hui un média particulièrement axé sur l’expression visuelle. La petite balise d’image (au départ, simple moyen d’ajouter du contenu visuel aux sites web) a permis aux concepteurs web de transformer d’ennuyeux documents en véritables expériences graphiques. Les graphistes s’en sont rapidement emparés pour embellir la présentation des pages. En fait, sans l’invention de la balise d’image, la profession de concepteur web n’aurait sans doute jamais évolué.

Malheureusement, cette balise a trop souvent servi à encombrer les pages d’images uniquement destinées à la présentation. Mais les CSS permettent heureusement d’afficher des images sans les intégrer dans le balisage, comme arrière-plan d’un élément existant. Au travers d’une série d’exemples pratiques, vous allez voir ici comment les images d’arrière-plan peuvent être utilisées pour créer une variété de techniques intéressantes et pratiques.

Au cours de ce chapitre, vous en apprendrez plus sur :

- les boîtes à bords arrondis à largeur fixe et flexible ;
- la technique des portes coulissantes ;
- les images multiples d’arrière-plan et la propriété `border-radius` ;
- les ombres portées CSS ;
- les propriétés `opacity` et `RGBa` ;
- l’affichage des PNG dans les anciennes versions d’Internet Explorer ;
- le défilement parallaxe ;
- le remplacement d’image.

Notions de base sur les images d’arrière-plan

Rien de plus simple que d’appliquer une image d’arrière-plan. Par exemple, si vous souhaitez un bel arrière-plan mosaïqué pour votre site, vous pouvez appliquer l’image comme arrière-plan à l’élément `body` :

```
body {  
background-image:url(/img/pattern.gif);  
}
```

Le comportement par défaut du navigateur consiste à répéter l’image horizontalement et verticalement afin qu’elle remplisse toute la page par un effet de mosaïque. Pour un contrôle plus précis, vous pouvez choisir si l’image d’arrière-plan doit se répéter verticalement, horizontalement ou pas du tout.

Les dégradés sont très en vogue. Imaginons donc que vous souhaitiez plutôt appliquer un dégradé vertical. Pour cela, créez une image de dégradé étroite mais très longue, puis appliquez-la à l'élément body de la page en la répétant horizontalement.

```
body {  
    background-image: url(/img/gradient.gif);  
    background-repeat: repeat-x;  
}
```

Comme le dégradé possède une hauteur fixe, il s'arrête brusquement si le contenu de la page est plus long que la hauteur de l'image. On peut, certes, créer une image très longue, qui se fond dans une zone de couleur unie couvrant tout le bas de l'image, mais il est difficile de prédire systématiquement la longueur des pages. Le plus simple est donc d'ajouter une couleur d'arrière-plan également. Les images d'arrière-plan viennent toujours se placer par-dessus la couleur d'arrière-plan, si bien que lorsque l'image s'interrompt, c'est la couleur d'arrière-plan qui prend le relais. Si vous choisissez une couleur d'arrière-plan identique à celle du bas du dégradé, la transition entre l'image et la couleur d'arrière-plan est parfaitement invisible.

```
body {  
    background-image: url(/img/gradient.gif);  
    background-repeat: repeat-x;  
    background-color: #ccc;  
}
```

Les mosaïques d'images peuvent être utiles mais, la plupart du temps, vous souhaiterez ajouter vos images sans les répéter dans la page. Par exemple, supposons que vous vouliez faire commencer votre page web par une grande image représentant votre marque. Vous pourriez ajouter directement l'image à la page et, dans bien des cas, il s'agirait de la bonne méthode, mais si l'image ne contient aucune information et n'est destinée qu'à des fins de présentation, il est préférable de la séparer du reste du contenu. Vous pouvez le faire en créant un *hook* (une sorte de point d'attache) pour elle dans le code HTML et en appliquant cette image avec des CSS. Dans l'exemple suivant, j'ai ajouté une div vide au balisage et je lui ai donné l'ID **branding**. Vous définissez ensuite les dimensions de la div pour les rendre identiques à celles de l'image, en l'appliquant comme arrière-plan, sans répétition.

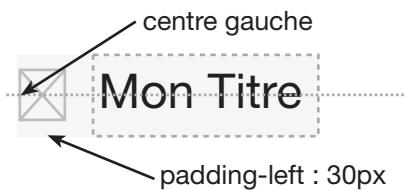
```
#branding {  
    width: 700px;  
    height: 200px;  
    background-image: url(/img/branding.gif)  
    background-repeat: no-repeat;  
}
```

Enfin, vous pouvez définir la position de l'image d'arrière-plan. Par exemple, supposons que vous souhaitiez ajouter une puce à chaque titre de votre site (voir Figure 4.1). Vous pouvez alors procéder de la manière suivante :

```
h1 {  
    padding-left: 30px;  
    background-image: url(/img/bullet.gif);  
    background-repeat: no-repeat;  
    background-position: left center;  
}
```

Figure 4.1

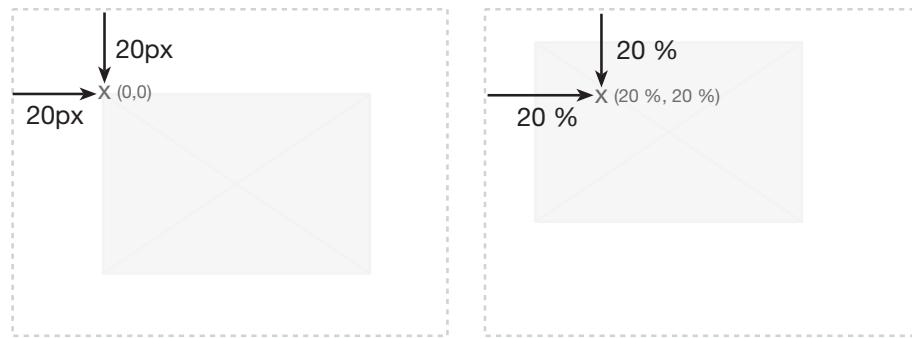
Création d'une puce à l'aide d'une image d'arrière-plan.



Les deux derniers mots-clés indiquent le positionnement de l'image. Dans le cas présent, l'image sera positionnée à gauche de l'élément et centrée verticalement. En plus des mots-clés, vous pouvez définir la position de l'image d'arrière-plan à l'aide d'unités comme des pixels ou des pourcentages.

Si vous définissez la position de l'arrière-plan en pixels ou en cadratins (em), vous positionnez le coin supérieur gauche de l'image en le décalant du nombre d'unités spécifié, par rapport au coin supérieur gauche de l'élément. Si vous spécifiez une position verticale et horizontale de 20 pixels, le coin supérieur gauche de l'image apparaît ainsi à 20 pixels du coin supérieur gauche de l'élément. Le positionnement de l'arrière-plan en pourcentage fonctionne un peu différemment. Au lieu de positionner le coin supérieur gauche de l'image, il utilise un point correspondant dans l'image. Si vous définissez une position verticale et horizontale de 20 %, vous positionnez en fait un point, situé à 20 % du coin supérieur gauche de l'image, afin de le placer à 20 % du coin supérieur gauche de l'élément parent (voir Figure 4.2).

Positionnement de l'arrière-plan en px Positionnement de l'arrière-plan en %

**Figure 4.2**

Lorsque vous positionnez des images d'arrière-plan en utilisant des pixels, c'est le coin supérieur gauche de l'image qui est pris comme référence. Lorsque vous effectuez un positionnement en pourcentage, c'est la position correspondante dans l'image qui est prise comme référence.

Si vous souhaitez positionner le précédent exemple de puces avec des pourcentages et non des mots-clés, vous pouvez fixer la position verticale à 50 % pour centrer l'image de la puce :

```
h1 {
  padding-left: 30px;
```

```
background-image: url(/img/bullet.gif);  
background-repeat: no-repeat;  
background-position: 0 50%;  
}
```

La spécification CSS indique que vous n'êtes pas censé mélanger des unités telles que des pixels ou des pourcentages avec des mots-clés. Cette règle n'a pas beaucoup de sens et, de fait, bon nombre des navigateurs web l'ignorent délibérément. Quoi qu'il en soit, le mélange des unités et des mots-clés échoue sur certains navigateurs et risque de rendre vos CSS invalides. Cette option n'est donc pas judicieuse, pour le moment.

Pour gagner du temps, les CSS proposent aussi une version raccourcie de la propriété `background`. Elle permet de définir toutes les propriétés en une fois, sans les détailler de manière individuelle.

```
h1 {  
  background: #ccc url(/img/bullet.gif) no-repeat left center;  
}
```

Si le fonctionnement des images d'arrière-plan est assez simple à comprendre, il n'en forme pas moins la base de nombreuses techniques CSS avancées.

Boîtes à bords arrondis

L'une des principales critiques qu'essuient les mises en page CSS tient au fait qu'elles sont très carrées, structurées de manière rectangulaire. Certains concepteurs, pour parer à ce problème, ont alors commencé à concevoir des pages en incorporant plus de formes courbes. Les rectangles à bords arrondis sont rapidement devenus l'une des techniques les plus recherchées en CSS. Il existe plusieurs moyens de créer des boîtes à bords arrondis. Chacun a ses avantages et ses inconvénients. Votre choix dépendra en grande partie des circonstances dans lesquelles vous souhaiterez les utiliser.

Boîtes à largeur fixe et bords arrondis

Les boîtes de largeur fixe à bords arrondis sont les plus faciles à créer. Elles ne requièrent que deux images : l'une en haut de la boîte et l'autre en bas. Par exemple, supposons que vous souhaitiez créer une boîte comme celle de la Figure 4.3.

Figure 4.3

Une boîte simple à bords arrondis

Section 2

Lorem ipsum dolor sit amet,
 consectetuer adipiscing elit.
 Proin venenatis turpis ut
 quam. In dolor. Nam ultrices
 nisl sollicitudin sapien. Ut
 lacinia aliquet ante.

Le balisage pour la boîte ressemble à ceci :

```
<div class="box">
  <h2>Titre</h2>
  <p>Contenu</p>
</div>
```

Dans votre logiciel d'édition d'image favori, vous devez créer deux images comme celles de la Figure 4.4 : l'une pour le haut de la boîte et l'autre pour le bas. Le code et les images de cet exemple comme ceux de tous les autres exemples du livre peuvent être téléchargés sur le site www.pearson.fr.

Figure 4.4

Les images courbes
du haut et du bas.



top.gif

bottom.gif

Ensuite, appliquez l'image du haut à l'élément titre et celle du bas à la div de la boîte. Ce style de boîte possédant un remplissage uni, vous devez créer le corps de la boîte en ajoutant une couleur d'arrière-plan à la div de la boîte :

```
.box {
  width: 418px;
  background: #efffce7 url(/img/bottom.gif) no-repeat left bottom;
}

.box h2 {
  background: url(/img/top.gif) no-repeat left top;
}
```

Comme il ne faut pas que le contenu vienne buter sur les bords de la boîte, vous devez aussi ajouter un remplissage aux éléments dans la div :

```
.box {
  width: 418px;
  background: #efffce7 url(/img/bottom.gif) no-repeat left bottom;
  padding-bottom: 1px;
}

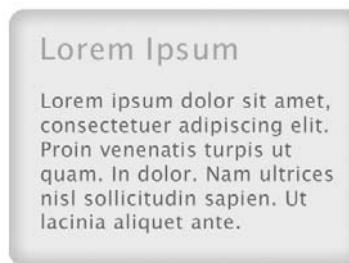
.box h2 {
  background: url(/img/top.gif) no-repeat left top;
  margin-top: 0;
  padding: 20px 20px 0 20px;
}

.box p {
  padding: 0 20px;
}
```

Voilà qui est parfait pour une boîte simple de couleur unie et sans bordures, mais comment créer un style un peu plus sophistiqué, comme celui de la Figure 4.5 ?

Figure 4.5

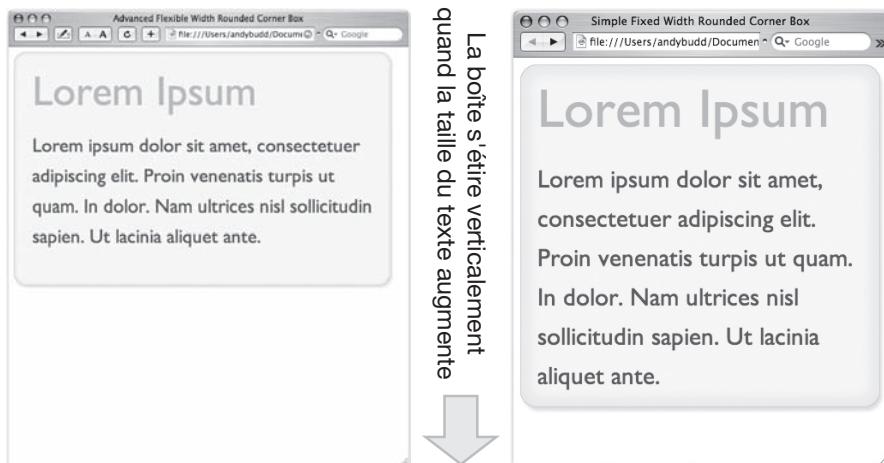
Exemple stylé de boîte
à bords arrondis.



La technique est alors la même, sauf qu'au lieu de définir une couleur d'arrière-plan pour la boîte vous devez choisir une image d'arrière-plan répétée. Pour que l'effet fonctionne, vous allez devoir appliquer l'image courbe du bas à un autre élément. Dans cet exemple, j'ai utilisé le dernier élément de paragraphe de la boîte :

```
.box {  
    width: 424px;  
    background: url(/img/tile2.gif) repeat-y;  
}  
  
.box h2 {  
    background: url(/img/top2.gif) no-repeat left top;  
    padding-top: 20px;  
}  
  
.box .last {  
    background: url(/img/bottom2.gif) no-repeat left bottom;  
    padding-bottom: 20px;  
}  
  
.box h2, .box p {  
    padding-left: 20px;  
    padding-right: 20px;  
}  
  
<div class="box">  
    <h2>Titre</h2>  
    <p class="last">Contenu</p>  
</div>
```

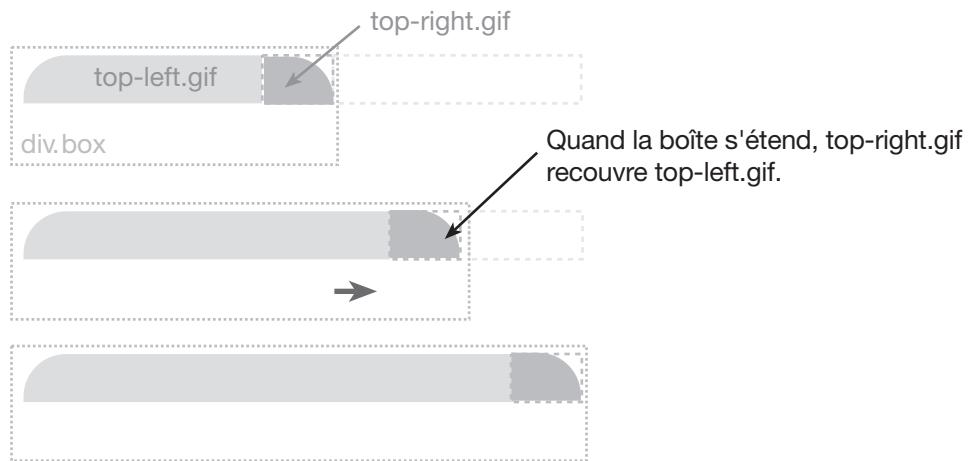
La Figure 4.6 présente la boîte mise en forme. Aucune hauteur ne lui ayant été donnée, elle s'étend verticalement quand la taille du texte augmente.

**Figure 4.6**

Boîte à largeur fixe mise en forme. La hauteur de la boîte s'étire pour faire de la place au texte.
La boîte s'étire verticalement quand la taille du texte augmente.

Boîte flexible à bords arrondis

Dans les exemples précédents, la boîte s'étend quand la taille du texte varie, mais elle ne s'étend pas horizontalement, car sa largeur doit rester la même que celle des images du haut et du bas. Si vous souhaitez créer une boîte flexible, vous devez procéder autrement. Au lieu de n'utiliser qu'une seule image pour les courbes du haut et du bas, vous devez en utiliser deux qui se chevauchent (voir Figure 4.7).

**Figure 4.7**

Ce schéma montre comment les images du haut s'étendent afin de former une boîte flexible à bords arrondis.

Lorsque la taille de la boîte augmente, une portion plus large de l'image est révélée, ce qui crée l'illusion que la boîte s'étend. C'est la technique des "portes coulissantes" : une image coulisse au-dessus de l'autre en la masquant. Il faut plus d'images pour cette technique et il faut, en outre, ajouter deux éléments non sémantiques dans le code.

```
<div class="box">
  <div class="box-outer">
    <div class="box-inner">
      <h2>Titre</h2>
      <p>Contenu</p>
    </div>
  </div>
</div>
```

Cette méthode requiert quatre images en tout : deux qui composent l'arrondi du haut et deux qui composent celui du bas (voir Figure 4.8). Les images du bas doivent être aussi hautes que la hauteur maximale de la boîte. Nous nommerons ces images top-left.gif, top-right.gif, bottom-left.gif et bottom-right.gif.

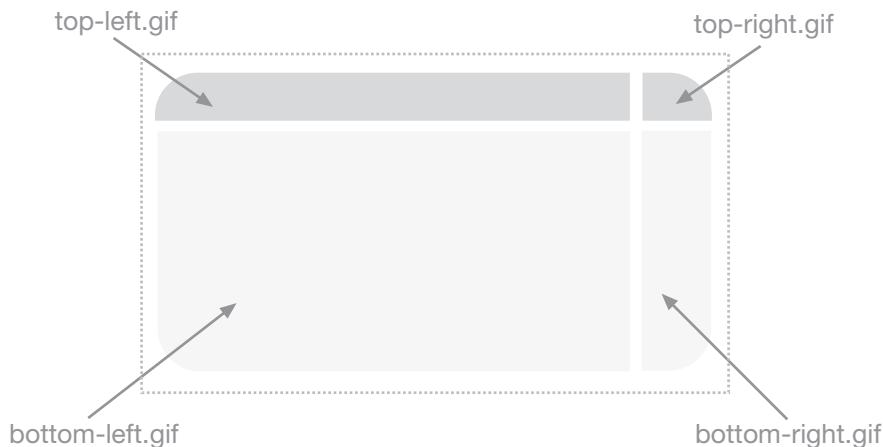


Figure 4.8

Les images requises pour créer la boîte flexible à bords arrondis. .

Pour commencer, vous devez appliquer l'image bottom-left.gif à la div principale de la boîte et bottom-right.gif à la div externe. Ensuite, appliquez top-left.gif à la div interne et top-right.gif au titre. Pour finir, ajoutez un remplissage pour espacer légèrement le contenu de la boîte.

```
.box {
  width: 20em;
  background: #efffce7 url(/img/bottom-left.gif) no-repeat left bottom;
}

.box-outer {
  background: url(/img/bottom-right.gif) no-repeat right bottom;
```

```

padding-bottom: 1em;
}

.box-inner {
  background: url(/img/top-left.gif) no-repeat left top;
}

.box h2 {
  background: url(/img/top-right.gif) no-repeat right top;
  padding-top: 1em;
}

.box h2, .box p {
  padding-left: 1em;
  padding-right: 1em;
}

```

Dans cet exemple, j'ai défini la largeur de la boîte en cadratins (em). La boîte peut ainsi s'étendre quand l'utilisateur augmente la taille du texte dans le navigateur (voir Figure 4.9). Vous pouvez cependant aussi définir cette largeur en pourcentage, afin que la boîte s'étende ou se contracte en fonction de la taille de la fenêtre du navigateur. Il s'agit là d'un des principes fondamentaux des mises en page élastiques et liquides, comme nous le verrons ultérieurement.



Figure 4.9

Les boîtes flexibles à bords arrondis s'étendent à la fois horizontalement et verticalement quand le texte change de taille.

L'ajout d'éléments non sémantiques n'est pas idéal. Si vous ne créez qu'une ou deux boîtes, ce n'est sans doute pas gênant, mais si cela devient un problème, vous pouvez envisager d'insérer ces éléments supplémentaires en JavaScript. Pour plus de détails à ce sujet, consultez l'excellent article de Roger Johansson sur le site 456 Berea Street (en anglais), à l'adresse www.456bereastreet.com/archive/200505/transparent_custom_corners_and_borders.

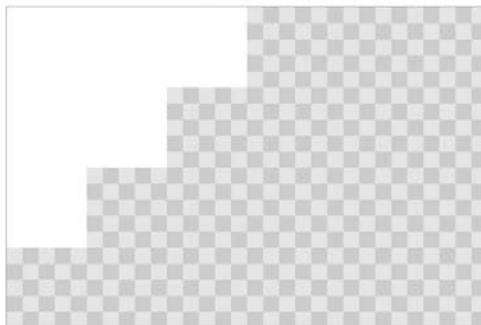
Coins masqués

La technique des coins masqués (en anglais, "mountaintop corners", par référence aux sommets neigeux qui se terminent par du blanc) est une technique simple mais très efficace élaborée par Dan Cederholm (www.simplebits.com), l'auteur du best-seller *Web Standards Solutions* (Friends of ED, 2004, réédition 2009). Supposons que vous souhaitez créer une variété de boîtes arrondies de différentes couleurs. Avec les précédentes méthodes, vous seriez contraint de créer des images différentes pour chacun des thèmes de couleur. Vous pourriez le faire avec deux ou trois thèmes de couleur, mais comment faire si vous souhaitez que vos utilisateurs aient la liberté de créer leurs propres thèmes ? Vous seriez contraint de créer des images dynamiquement sur le serveur, ce qui deviendrait très compliqué.

Il existe heureusement un autre moyen. Au lieu de créer des images de coins colorés, vous pouvez créer des masques de coins (voir Figure 4.10). La zone masquée correspond à la couleur d'arrière-plan utilisée tandis que la zone du coin lui-même est transparente. Lorsque l'image est placée sur une boîte colorée, elle crée alors l'impression que ses bords sont arrondis (voir Figure 4.11).

Figure 4.10

Avec un masque de coin, le masque blanc couvre la couleur d'arrière-plan pour créer un effet arrondi.



Comme ces masques de coin doivent être des images bitmap, mieux vaut-il utiliser des courbes subtiles. Si vous tracez un arrondi trop prononcé, il risque de sembler édenté ou disgracieux.

Le code de base est analogue à celui de la précédente méthode. Il requiert quatre éléments auxquels appliquer les quatre masques de coin :

```
<div class="box">
  <div class="box-outer">
    <div class="box-inner">
      <h2>Titre</h2>
      <p>Contenu</p>
    </div>
  </div>
</div>
```

Le code CSS est aussi très similaire :

```
.box {
  width: 20em;
```

```

background: #effce7 url(/img/bottom-left.gif) É
no-repeat left bottom;
}

.box-outer {
background: url(/img/bottom-right.gif) no-repeat right bottom;
padding-bottom: 5%;
}

.box-inner {
background: url(/img/top-left.gif) no-repeat left top;
}

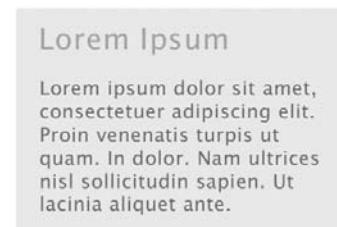
.box h2 {
background: url(/img/top-right.gif) no-repeat right top;
padding-top: 5%;
}

.box h2, .box p {
padding-left: 5%;
padding-right: 5%;
}

```

Figure 4.11

Boîte à coins masqués.



La principale différence, hormis le fait que vous utilisez des images différentes, tient à l'ajout d'une couleur d'arrière-plan à la div principale de la boîte. Si vous souhaitez modifier la couleur de la boîte, il suffit de modifier la valeur de couleur dans le fichier CSS, sans créer de nouvelle image. Cette méthode ne convient que pour des boîtes très simples, mais elle est d'une extrême flexibilité et peut être utilisée à l'infini dans différents projets.

Images d'arrière-plan multiples

Les précédents exemples sont très intéressants, mais ils imposent presque tous d'ajouter des éléments non sémantiques dans le code, car il n'est possible d'ajouter qu'une seule image d'arrière-plan par élément. Ne serait-il pas formidable qu'on puisse ajouter plusieurs images d'arrière-plan à la fois ? Les CSS 3 le permettent. La syntaxe est, en outre, d'une simplicité absolue et prend la même forme que pour les images d'arrière-plan standard.

Seule différence : au lieu de définir une seule image d'arrière-plan, vous pouvez utiliser autant d'images que vous le souhaitez. Voici le principe :

```
.box {
    background-image: url(/img/top-left.gif),
                      url(/img/top-right.gif),
                      url(/img/bottom-left.gif),
                      url(/img/bottom-right.gif);
    background-repeat: no-repeat,
                      no-repeat,
                      no-repeat,
                      no-repeat;
    background-position: top left,
                         top right,
                         bottom left,
                         bottom right;
}

<div class="box">
    <h2>Titre</h2>
    <p>Contenu</p>
</div>
```

Commencez par définir toutes les images que vous souhaitez utiliser avec la propriété `background-image`. Ensuite, indiquez si vous voulez les répéter ou non. Puis, fixez leurs positions respectives avec la propriété `background-position`. Le résultat est présenté à la Figure 4.12. Safari prenait en charge les images d'arrière-plan multiples dès sa version 1.3. Firefox et Opera ont emboîté le pas dans leurs dernières versions. Internet Explorer ne les reconnaît pas encore, mais ne vous privez pas d'utiliser cette technique si vous estimez qu'il est acceptable que les utilisateurs d'Internet Explorer voient des coins carrés à la place.

Figure 4.12

Une boîte à bords arrondis créée avec des arrière-plans multiples en CSS 3.



border-radius

À une époque où pullulent les jeux vidéo haute définition qui redessinent des textures à la volée, il ne paraîtrait pas totalement insensé que les navigateurs soient capables de dessiner par eux-mêmes des boîtes simples à bords arrondis, sans utiliser d'images vectorielles.

Ce rêve un peu fou devient aujourd’hui réalité, grâce à la nouvelle propriété CSS 3 `border-radius`. Tout ce que vous avez à faire est de définir le rayon de l’arrondi désiré ; le navigateur s’occupe du reste (voir Figure 4.13).

```
.box {  
    border-radius: 1em;  
}
```

Figure 4.13

Une boîte à bords arrondis créée avec la propriété CSS 3 `border-radius`.



Cette propriété étant récente, des désaccords demeurent sur la manière dont elle devrait fonctionner. Jusqu'à ce qu'elle soit uniformément acceptée, vous devrez donc utiliser des extensions spécifiques des navigateurs pour l'invoquer. Actuellement, seuls Firefox et Safari la prennent en charge. C'est la raison pour laquelle j'utilise les préfixes `-moz` et `-webkit`.

```
.box {  
    -moz-border-radius: 1em;  
    -webkit-border-radius: 1em;  
    border-radius: 1em;  
}
```

Les éditeurs de navigateurs expérimentent toujours de nouvelles extensions pour les CSS. Certaines proviennent de versions CSS qui ne sont pas encore implémentées, alors que d'autres ne se fraient un chemin dans les spécifications que plus tardivement. Certaines encore peuvent ne jamais atteindre la spécification officielle, comme celles utilisées par Safari pour afficher des éléments d'interface utilisateur sur l'iPhone.

Pour ne pas perturber d'autres agents utilisateur ou invalider votre code, vous pouvez invoquer ces extensions en ajoutant un préfixe propre à l'éditeur dans votre sélecteur, votre propriété ou votre valeur. Par exemple, Mozilla utilise le préfixe `-moz`, alors que Safari utilise le préfixe `-webkit`. Il existe des préfixes similaires pour Internet Explorer, Opera et tous les principaux navigateurs. Chaque navigateur possède son propre jeu de fonctionnalités spéciales auquel vous pouvez accéder avec ces préfixes. Vous devrez donc probablement vérifier lesquels sont disponibles sur le site de développement des éditeurs.

Grâce à ce mécanisme, vous pouvez tester de nouvelles fonctionnalités CSS 3 avant qu'elles ne fassent partie de la recommandation officielle. Cependant, utilisez ces extensions prudemment, car le format de ces fonctionnalités expérimentales peut différer d'un navigateur à l'autre et peut même changer ou disparaître d'ici à ce que la spécification devienne une recommandation officielle.

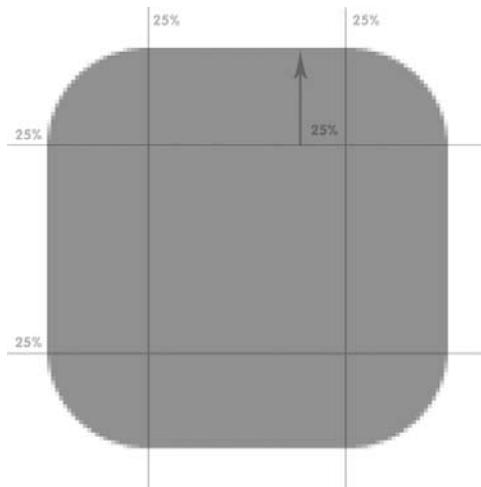
border-image

Voici le dernier élément de ma liste de nouvelles astuces CSS 3 : la propriété `border-image`. Cette excellente nouveauté CSS permet d'indiquer une unique image comme bordure d'un élément. À quoi bon utiliser une seule image, direz-vous ? C'est là toute la magie de cette propriété : elle permet de découper l'image en neuf secteurs distincts, selon de simples règles de pourcentage, en laissant le soin au navigateur d'utiliser le secteur approprié pour la partie correspondante de la bordure. Cette technique de "mise à l'échelle en neuf découpes" (en anglais, *nine-slice scaling*) permet d'éviter l'effet de distorsion qu'on obtient normalement en redimensionnant des boîtes à bords arrondis. Pour mieux le comprendre, prenons un exemple.

Vous disposez d'une image de 100 pixels de haut représentant une boîte à bords arrondis comme celle de la Figure 4.14. Si vous tracez deux lignes à 25 % en partant du haut et du bas de la boîte, puis deux autres à 25 % de la gauche et de la droite, vous aurez divisé votre boîte en neuf secteurs.

Figure 4.14

Le fichier source pour notre image de bordure avec, pour l'illustration, ses lignes de division.



La propriété `border-image` utilise automatiquement les images de chaque secteur pour créer l'arrondi ou la bordure correspondants. L'image du secteur supérieur gauche est ainsi utilisée pour créer l'arrondi supérieur gauche et l'image du secteur droit central pour créer la bordure latérale droite. Je veux que les bordures fassent 25 pixels de large : c'est la largeur que je définis donc dans les CSS. Si les images ne sont pas suffisamment grandes, elles sont automatiquement répétées, de manière à créer une boîte extensible (voir Figure 4.15). Voici comment procéder dans le code :

```
.box {  
  -webkit-border-image: url(/img/corners.gif)  
  25% 25% 25% 25% / 25px round round;  
}
```

Safari prend en charge cette propriété depuis un certain temps, avec l'extension `-webkit`, comme le montre cet exemple. Firefox 3.5 et Opera 9.5 la reconnaissent maintenant aussi, ce qui ouvre considérablement son champ d'application.

Figure 4.15

Une boîte à bords arrondis créée avec la propriété *border-image*.



Ombres portées

Les ombres portées sont des artifices visuels populaires et attrayants. Elles donnent de la profondeur et rehaussent l'apparence des pages trop plates. La plupart des utilisateurs se servent de logiciels comme Photoshop pour les ajouter directement aux images. Mais avec les CSS, il est possible d'appliquer des effets simples d'ombre portée sans altérer les images d'origine.

Plusieurs raisons peuvent pousser à procéder de cette manière, par exemple, pour que des utilisateurs qui ne savent pas retoucher des images puissent administrer votre site web ou qu'ils puissent le faire depuis un lieu où ils n'ont pas accès à Photoshop, comme un cyber café. À l'aide d'un style d'ombre portée prédefini, vous pourrez télécharger de simples images et les afficher sur le site avec une ombre portée.

L'un des plus intéressants avantages liés à l'utilisation des CSS tient à ce que cette technique ne dénature pas les images. Si vous décidez de supprimer l'effet d'ombre portée par la suite, vous n'aurez que deux lignes à modifier dans vos fichiers CSS au lieu de retraiter la totalité des images.

Ombres portées simples en CSS

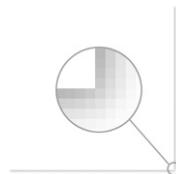
Cette méthode très simple a été décrite la première fois par Dunstan Orchard (www.1976design.com). Elle consiste à appliquer une grande image d'ombre portée à l'arrière-plan d'une div conteneur. L'ombre portée est ensuite révélée par décalage de l'image à l'aide de marges négatives.

La première chose à faire est de créer l'image de l'ombre portée. C'est ce que j'ai fait avec Photoshop. Créez un fichier Photoshop, dont les dimensions doivent atteindre la taille maximale de l'image. J'ai, pour ma part, créé un fichier de 800 pixels sur 800 pixels, par précaution. Déverrouillez le calque d'arrière-plan et remplissez-le avec la couleur sur laquelle vous souhaitez projeter l'ombre portée. Pour cet exemple, j'ai conservé l'arrière-plan en blanc. Créez un nouveau calque et remplissez-le en blanc. Ensuite, décalez ce calque vers le haut

et la gauche d'environ 4 ou 5 pixels puis appliquez une ombre portée de la même taille à ce calque. Enregistrez l'image pour le Web et appelez-la shadow.gif (voir Figure 4.16).

Figure 4.16

L'image shadow.gif agrandie pour révéler l'ombre portée de 5 pixels.



Le code pour cette technique est très simple :

```
<div class="img-wrapper"></div>
```

Pour créer l'effet, vous devez commencer par appliquer l'image d'ombre à l'arrière-plan de la div d'enveloppement. Comme les div sont des éléments de niveau bloc, elles s'étendent en largeur jusqu'à occuper tout l'espace disponible. Dans le cas présent, nous souhaitons que la div enveloppe l'image. On pourrait, à cette fin, définir une largeur explicite pour la div, mais l'utilité de la technique s'en trouverait réduite. Au lieu de cela, vous pouvez la faire flotter, afin qu'elle se rétrécisse pour envelopper l'image dans les navigateurs modernes, à l'exception d'Internet Explorer 5.x sur Mac. Vous pouvez éventuellement masquer ces styles pour cette version de navigateur. Pour plus d'informations sur le masquage des règles selon les différents navigateurs, consultez le Chapitre 8, qui traite des hacks et des filtres.

```
.img-wrapper {
background: url(/img/shadow.gif) no-repeat bottom right;
clear: right;
float: left;
}
```

Pour révéler l'image de l'ombre et créer l'effet d'ombre portée (voir Figure 4.17), vous devez décaler l'image en spécifiant des marges négatives :

```
.img-wrapper img {
margin: -5px 5px 5px -5px;
}
```

Figure 4.17

L'image avec son ombre portée.



Vous pouvez aussi créer un bel effet de papier photo en donnant à l'image une bordure et un remplissage (voir Figure 4.18) :

```
.img-wrapper img {  
    background-color: #fff;  
    border: 1px solid #a9a9a9;  
    padding: 4px;  
    margin: -5px 5px 5px -5px;  
}
```

Figure 4.18

Le résultat final de notre simple technique d'ombre portée.



Cette technique fonctionne pour la plupart des navigateurs modernes. Il faut cependant ajouter quelques règles simples pour qu'elle fonctionne avec Internet Explorer 6 :

```
.img-wrapper {  
    background: url(/img/shadow.gif) no-repeat bottom right;  
    clear: right;  
    float: left;  
    position: relative;  
}  
  
.img-wrapper img {  
    background-color: #fff;  
    border: 1px solid #a9a9a9;  
    padding: 4px;  
    display: block;  
    margin: -5px 5px 5px -5px;  
    position: relative;  
}
```

L'effet d'ombre portée fonctionne maintenant aussi dans Internet Explorer 6.

Ombres portées "à la Clagnut"

Richard Rutter (www.clagnut.com) a proposé une méthode similaire pour créer des ombres portées. Au lieu d'utiliser des marges négatives, il préfère un positionnement relatif pour décaler l'image :

```
.img-wrapper {
  background: url(/img/shadow.gif) no-repeat bottom right;
  float:left;
  line-height:0;
}

.img-wrapper img {
  background:#fff;
  padding:4px;
  border:1px solid #a9a9a9;
  position:relative;
  left:-5px;
  top:-5px;
}
```

box-shadow

Les précédentes techniques sont utiles, mais elles sont tout de même quelque peu fastidieuses. Ne serait-il pas intéressant que les navigateurs puissent créer leurs propres ombres portées, sans avoir besoin de filtres Photoshop ou d'images vectorielles ? Vous l'aurez deviné : la CSS 3 le permet, avec la propriété `box-shadow`. Cette propriété prend quatre valeurs : les décalages vertical et horizontal, la largeur de l'ombre (et donc son niveau de flou) et sa couleur. Dans l'exemple suivant, j'ai décalé l'ombre de 3 pixels, en lui donnant 6 pixels de largeur et une couleur gris clair (voir Figure 4.19).

```
img {
  box-shadow: 3px 3px 6px #666;
}
```

Figure 4.19

Une ravissante ombre portée avec l'effet `box-shadow`.



Comme il s'agit d'une propriété CSS 3 encore expérimentale, vous devez utiliser les extensions Safari et Firefox pour l'instant. Il ne faudra cependant sans doute pas attendre bien longtemps pour qu'elle soit plus largement prise en charge.

```
img {
  -webkit-box-shadow: 3px 3px 6px #666;
  -moz-box-shadow: 3px 3px 6px #666;
  box-shadow: 3px 3px 6px #666;
}
```

Cette propriété est particulièrement intéressante parce qu'elle fonctionne en conjonction avec la propriété `border-radius` (voir Figure 4.20). Vous pouvez donc maintenant créer

des ombres portées de manière programmatique sur les boîtes à bords arrondis, sans même avoir à ouvrir votre logiciel d'édition d'image !

Figure 4.20

Une ombre portée sur une boîte à bords arrondis.



Nous avons copieusement utilisé cet effet dans le site web UX London 2009, en proposant des ombres portées aux navigateurs modernes (voir Figure 4.21) et des boîtes standard pour les navigateurs moins avancés (voir Figure 4.22).



Figure 4.21

Le site web UX London tel que l'affiche Firefox. Observez les ombres portées unies créées avec des CSS 3.



Figure 4.22

Le site web UX London affiché cette fois dans Internet Explorer. Les ombres ne s'affichent plus, mais la plupart des utilisateurs ne remarqueront pas ce qui manque.

Opacité

S'il est bien utilisé, l'effet d'opacité peut donner à vos mises en page une autre dimension. Il peut aussi permettre de positionner des éléments les uns au-dessus des autres en livrant des indices sur ce qui se cache dessous. Au-delà du simple gadget, ces petits détails peuvent améliorer l'utilisabilité du site.

Opacité CSS

L'opacité CSS est disponible depuis un certain temps dans la plupart des navigateurs modernes. On peine du coup à comprendre pourquoi elle n'est pas utilisée plus souvent. On ne sera toutefois pas surpris d'apprendre qu'elle n'est pas prise en charge par les anciennes versions d'Internet Explorer. Un petit ajout de code propre à Internet Explorer suffit néanmoins à corriger le problème. À titre d'exemple, imaginons que vous souhaitez faire apparaître un message d'alerte à vos utilisateurs, en le superposant au document existant afin qu'ils puissent voir ce qui se passe dessous (voir Figure 4.23).

```
.alert {
  background-color: #000;
  border-radius: 2em;
  opacity: 0.8;
  filter: alpha(opacity=80); /*proprietary IE code*/
}
```

Figure 4.23

Boîte d'alerte à bords arrondis à 80 % d'opacité.



Le principal problème avec l'opacité CSS tient à ce que le contenu de l'élément auquel vous l'appliquez en hérite, comme l'arrière-plan. Si vous examinez la Figure 4.23, vous verrez ainsi que le texte dans la page s'affiche à travers la boîte d'alerte également. Ce n'est pas un problème si vous utilisez une très forte opacité avec un texte contrasté. Mais avec une opacité faible, le contenu de la boîte peut commencer à devenir illisible. C'est là qu'intervient le RGBA.

RGBa

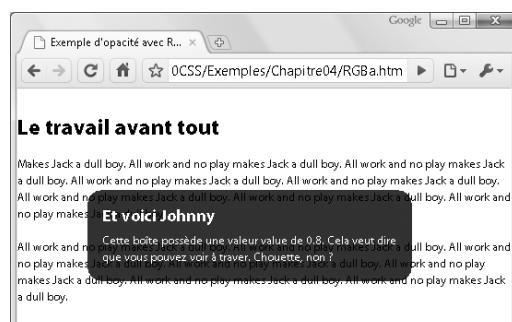
RGBa est un mécanisme permettant de définir la couleur et l'opacité en même temps. RGB est l'acronyme de *Red, Green, Blue* (rouge, vert, bleu), "a" signifie "alpha transparency" (transparence alpha). Pour utiliser RGBa dans l'exemple précédent, vous procéderiez de la manière suivante :

```
.alert {  
  background-color: rgba(0,0,0,0.8);  
  border-radius: 2em;  
}
```

Les trois premiers chiffres représentent les valeurs de rouge, de vert et de bleu de la couleur. Ici, la boîte d'alerte est noire, aussi toutes les valeurs sont-elles égales à 0. Le dernier chiffre définit la valeur décimale de l'opacité. 0.8 correspond à un arrière-plan opaque à 80 % (ou transparent à 20 %) [voir Figure 4.24]. La valeur 1 rend la boîte d'alerte opaque à 100 %. La valeur 0 la rend complètement transparente.

Figure 4.24

Boîte d'alerte à bords arrondis opaque à 80 % en RGBa.



Transparence PNG

L'un des plus grands atouts du format de fichier PNG concerne sa prise en charge de la transparence alpha. Il permet de produire des maquettes particulièrement créatives (voir Figure 4.25). Malheureusement, Internet Explorer 6 ne prend pas en charge la transparence PNG de manière native, au contraire d'Internet Explorer 7 et 8. Il existe cependant quelques astuces pour ramener les anciennes versions d'Internet Explorer dans le giron.

Figure 4.25

L'ancien site

Revver incluait un magnifique exemple de transparence PNG en bas de la fenêtre d'affichage. Lorsque la page défilait, des portions du contenu transparaissaient sous les branches de l'arbre et sous l'arc-en-ciel.



La méthode la plus connue pour forcer la prise en charge de la transparence PNG dans Internet Explorer 6 consiste à utiliser le filtre propriétaire `AlphaImageLoader`. Pour cela, vous devez inclure la ligne de code suivante dans le fichier CSS :

```
filter:progid:DXImageTransform.Microsoft.AlphaImageLoader
(src='/img/my-image.png', sizingMethod='crop');
```

Malheureusement, ce code invalide le fichier CSS. Il est donc préférable de le filtrer dans une feuille de styles séparée spécifique d'Internet Explorer 6.

```
.img-wrapper div {
  filter:progid:DXImageTransform.Microsoft.AlphaImageLoader
  (src='/img/shadow2.png', sizingMethod='crop');
  background: none;
}
```

La première règle utilise un filtre propriétaire pour charger le PNG et forcer la transparence alpha. L'image d'arrière-plan d'origine reste affichée, si bien que la seconde règle vient simplement la masquer.

Internet Explorer reconnaît un autre type de code propriétaire appelé *commentaire conditionnel*, qui permet de fournir une feuille de styles particulière aux versions spécifiques d'Internet Explorer. Dans le cas présent, vous souhaitez qu'Internet Explorer 6 seul voie la feuille de styles. Vous pouvez donc placer le code suivant en haut de la page :

```
<!--[if ie 6]>
<link rel="stylesheet" type="text/css" href="ie6.css" />
<![endif]-->
```

Ne vous préoccupez pas pour l'instant des commentaires conditionnels. Vous en apprendrez plus à ce sujet au Chapitre 8.

Le problème avec cette technique tient à ce qu'elle force à inclure cette ligne de code pour chaque image PNG transparente que vous souhaitez utiliser. Elle est donc peu commode.

L'autre approche consiste à utiliser une technique appelée "IE PNG fix" (correctif pour les PNG sous Internet Explorer). Elle se sert d'une extension CSS peu connue et propre à Microsoft, nommée "comportements". En téléchargeant le fichier .htc approprié et en le faisant pointer sur votre feuille de styles spécifique d'Internet Explorer 6, vous pouvez activer la transparence PNG sur l'élément de votre choix.

```
img, div {
    behavior: url(iepngfix.htc);
}
```

Pour plus d'informations sur cette technique et pour télécharger le fichier .htc, consultez la page www.twinhelix.com/css/iepngfix.

Effet parallaxe CSS

Les images d'arrière-plan ne servent pas qu'à créer des bords arrondis et des ombres portées. On peut s'amuser avec. C'est ce que nous avons fait chez Clearleft, lors du lancement de notre application de test d'utilisabilité Silverback. Si vous accédez au site www.silverbackapp.com et redimensionnez la taille de la fenêtre, vous remarquerez un étrange effet (voir Figure 4.26). Les images d'arrière-plan se déplacent à des vitesses différentes, ce qui donne une sensation de profondeur à la page. Il s'agit d'un *défilement parallaxe*. Ce mécanisme a constitué l'épine dorsale de bien des jeux vidéo aux premières heures des ordinateurs personnels.

Figure 4.26

Modifiez la taille de la fenêtre sur le site www.silverbackapp.com et observez ce qui se passe.



Pour réaliser cet effet, vous devez d'abord créer plusieurs images d'arrière-plan. Ici, nous avons créé une image de feuillage sur un arrière-plan vert, puis deux images de feuillage plus lointaines sur un arrière-plan transparent. Les images du premier plan et du plan intermédiaire peuvent ainsi être superposées l'une à l'autre ainsi qu'à l'arrière-plan, sans en masquer la vue.

Le principal arrière-plan est appliqué à l'élément `body`. Si vous n'utilisez pas d'images d'arrière-plan multiples en CSS 3, vous devez cependant ajouter deux nouveaux éléments auxquels lier ces arrière-plans. Le contenu de la page doit ensuite venir se placer devant ces éléments afin d'interagir avec. Vous pourriez placer la div de premier plan devant le contenu, mais vous en bloqueriez partiellement le contenu et il deviendrait plus difficile d'interagir avec. La structure du code doit donc ressembler à ceci :

```
<body>
  <div class="midground">
    <div class="foreground">
      <p>Votre contenu vient ici !</a>
    </div>
  </div>
</body>
```

La première chose à faire est d'ajouter l'arrière-plan principal à l'élément `body`. Il faut que cette image se répète à l'horizontale. Vous devrez donc attribuer la valeur `repeat-x` à la propriété `image-repeat`. Il faut aussi que l'élément `body` prenne la couleur de l'arrière-plan, qui se trouve être vert clair. Enfin, il faut décaler l'image horizontalement de 20 % par rapport à la taille de la fenêtre. C'est là que la magie fait son effet. Lorsque la fenêtre se redimensionne, l'image d'arrière-plan change de position et elle semble se déplacer dans l'écran.

```
body {
  background-image: url(/img/bg-rear.jpg);
  background-repeat: repeat-x;
  background-color:#d3ff99;
  background-position: 20% 0;
}
```

Vous devez maintenant en faire de même avec les images de plan intermédiaire et de premier plan, en choisissant des pourcentages plus élevés pour qu'elles se déplacent plus rapidement et suscitent ainsi un effet de profondeur. Nous avons décidé de fixer la position du plan intermédiaire à 40 % et celle du premier plan, chicement, à 150 % (voir Figure 4.27). Vous pouvez jouer sur ces valeurs pour générer l'effet qui vous convient.

```
body {
  background-image: url(/img/bg-rear.jpg);
  background-repeat: repeat-x;
  background-color:#d3ff99;
  background-position: 20% 0;
}

.midground {
  background-image: url(/img/bg-mid.png);
  background-repeat: repeat-x;
  background-color: transparent;
```

```
background-position: 40% 0;  
}  
  
.foreground {  
background-image: url(/img/bg-front.png);  
background-repeat: repeat-x;  
background-color: transparent;  
background-position: 150% 0;  
}
```

Figure 4.27

Quand la taille de la fenêtre change, les feuilles à l'arrière-plan se déplacent à différentes vitesses et créent un effet de profondeur.



Remplacement d'images

Le texte HTML, c'est super. Les moteurs de recherche peuvent le lire, vous pouvez le copier et le coller, il s'agrandit quand vous augmentez la taille du texte du navigateur. Il est donc judicieux de l'utiliser à la place des images chaque fois que c'est possible. Malheureusement, les concepteurs web ne disposent que d'un ensemble très limité de polices pour travailler. En outre, s'il est possible de contrôler la typographie dans une certaine mesure à l'aide des CSS, plusieurs contraintes sont indépassables avec du texte pur. Il arrive ainsi parfois, pour respecter la charte graphique d'une marque, qu'il faille utiliser des images de texte à la place.

Au lieu d'incorporer ces images directement dans la page, les auteurs CSS ont développé une technique de remplacement des images. Elle consiste à ajouter le texte au document

normalement, puis à utiliser des CSS pour le masquer et afficher une image d'arrière-plan à la place. Les moteurs de recherche peuvent ainsi trouver le texte HTML, qui reste disponible quand vous désactivez les CSS.

L'idée a semblé géniale pendant un temps, jusqu'à ce que des failles apparaissent. Certaines des méthodes les plus populaires sont inaccessibles aux lecteurs d'écran et la plupart ne fonctionnent pas lorsque les images sont désactivées et les CSS activées. Du coup, un grand nombre d'auteurs ont cessé d'utiliser les méthodes de remplacement des images et sont repassés au texte pur. Si j'invite à éviter le remplacement des images quand c'est possible, je continue de croire qu'il existe des cas où il est judicieux, comme lorsque vous devez utiliser une police particulière pour respecter des directives graphiques liées à la charte d'une entreprise ou d'une marque. Pour cela, vous devez connaître les différentes techniques disponibles et comprendre leurs limites respectives.

Remplacement des images de Fahrner

Créé par Todd Fahrner, le FIR (*Fahrner Image Replacement* ou remplacement des images de Fahrner) est une technique originale de remplacement des images qui est, sans doute aussi, la plus populaire. Je la présente ici pour des raisons historiques et parce qu'il s'agit d'une des méthodes les plus faciles à comprendre. Elle possède cependant de sérieux inconvénients en matière d'accessibilité, comme nous le verrons dans un instant, et doit donc être évitée.

Le principe de base est très simple. Vous enveloppez le texte à remplacer dans une balise `span` :

```
<h2>
  <span>Hello World</span>
</h2>
```

Ensuite, vous appliquez l'image de remplacement comme image d'arrière-plan à l'élément de titre :

```
h2 {
  background:url(hello_world.gif) no-repeat;
  width: 150px;
  height: 35px;
}
```

Vous masquez enfin le contenu de l'élément `span` en attribuant la valeur `none` à sa propriété `display` :

```
span {
  display: none;
}
```

Cette méthode fonctionne comme un charme, mais c'est cette dernière règle qui pose problème. La plupart des lecteurs d'écran populaires ignorent les éléments dont la propriété `display` vaut `none` ou la propriété `visibility` vaut `hidden`. Ils ignorent du coup complètement le texte, ce qui pose un considérable problème d'accessibilité. Cette technique destinée à améliorer l'accessibilité du site a donc un effet exactement inverse. Il vaut mieux dès lors l'éviter.

Phark

Mike Rundle (www.phark.net) a, pour sa part, inventé une technique de remplacement d'images, adaptée aux lecteurs d'écran et qui possède, en outre, l'avantage d'éviter la div supplémentaire non sémantique :

```
<h2>
  Hello World
</h2>
```

Au lieu d'utiliser la propriété `display` pour masquer le texte, Phark applique un retrait négatif très élevé au texte du titre :

```
h2 {
  text-indent: -5000px;
  background:url(/img/hello_world.gif) no-repeat;
  width: 150px;
  height:35px;
}
```

Cette méthode fonctionne bien et résout le problème des lecteurs d'écran. Mais comme la méthode FIR, elle ne fonctionne pas quand les images sont désactivées et les CSS activées. Il s'agit là d'un cas limite, qui ne s'applique certainement qu'aux personnes dotées de connexions à bas débit ou qui utilisent leur téléphone portable comme moyen de connexion à Internet. On peut arguer que les visiteurs du site ont la possibilité d'activer les images et que ce sont eux qui décident de ne pas le faire. Considérez, cependant, que certaines personnes ne verront pas le texte remplacé et qu'il est préférable d'éviter cette méthode pour les informations cruciales ou la navigation.

Méthode IFR évolutive

L'un des principaux problèmes que le remplacement des images tente de résoudre est celui du manque de polices disponibles sur la plupart des ordinateurs. Au lieu de remplacer le texte par des images de texte, Mike Davidson et Shaun Inman ont adopté une autre technique, plus inventive.

Flash permettant d'incorporer des polices dans un fichier SWF, ils ont donc décidé de remplacer le texte par un fichier Flash. La substitution s'opère en JavaScript : le document est parcouru et tout le texte d'un élément particulier ou possédant un nom de classe particulier est récupéré. Le code JavaScript le remplace ensuite par un petit fichier Flash. La partie véritablement ingénieuse vient après. Au lieu de créer un fichier Flash distinct pour chaque portion de texte, cette technique place le texte remplacé dans un unique fichier dupliqué. Tout ce que vous avez à faire pour déclencher le remplacement consiste à ajouter une classe. La combinaison du Flash et du JavaScript se charge du reste. L'autre avantage tient à ce que le texte dans le fichier Flash peut être sélectionné, ce qui signifie qu'il peut être facilement copié et collé.

Shaun Inman a inauguré cette méthode très légère de remplacement par des images Flash et l'a nommée IFR (*Inman Flash Replacement*). Pour plus de détails à son sujet, et notamment pour en télécharger le code source, consultez la page www.shauninman.com/plete/2004/04/ifr-revisited-and-revised.

Mike Davidson s'est ensuite appuyé dessus pour créer la méthode sIFR (*Scalable Inman Flash Replacement* ou méthode IFR évolutive), qui étend l'IFR en autorisant, par exemple, le redimensionnement du texte ou le remplacement des textes multilignes. Elle est maintenant gérée et développée par Mark Wubben et inclut de nouvelles fonctionnalités intéressantes.

Pour utiliser la méthode sIFR dans votre site, vous devez d'abord en télécharger la dernière version à l'adresse <http://novemberborn.net/sifr3>. sIFR s'installe très facilement, mais nous conseillons de commencer par lire la documentation. La première chose à faire est d'ouvrir le fichier Flash, d'incorporer la police à utiliser et d'exporter l'animation. Vous devez ensuite appliquer les styles d'impression et d'écran inclus ou créer les vôtres, puis ajouter le fichier JavaScript sifr.js à chaque page dans laquelle vous souhaitez que sIFR fonctionne. Ce fichier est parfaitement configurable et permet de spécifier les éléments à remplacer, la couleur du texte, le remplissage, la casse et une variété d'autres paramètres de style. Le remplissage et la hauteur de ligne affectent cependant la taille du texte, ce qui ne simplifie pas les choses. Une fois que vous avez terminé, placez tous les fichiers sur votre serveur et observez vos vieilles polices déprimantes remplacées par le contenu Flash dynamique !

Le principal problème avec ces techniques tient au temps de téléchargement. Les pages doivent entièrement se charger avant que le code JavaScript ne puisse procéder au remplacement du texte. Bien souvent, on observe du coup un léger clignotement avant que le texte n'ait été remplacé par l'équivalent Flash (voir Figure 4.28).

Figure 4.28

Vous remarquerez que les titres du site www.fortymedia.com ne s'affichent qu'une fois que la page est chargée. C'est l'indice que la méthode sIFR est utilisée dans ce site.



Si ce clignotement n'a pas de grande conséquence, il se remarque pourtant et peut donner l'impression que la page se charge lentement. Certaines pages, aussi, semblent lentes lorsque de nombreux remplacements sont opérés. En outre, les utilisateurs peuvent apercevoir subrepticement du contenu non mis en forme, ce qui peut faire craindre un bogue ou engendrer une certaine confusion. Mieux vaut donc limiter cette technique aux seuls titres. La méthode sIFR est excellente pour intégrer une typographie riche et parfaite pour les sites relativement petits qui utilisent une typographie uniforme. Elle est, en revanche, moins commode pour les grands sites où figurent de nombreux titres de différentes tailles, de différents

styles et de différentes couleurs. Elle devient aussi compliquée lorsque certains titres couvrent plusieurs lignes ou possèdent des couleurs d'arrière-plan. Je conseille donc de l'éviter pour les grands projets et de ne l'utiliser que pour des sites personnels.

En résumé

Vous avez découvert de quelle manière les images d'arrière-plan pouvaient être appliquées à des éléments pour produire une variété d'effets comme des boîtes à bords arrondis et des ombres portées CSS. Vous avez également vu comment les nouvelles propriétés CSS 3 telles que `border-radius` et `box-shadow` rendent ces effets inutiles. Vous avez découvert l'opacité et vu comment forcer la prise en charge de la transparence PNG dans Internet Explorer. Vous vous êtes enfin familiarisé avec quelques méthodes de remplacement des images.

Au chapitre suivant, vous verrez comment les images d'arrière-plan et les liens peuvent être combinés pour créer d'intéressants effets interactifs.

5

Mise en forme des liens

Le modeste lien d'ancre est ni plus ni moins que la pierre angulaire du Web. Il s'agit du mécanisme qui permet aux pages web de se connecter les unes aux autres et aux personnes d'explorer et de parcourir des sites. La mise en forme par défaut des liens est assez peu flamboyante, mais avec une pincée de CSS, il est possible de réaliser d'incroyables transformations.

Au cours de ce chapitre, vous apprendrez à :

- ordonner les sélecteurs de liens d'après la cascade ;
- créer des soulignements stylés ;
- mettre en forme des liens externes avec des sélecteurs d'attribut ;
- amener des liens à se comporter comme des boutons ;
- créer des styles de liens visités ;
- créer des info-bulles CSS.

Mise en forme simple des liens

Le moyen le plus simple de mettre en forme un lien consiste à utiliser le sélecteur de type d'ancre. Par exemple, la règle suivante passe toutes les ancre en rouge :

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

Les ancre peuvent cependant agir en guise de références internes en plus de liens externes, si bien qu'il n'est pas toujours idéal d'utiliser un sélecteur de type. Considérez l'exemple suivant. La première ancre contient un identificateur de fragment. Lorsque l'utilisateur clique dessus, il saute directement à la seconde ancre nommée :

```
<p><a href="#contenuPrincipal">Sauter au contenu principal</a></p>
...
<h1><a name="contenuPrincipal">Bienvenue</a></h1>
```

Dans ce cas, alors qu'il serait préférable que seul le lien soit en rouge, le titre va le devenir aussi. Pour éviter cela, les CSS possèdent deux sélecteurs spéciaux appelés *sélecteurs de pseudo-classe de lien* : `:link`, utilisé pour cibler les liens qui n'ont pas été visités et `:visited`, pour cibler les liens visités. Dans cet exemple, tous les liens non visités passent en bleu et tous les liens visités en vert :

```
a:link {color: blue;}      /* Passe les liens non visités en bleu */
a:visited {color: green;} /* Passe les liens visités en vert */
```

Deux autres sélecteurs peuvent être utilisés pour la mise en forme des liens : les sélecteurs de pseudo-classe dynamique `:hover` et `:active`. Le premier, `:hover`, est utilisé pour cibler

les éléments pendant leur survol, et le second, `:active`, cible les éléments quand ils sont activés. Dans le cas des liens, l'activation intervient lorsque l'utilisateur clique dessus. Dans cet exemple, les liens passent en rouge lorsque l'utilisateur les survole ou clique dessus :

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

Pour vous assurer que vos pages sont accessibles du mieux possible, vous avez tout intérêt à ajouter une pseudo-classe `:focus` à vos liens lorsque vous définissez vos états de survol. Elle permet aux liens de récupérer les mêmes styles et, cela, qu'on y accède avec la touche Tab ou qu'on les survole avec le curseur.

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

D'autres éléments peuvent aussi utiliser les sélecteurs de pseudo-classe `:hover`, `:active` ou `:focus`. Par exemple, vous pourriez ajouter une pseudo-classe `:hover` aux lignes de tableau, une pseudo-classe `:active` aux boutons d'envoi ou une pseudo-classe `:focus` aux champs de saisie afin de signaler différentes formes d'interactivité. Sachez que, malheureusement, à la différence des navigateurs modernes, Internet Explorer 7 et ses versions antérieures ne prennent en charge les sélecteurs de pseudo-classe que sur les liens.

```
/* passe les lignes de tableau en jaune lors du survol */
tr:hover {
    background: yellow;
}

/* affiche les boutons d'envoi en jaune quand on clique dessus dans certains
   ➔navigateurs */
input[type="submit"]:active {
    background: yellow;
}
/* passe les champs de saisie en jaune lorsqu'ils sont sélectionnés */
input:focus {
    background: yellow;
}
```

La plupart des développeurs utilisent ces sélecteurs d'abord pour désactiver le soulignement des liens, puis pour le réactiver en cas de survol ou de clic. Il suffit pour cela d'attribuer la valeur `none` à la propriété `text-decoration` des liens visités et non visités, puis la valeur `underline` à celle des liens actifs ou survolés :

```
a:link, a:visited {text-decoration: none;}
a:hover, a:focus, a:active {text-decoration: underline;}
```

L'ordre des sélecteurs est très important dans cet exemple. S'il est inversé, les styles de survol et de lien actif ne fonctionnent pas.

```
a:hover, a:focus, a:active {text-decoration: underline;}
a:link, a:visited {text-decoration: none;}
```

Tout vient du système de cascade. Au Chapitre 1, vous avez appris que lorsque deux règles possèdent la même spécificité, c'est la dernière à être définie qui prend le dessus. Dans le cas présent, les deux règles possèdent la même spécificité. Les styles `:link` et `:visited`

viennent donc redéfinir les styles :hover et :active. Pour éviter cela, il est judicieux d'appliquer les styles de liens dans l'ordre suivant :

```
a:link, a:visited, a:hover, a:focus, a:active
```

En anglais, une petite formule a été retenue pour se remémorer cet ordre : "Lord Vader Hates Furry Animals" (Darth Vader n'aime pas les bêtes à poil). Si vous n'aimez pas l'anglais, creusez-vous la tête et trouvez une formule pour L-V-H-F-A !

Soulignements amusants

Du point de vue de l'utilisabilité et de l'accessibilité, il est important que vos liens puissent se signaler par d'autres moyens que leur seule couleur. Trop de malvoyants ont du mal à faire la distinction entre les couleurs de faible contraste, notamment sur le texte de petite taille. Par exemple, ceux qui ne distinguent pas les couleurs ne peuvent pas différencier certaines combinaisons de couleurs dont les niveaux de luminosité ou de saturation sont trop proches. C'est pour cette raison que les liens sont soulignés par défaut.

Embellissement simple des liens

Les concepteurs n'aiment généralement pas les liens soulignés, qui alourdissent la page et l'encombrent. Si vous décidez de supprimer les soulignements, vous pouvez passer les liens en gras. La page a l'air moins chargée, mais les liens sont clairement visibles :

```
a:link, a:visited {  
    text-decoration: none;  
    font-weight: bold;  
}
```

Vous pouvez ensuite réappliquer les soulignements en cas de survol ou de sélection des liens, afin de renforcer leur caractère interactif :

```
a:hover, a:focus, a:active {  
    text-decoration: underline;  
    font-weight: bold;  
}
```

Il est aussi possible de créer un soulignement discret avec des bordures. Dans cet exemple, le soulignement par défaut est remplacé par une ligne en pointillé moins imposante. Lorsque l'utilisateur survole le lien ou clique dessus, la ligne devient continue afin d'indiquer visuellement que quelque chose vient de se passer :

```
a:link, a:visited {  
    text-decoration: none;  
    border-bottom: 1px dotted #000;  
}  
  
a:hover, a:focus, a:active {  
    border-bottom-style: solid;  
}
```

Soulignement de liens originaux

Vous pouvez créer de très intéressants effets en utilisant des images pour créer les soulignements de liens. Par exemple, j'ai créé une image de soulignement très simple constituée de lignes diagonales (voir Figure 5.1).

Figure 5.1

Image de soulignement simple.



Vous pouvez ensuite appliquer cette image aux liens à l'aide du code suivant :

```
a:link, a:visited {
  color:#666;
  text-decoration: none;
  background: url(/img/underline1.gif) repeat-x left bottom;
}
```

Le résultat mis en forme est présenté à la Figure 5.2.

Figure 5.2

Soulignement de lien personnalisé.



Vous n'êtes pas limité aux styles de liens ou de liens visités. Dans l'exemple suivant, j'ai créé une image GIF animée pour les états de survol et de sélection que j'applique avec ce code CSS :

```
a:hover, a:focus, a:active {
  background-image: url(/img/underline1-hover.gif);
}
```

Lorsque vous survolez le lien ou que vous cliquez dessus, les lignes diagonales semblent défiler de gauche à droite, ce qui crée un intéressant effet de tournoiement. Tous les navigateurs ne prennent pas en charge les animations d'image, mais ceux qui ne le font pas affichent généralement la première image de l'animation, si bien que l'effet rétrograde également sous la forme d'un style plus simple dans les anciens navigateurs.

Utilisez les animations avec prudence, car elles peuvent générer des problèmes d'accessibilité pour certains. En cas de doute, vérifiez toujours les directives pour l'accessibilité aux contenus web (WCAG 1.0) à l'adresse www.w3.org/TR/WAI-WEBCONTENT.

Style de liens visités

Les concepteurs et les développeurs oublient souvent le style des liens visités auxquels ils finissent par donner le même aspect qu'aux liens non visités. Le choix d'un style de lien visité différent peut cependant faciliter l'orientation des utilisateurs en signalant les pages ou les sites déjà visités afin d'éviter des marches arrière inutiles.

Vous pouvez en créer un, très simple, en ajoutant une case à cocher à chaque lien visité :

```
a:visited {
    padding-right: 20px;
    background: url(/img/check.gif) no-repeat right middle;
}
```

Mise en forme des cibles des liens

En plus de créer des liens vers des documents spécifiques, vous pouvez aussi en créer avec un identificateur de fragment qui renvoie vers une partie précise d'une page. Pour cela, vous devez ajouter un caractère dièse suivi par l'ID de l'élément vers lequel le lien doit pointer, à la fin de l'attribut `href`. C'est très utile si vous souhaitez pointer vers un commentaire en particulier dans un fil de commentaires assez long. Par exemple, supposons que vous souhaitez faire un lien vers le troisième commentaire de cette page :

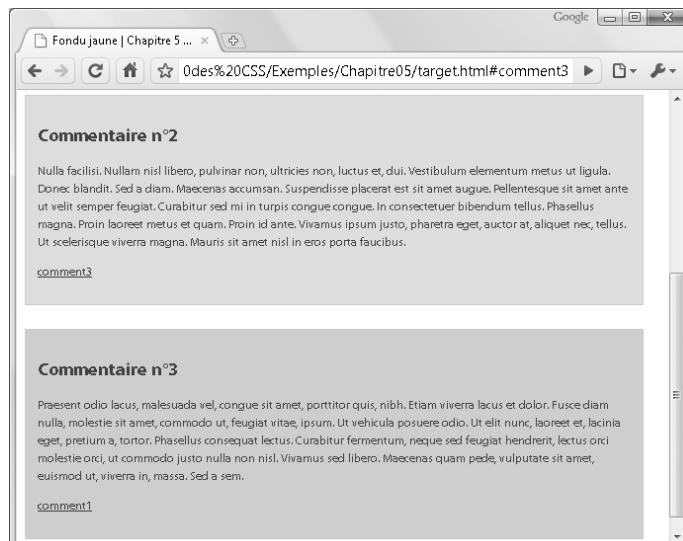
```
<a href="http://example.com/story.htm#comment3">
    A great comment by Simon
</a>
```

Lorsque vous cliquez sur ce lien, vous êtes conduit au document approprié et la page défile directement jusqu'à l'élément `comment3`. Malheureusement, si la page est assez dense, c'est souvent difficile de savoir quel élément a été ciblé par le lien. Pour résoudre ce problème, la spécification CSS 3 permet de mettre en forme l'élément cible à l'aide de la pseudo-classe `:target`. Dans l'exemple suivant, je signale l'élément cible en lui donnant un arrière-plan jaune (voir Figure 5.3).

```
.comment:target {
    background-color: yellow;
}
```

Figure 5.3

Le troisième commentaire est signalé avec un arrière-plan jaune lorsqu'un lien pointe vers lui, grâce au sélecteur `:target`.



Si vous souhaitez faire mieux encore, vous pouvez choisir d'attribuer à l'élément une image d'arrière-plan animée qui s'estompe en passant du jaune au blanc, afin de simuler la technique de fondu jaune popularisée par les sites comme 37 Signals.

```
.comment:target {  
    background-image: url(img/fade.gif);  
}
```

Le sélecteur de cible est pris en charge par toutes les versions récentes de Safari et de Firefox, mais il n'est pas reconnu par Internet Explorer à ce jour.

Différenciation des types de liens

Dans de nombreux sites, il est difficile de dire si un lien pointe vers une autre page ou vers un autre site. Combien de fois vous est-il arrivé de cliquer sur un lien en pensant atterrir sur une autre page du site pour vous retrouver soudain transporté dans un autre univers ? Pour résoudre ce problème, de nombreux sites ouvrent les liens externes dans une nouvelle fenêtre. Cette solution n'est cependant pas idéale, car elle dépossède l'utilisateur du contrôle de la navigation et peut accumuler les fenêtres les unes sur les autres. Elle peut aussi poser des problèmes aux utilisateurs de lecteurs d'écran si la nouvelle fenêtre n'est pas annoncée. Les nouvelles fenêtres interrompent aussi la continuité de la navigation avec le bouton de page précédente, car il est impossible de revenir en arrière vers le précédent écran.

L'une des solutions possibles consiste à signaler les liens externes d'une manière ou d'une autre et à laisser le choix à l'utilisateur de quitter le site ou non, d'ouvrir le lien dans une nouvelle fenêtre ou, comme il est plus habituel de le faire de nos jours, dans un nouvel onglet. Vous pouvez ajouter une petite icône à côté de tous les liens externes. Certains sites, comme www.wikipedia.com, le font déjà et une convention qui illustre les liens externes avec l'icône d'une boîte surmontée d'une flèche semble émerger (voir Figure 5.4).

Figure 5.4

Icône de lien externe.



Le moyen le plus simple d'inclure une icône de lien externe dans les pages consiste à ajouter une classe à tous les liens externes et à appliquer l'icône comme image d'arrière-plan. Dans l'exemple suivant, j'ai créé un espace pour l'icône en donnant au lien un léger remplissage droit, puis en appliquant l'icône comme image d'arrière-plan en haut à droite du lien (voir Figure 5.5).

```
.external {  
    background: url(/img/externalLink.gif) no-repeat right top;  
    padding-right: 10px;  
}
```

Figure 5.5

Mise en forme des liens externes.

Voici un lien externe 

Cette méthode fonctionne, mais elle n'est pas particulièrement intelligente ni élégante, car vous devez ajouter manuellement cette classe à chaque lien externe. Ne serait-il pas intéressant que les CSS puissent déterminer à votre place ce qui est ou n'est pas un lien externe ? En fait, c'est parfaitement possible, grâce aux sélecteurs d'attribut.

Comme vous l'avez appris au Chapitre 1, les sélecteurs d'attribut permettent de cibler un élément en fonction de l'existence ou de la valeur d'un attribut. La spécification CSS 3 étend cette capacité à la mise en correspondance de sous-chaînes. Avec ces sélecteurs, on peut cibler un élément en établissant la correspondance entre un fragment de texte choisi et une partie de la valeur d'un attribut. La CSS 3 n'est pas encore une spécification officielle, aussi l'utilisation de ces sélecteurs avancés peut-elle invalider votre code. Cependant, la majorité des navigateurs compatibles avec les standards les prennent en charge.

Cette technique fonctionne en ciblant d'abord tous les liens qui commencent par le texte `http:` avec le sélecteur d'attribut `[att^=val]` :

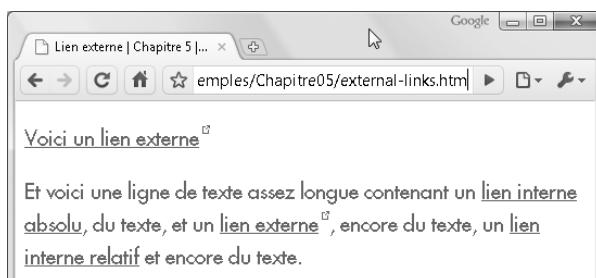
```
a[href^="http:"] {
    background: url(/img/externalLink.gif) no-repeat right top;
    padding-right: 10px;
}
```

Ce code doit mettre en valeur tous les liens externes. Il sélectionne cependant aussi tous les liens internes qui utilisent des URL absolues et non relatives. Pour éviter cela, vous devez redéfinir tous les liens vers votre site ainsi : supprimez l'icône de lien externe, retrouvez les liens qui pointent vers votre nom de domaine, supprimez l'icône de lien externe et redéfinissez le remplissage droit (voir Figure 5.6).

```
a[href^="http://www.yoursite.com"],
a[href^="http://yoursite.com"] {
    background-image: none;
    padding-right: 0;
}
```

Figure 5.6

Cette page affiche les liens externes différemment des liens internes.



La plupart des navigateurs modernes prennent en charge cette technique, mais les plus anciens, comme Internet Explorer 6 et ses versions antérieures, l'ignorent tout simplement.

Si vous le souhaitez, vous pouvez étendre cette méthode en distinguant aussi les liens de messagerie. Dans l'exemple suivant, j'ajoute une petite icône d'e-mail à tous les liens `mailto` :

```
a[href^="mailto:"] {
    background: url(img/email.png) no-repeat right top;
    padding-right: 10px;
}
```

Vous pouvez même signaler les protocoles non standard comme le protocole de messagerie instantanée AOL (AIM), avec une petite icône de personnage AIM (voir Figure 5.7) :

```
a[href^="aim:"] {
    background: url(img/im.png) no-repeat right top;
    padding-right: 10px;
}

<a href="aim:goim?screenname=andybudd">instant message</a>
```

Figure 5.7

Styles de liens d'e-mail et de messagerie instantanée.



Signalement de documents téléchargeables et de flux

Quelle frustration que de cliquer sur un lien pour accéder à une autre page et de découvrir que le site lance le téléchargement d'un PDF ou d'un document Word ! Par chance, les CSS peuvent aider à distinguer ces types de liens également. Le sélecteur d'attribut `[att$=val]` peut cibler les attributs qui se terminent par une valeur particulière, comme `.pdf` ou `.doc` :

```
a[href$=".pdf"] {
    background: url(img/pdfLink.gif) no-repeat right top;
    padding-right: 10px;
}

a[href$=".doc"] {
    background: url(img/wordLink.gif) no-repeat right top;
    padding-right: 10px;
}
```

Comme pour les précédents exemples, vous pouvez donc signaler les liens vers les documents Word ou PDF avec leur propre icône, afin de prévenir les utilisateurs qu'ils cliquent sur un document téléchargeable au lieu d'un lien vers une autre page.

De nombreux sites proposent des flux RSS avec des liens que les utilisateurs doivent copier dans leurs lecteurs de flux. Malheureusement, ceux qui cliquent par inadvertance sur les liens de ce type se trouvent conduits vers une page de données apparemment insensée. Pour éviter cette surprise, vous pouvez signaler les flux RSS à l'aide d'une méthode similaire, en affichant votre propre icône RSS :

```
a[href$=".rss"], a[href$=".rdf"] {  
    background: url(img/feedLink.gif) no-repeat right top;  
    padding-right: 10px;  
}
```

Toutes ces techniques peuvent contribuer à améliorer le confort d'usage de votre site. En signalant les liens externes ou les documents téléchargeables, vous indiquez aux utilisateurs à quoi ils doivent s'attendre en cliquant sur un lien. Vous leur évitez donc d'être frustrés et d'avoir à revenir sur leurs pas.

Internet Explorer 6 et ses versions antérieures ne prennent malheureusement pas en charge le sélecteur d'attribut. Vous pouvez cependant créer un effet semblable en ajoutant une classe à chaque élément grâce à du JavaScript et au DOM. L'excellente fonction `getElementsBySelector` de Simon Willison en est l'un des meilleurs exemples. Pour plus d'informations à ce sujet, consultez la page (en anglais) <http://simonwillison.net/2003/Mar/25/getElementsByTagNameBySelector/>. jQuery permet aussi de réaliser une opération très similaire.

Création de liens qui ressemblent à des boutons

Les ancrées sont des éléments incorporés. Cela signifie qu'elles ne s'activent que lorsque vous cliquez sur le contenu du lien. Pourtant, il peut arriver que vous souhaitiez créer des mécanismes plus proches des boutons, avec des zones réactives plus larges. Pour cela, vous devez attribuer la valeur `block` à la propriété `display` et modifier la largeur, la hauteur et les autres propriétés de l'élément pour créer le style et la zone réactive désirés.

```
a {  
    display: block;  
    width: 6.6em;  
    line-height: 1.4;  
    text-align: center;  
    text-decoration: none;  
    border: 1px solid #66a300;  
    background-color: #8cca12;  
    color: #fff;  
}
```

Le lien résultant est présenté à la Figure 5.8.

Figure 5.8

Lien mis en forme à la manière d'un bouton.

[Book Now »](#)

À présent, le lien s'affiche comme un élément de niveau bloc dans lequel vous pouvez cliquer n'importe où pour activer le lien.

Si vous examinez le code CSS, vous verrez que la largeur a été explicitement définie en cadratins (em). Par nature, les éléments de niveau bloc s'étendent sur toute la largeur disponible. Si la largeur de leur élément parent dépasse celle requise pour le lien, il faut attribuer la largeur désirée au lien. Il le faudrait si vous souhaitiez utiliser un lien de ce type dans

la zone de contenu principal de votre page, mais si vos liens se trouvent dans une colonne latérale, vous pouvez sans doute vous contenter de définir la largeur de la colonne sans vous préoccuper de la largeur des liens.

Vous vous demandez peut-être pourquoi j'utilise `line-height` pour contrôler la hauteur du bouton au lieu de la propriété `height`. Il s'agit, en fait, d'une astuce pour centrer verticalement le texte dans le bouton. Si vous définissez la propriété `height`, vous aurez certainement besoin d'utiliser un remplissage pour "pousser" le texte vers le bas et simuler un centrage vertical. Le texte est en revanche toujours centré verticalement dans une boîte incorporée, si bien qu'en modulant `line-height`, on s'assure qu'il figure toujours au milieu de la boîte. Il y a un inconvénient, toutefois. Si le texte du bouton s'étend sur deux lignes, il devient deux fois plus haut qu'on ne le souhaite. Le seul moyen d'éviter cela est de redimensionner les boutons et le texte de manière que ce dernier ne soit pas renvoyé à la ligne ou, à tout le moins, ne le fasse pas tant que le texte conserve une longueur raisonnable.

Si vous choisissez cette technique, assurez-vous de ne l'utiliser que sur des éléments de liens simples et non sur des éléments qui enclenchent des mises à jour du serveur. Sans cela, vous risqueriez de produire quelques effets indésirables. Lorsque l'accélérateur de Google a été lancé, certains utilisateurs ont constaté que le contenu dans leurs applications web et de leurs applications de gestion de contenu disparaissait mystérieusement. Parfois, le contenu entier d'un site pouvait se volatiliser pendant la nuit. Les auteurs de ces outils avaient en fait utilisé des liens d'ancre au lieu d'éléments de formulaire pour leurs boutons de suppression. L'accélérateur Google suivait ces liens afin de les mettre en cache et supprimait involontairement le contenu ! Les robots des moteurs de recherche peuvent avoir le même effet, en supprimant de manière récursive de vastes pans de données. Voilà pourquoi vous ne devez jamais utiliser de liens pour opérer des changements sur le serveur. Ou pour le dire en termes techniques, les liens ne doivent être utilisés que pour des requêtes GET et jamais pour des requêtes POST.

Survol simple

À l'époque, quand on vivait à la dure, il fallait programmer de grandes et complexes fonctions JavaScript pour créer des effets de survol. La pseudo-classe `:hover` permet maintenant de s'épargner ces efforts. On peut ainsi étendre l'exemple précédent afin d'inclure un effet de survol simple en définissant la couleur d'arrière-plan et la couleur du texte du lien survolé (voir Figure 5.9) :

```
a:hover,  
a:focus {  
    background-color: #f7a300;  
    border-color: #ff7400;  
}
```

Figure 5.9

Style de survol signalant une zone réactive.

Book Now »

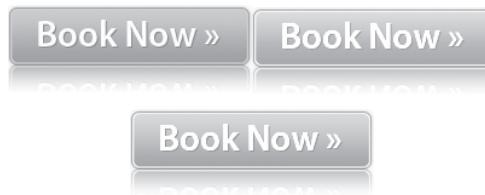


Survol avec images

Le changement des couleurs d'arrière-plan fonctionne bien pour les boutons simples, mais, pour des boutons plus sophistiqués, vous chercherez sans doute à utiliser des images d'arrière-plan. Dans l'exemple suivant, j'ai créé trois images de bouton : une pour l'état par défaut, une autre pour l'état de survol et le focus et une dernière pour l'état actif (voir Figure 5.10).

Figure 5.10

Les images pour l'état par défaut, le survol et l'état actif du bouton.



Le code de cet exemple ressemble à celui de l'exemple précédent. La principale différence tient à l'utilisation des images d'arrière-plan au lieu des bordures et des couleurs d'arrière-plan.

```
a:link, a:visited {  
    display: block;  
    width: 203px;  
    height: 72px;  
    text-indent: -1000em;  
    background: url(/img/button.png) left top no-repeat;  
}  
  
a:hover, a:focus { background-image: url(/img/button-over.png);  
}  
  
a:active {  
    background-image: url(/img/button-active.png);  
}
```

Cet exemple utilise des boutons de largeur et de hauteur fixes. J'ai donc défini des dimensions explicites en pixels dans le code CSS. Pour obtenir le traitement précis que je souhaitais pour le texte, j'ai inclus le texte du bouton sur l'image puis sorti le texte d'ancre de l'écran à l'aide un retrait négatif très grand. Rien n'empêche cependant de créer des images de bouton de grand format ou d'utiliser une combinaison de couleurs d'arrière-plan et d'images pour créer un bouton fluide ou élastique.

Survol style Pixy

Le principal inconvénient avec la méthode des images multiples tient au léger délai qu'elle génère quand les navigateurs chargent l'image de survol la première fois. Ce délai peut provoquer un effet de clignotement indésirable et donner l'impression que les boutons réagissent mal. On peut précharger les images de survol en les appliquant comme arrière-plan à l'élément parent. Mais il existe un autre moyen. Au lieu de permutez plusieurs images d'arrière-plan, utilisez une seule image et décalez sa position d'arrière-plan. Le recours à une unique image a l'avantage de réduire le nombre de requêtes serveur en plus de permettre de conserver tous les états de bouton au même endroit. Cette méthode est appelée "méthode Pixy", du surnom de son créateur, Petr Staníček (pour plus d'informations à ce sujet, consultez son site web : <http://wellstyled.com/css-nopreload-rollovers.html>).

Commencez par créer une image de bouton combiné (voir Figure 5.11). Dans le cas présent, je limite le bouton à un état relevé, un état de survol et un état actif. Vous pouvez cependant aussi inclure un état visité si vous le souhaitez.

Figure 5.11

Les trois états de bouton sous forme d'image unique.



Le code est presque identique à l'exemple précédent, mais cette fois, vous alignez l'image d'arrière-plan de façon que l'état normal se trouve au centre, puis vous décalez l'arrière-plan vers la droite ou vers la gauche pour les états de survol et actif.

```
a:link, a:visited {
    display: block;
    width: 203px;
    height: 72px;
    text-indent: -1000em;
    background: url(/img/buttons.png) -203px 0 no-repeat;
}

a:hover, a:focus {
    background-position: right top;
}

a:active {
    background-position: left top;
}
```

Internet Explorer s'entête malheureusement à faire un nouvel aller-retour vers le serveur pour requérir une nouvelle image alors que vous ne faites que modifier l'alignement de la même image. Cette action provoque un léger clignotement, qui peut être agaçant. Pour l'éviter, appliquez l'état de survol, non pas au lien lui-même, mais à son élément parent, par exemple au paragraphe qui le contient :

```
p {
    background: url(/img/ buttons.png) no-repeat right top;
}
```

L'image disparaît aussi un instant pendant son chargement, mais cette fois, c'est la même image qui apparaît en dessous, ce qui permet de masquer l'effet de clignotement.

L'autre astuce pour supprimer le clignotement consiste à inclure la ligne de code suivante dans le fichier CSS spécifique d'Internet Explorer, qui active la mise en cache de l'arrière-plan.

```
html {  
    filter: expression(document.execCommand("BackgroundImageCache", false, true));  
}
```

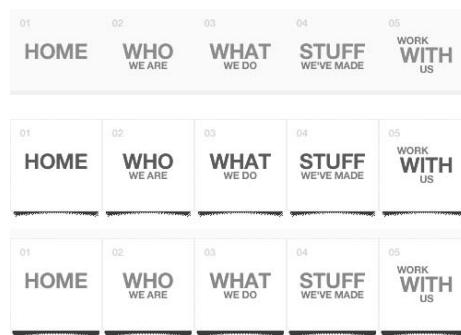
Sprites CSS

Les requêtes multiples au serveur peuvent avoir un effet désastreux sur les performances des sites. La méthode Pixy vise donc à réduire le nombre de requêtes en incluant tous les états du bouton dans une unique image. Mais pourquoi s'arrêter en si bon chemin ? Vous pouvez réduire le nombre d'appels au serveur à deux ou trois. C'est exactement ce que font les sprites CSS – une unique image contenant une multitude d'icônes, de boutons et d'autres images. De nombreux sites web importants utilisent cette technique, dont la page d'accueil de Yahoo!, ou bien encore notre site Clearleft, pour sa barre de navigation (voir Figure 5.12).

```
#nav li a {  
    display: block;  
    text-indent: -9999px;  
    height: 119px;  
    width: 100px;  
    background-image: url('/img/nav.png');  
    background-repeat: no-repeat;  
}  
  
#nav li.home a,  
#nav li.home a:link,  
#nav li.home a:visited {  
    background-position: 0 0;  
}  
  
#nav li.home a:hover,  
#nav li.home a:focus,  
#nav li.home a:active {  
    background-position: 0 -119px;  
}  
  
#nav li.who-we-are a,  
#nav li.who-we-are a:link,  
#nav li.who-we-are a:visited {  
    background-position: -100px 0;  
}  
#nav li.who-we-are a:hover,  
#nav li.who-we-are a:focus,  
#nav li.who-we-are a:active {  
    background-position: -100px -119px;  
}
```

Figure 5.12

Le fichier de sprites CSS utilisé sur le site Clearleft.



Cette technique permet de réduire le nombre de requêtes transmises par le navigateur web au serveur et d'accélérer considérablement les temps de téléchargement. En outre, avec les sprites, les boutons, les icônes et les diverses images sont conservés dans un même emplacement, afin de faciliter la gestion. On y gagne donc sur tous les plans.

Survol CSS 3

La spécification CSS 3 inclut un certain nombre de propriétés comme `text-shadow`, `box-shadow` et `border-radius` pour créer des boutons sophistiqués sans utiliser la moindre image. Pour créer ce genre de bouton, commencez par le code de notre premier exemple :

```
a {
  display: block;
  width: 6.6em;
  height: 1.4em;
  line-height: 1.4;
  text-align: center;
  text-decoration: none;
  border: 1px solid #66a300;
  background-color: #8cca12;
  color: #fff;
}
```

Ensuite, ajoutez des bords arrondis et une ombre portée. Vous pouvez aussi attribuer au texte du bouton une légère ombre portée (voir Figure 5.13).

```
a {
  display: block;
  width: 6.6em;
  height: 1.4em;
  line-height: 1.4;
  text-align: center;
  text-decoration: none;
  border: 1px solid #66a300;
  -moz-border-radius: 6px;
  -webkit-border-radius: 6px;
  border-radius: 6px;
  background-color: #8cca12;
  color: #fff;
  text-shadow: 2px 2px 2px #66a300;
```

```
-moz-box-shadow: 2px 2px 2px #ccc;
-webkit-box-shadow: 2px 2px 2px #ccc;
box-shadow: 2px 2px 2px #ccc;
}
```

Figure 5.13

Un bouton à bords arrondis réalisé entièrement en CSS.

[Book Now »](#)

Pour recréer le dégradé, Safari 4 bêta prend en charge une valeur propriétaire appelée `-webkit-gradient`. En temps normal, je ne conseille jamais d'utiliser du code propriétaire, mais ce cas est révélateur de la direction que pourraient emprunter les CSS à l'avenir. Cette valeur propriétaire prend un certain nombre d'arguments, dont le type de dégradé (linéaire ou radial), son orientation (ici, du haut à gauche vers le bas à droite), une couleur de départ, une couleur finale et d'éventuelles étapes intermédiaires. Évidemment, si vous ne souhaitez pas utiliser ce code propriétaire, vous pouvez réaliser votre propre image de dégradé à la place.

```
a {
  display: block;
  width: 6.6em;
  height: 1.4em;
  line-height: 1.4;
  text-align: center;
  text-decoration: none;
  border: 1px solid #66a300;
  -moz-border-radius: 6px;
  -webkit-border-radius: 6px;
  border-radius: 6px;
  background-image: -webkit-gradient(linear, left top, left bottom, É
from(#abe142), to(#67a400));
  background-color: #8cca12;
  color: #fff;
  text-shadow: 2px 2px 2px #66a300;
  -moz-box-shadow: 2px 2px 2px #ccc;
  -webkit-box-shadow: 2px 2px 2px #ccc;
  box-shadow: 2px 2px 2px #ccc;
}
```

Pour finir, Safari inclut une autre propriété propriétaire, appelée `box-reflect`, qui permet de créer des reflets d'objets. Elle contient plusieurs arguments, dont la position et la distance du reflet ainsi qu'une image de masque. Détail intéressant, vous pouvez utiliser la valeur `-webkit-gradient` pour générer ce masque de manière automatique. Ici, je positionne le reflet à 2 pixels en dessous du bouton et j'utilise un masque de fondu vers le blanc (voir Figure 5.14).

```
a {
  display: block;
  width: 6.6em;
```

```

height: 1.4em;
line-height: 1.4;
text-align: center;
text-decoration: none;
border: 1px solid #66a300;
-moz-border-radius: 6px;
-webkit-border-radius: 6px;
border-radius: 6px;
background-image: -webkit-gradient(linear, left top, left bottom, É
from(#abe142), to(#67a400));
background-color: #8cca12;
color: #fff;
text-shadow: 2px 2px 2px #66a300;
-moz-box-shadow: 2px 2px 2px #ccc;
-webkit-box-shadow: 2px 2px 2px #ccc;
box-shadow: 2px 2px 2px #ccc;
-webkit-box-reflect: below 2px -webkit-gradient (linear, left top, left bottom,
from(transparent), color-stop(0.52, transparent), to(white));
}

```

Figure 5.14

Ce bouton à bords arrondis a été réalisé avec l'extension CSS de Safari.

[Book Now »](#)

Un débat s'est engagé pour savoir si ces types d'effets devaient ou non être réalisés en CSS. Il n'est donc pas certain que ces propriétés finissent par trouver place dans la spécification officielle. Comme ces techniques sont en outre mal prises en charge par les navigateurs, il n'est pas judicieux de les utiliser dans un environnement de production. Cela ne doit cependant pas vous empêcher de vous amuser dans vos sites personnels si vous comprenez bien qu'il s'agit de code CSS invalide qui pourra être supprimé ou modifié dans les prochaines versions du navigateur.

Info-bulles CSS

Les info-bulles sont ces petits encarts de texte jaunes qui s'affichent dans certains navigateurs lorsque vous survolez des éléments qui possèdent une balise de titre. Plusieurs développeurs ont créé leurs propres info-bulles en combinant du JavaScript et des CSS. Il est cependant possible de créer des info-bulles entièrement en CSS à l'aide des techniques de positionnement. Pour bien fonctionner, cette technique requiert un navigateur moderne compatible avec les standards, comme Firefox. Elle n'est pas à inclure dans l'arsenal des outils quotidiens, mais elle montre combien les CSS avancées sont efficaces et fait imaginer ce qui sera possible quand la prise en charge s'améliorera.

Comme avec tous les exemples de ce livre, vous devez commencer par du code HTML bien structuré et sémantiquement cohérent :

```

<p>
<a href="http://www.andybudd.com/" class="tooltip">

```

```
Andy Budd(ce site web est génial) est un développeur web  
résidant à Brighton en Angleterre.  
</p>
```

J'ai attribué au lien la classe `tooltip` afin de le différencier des autres liens. À l'intérieur, j'ai ajouté le texte que je souhaitais afficher comme texte du lien, suivi du texte de l'info-bulle entouré d'une balise `span`. J'ai mis le texte d'info-bulle entre parenthèses afin que la phrase conserve son sens même si les styles sont désactivés.

La première chose à faire est de définir la propriété `position` de l'ancre en lui attribuant la valeur `relative`. Vous pouvez ainsi positionner le contenu de la balise `span` de manière absolue, relativement à la position de son ancre parente.

Le texte de l'info-bulle ne doit pas s'afficher au départ. Vous devez donc attribuer la valeur `none` à sa propriété `display` :

```
a.tooltip {  
  position: relative;  
}  
  
a.tooltip span {  
  display: none;  
}
```

Le contenu de la balise `span` doit s'afficher quand l'utilisateur survole l'ancre. Pour cela, vous devez attribuer la valeur `block` à la propriété `display` de la balise `span`, mais uniquement lorsque l'utilisateur survole le lien. Pour le moment, si vous testez le code et survolez le lien, vous verrez le texte de la balise `span` à côté du lien.

Pour positionner le contenu de la balise `span` en dessous et à droite de l'ancre, vous devez attribuer la valeur `absolute` à sa propriété `position` et la positionner à 1 cadratin (em) du haut de l'ancre et 2 cadratins de la gauche.

```
a.tooltip:hover span {  
  display: block;  
  position: absolute;  
  top: 1em;  
  left: 2em;  
}
```

Rappelez-vous qu'un élément positionné de manière absolue l'est en fonction de son ancêtre positionné le plus proche ou, en son absence, de l'élément racine. Dans cet exemple, nous avons positionné l'ancre, si bien que la balise `span` l'est en fonction de celle-ci.

C'est le principe fondamental de cette technique. Il ne reste plus qu'à ajouter une mise en forme pour faire ressembler la balise `span` à une véritable info-bulle. Vous pouvez définir un remplissage, une bordure et une couleur d'arrière-plan :

```
a.tooltip:hover span, a.tooltip:focus span {  
  display: block;  
  position: absolute;  
  top: 1em;
```

```
left:2em;  
padding: 0.2em 0.6em;  
  
border:1px solid #996633;  
background-color:#FFFF66;  
color:#000;  
}
```

Dans Firefox, le résultat doit ressembler à celui de la Figure 5.15.

Figure 5.15

Une info-bulle
entièrement réalisée
en CSS.

Andy Budd est un développeur web résidant à Brighton en Angleterre.
▼ (ce site est génial)

En résumé

Vous avez appris à mettre en forme des liens de plusieurs manières et vous savez maintenant le faire en fonction du site ou du fichier vers lequel ils pointent, transformer des liens en boutons et créer des effets de survol en utilisant des couleurs ou des images. Vous pouvez même créer des effets avancés comme des info-bulles entièrement en CSS.

Au chapitre suivant, vous allez apprendre à manipuler des listes. Grâce aux acquis de ce chapitre, vous créerez des listes de navigation, des cartes-images et des survols distants entièrement en CSS. Amusez-vous !

6

Mise en forme des listes et création de barres de navigation

L'homme a par nature tendance à organiser le monde qui l'entoure. Les scientifiques créent des listes d'animaux, de plantes et d'éléments chimiques. Les magazines créent des listes présentant les meilleurs films, les dernières tendances de la mode et les célébrités les plus mal habillées. Les gens font des listes de courses, des listes de choses à faire et écrivent des listes au Père Noël. Tout le monde adore faire des listes.

Les listes permettent de regrouper des éléments apparentés et, ce faisant, de leur donner une signification et une structure. La plupart des pages web en contiennent, qu'il s'agisse de listes des derniers articles, de listes de liens vers des pages web favorites ou de listes de liens vers d'autres parties du site. Le fait d'identifier ces éléments sous forme de liste et de les marquer de la sorte peut aider à structurer les documents HTML et donc fournir des ancrages utiles pour appliquer les styles.

Au cours de ce chapitre, vous apprendrez à :

- mettre en forme des listes avec des CSS ;
- utiliser des images d'arrière-plan en guise de puces ;
- créer des barres de navigation verticale et horizontale ;
- créer des systèmes de navigation élastiques à onglets ;
- créer des menus déroulants entièrement en CSS ;
- créer des cartes-images CSS ;
- créer des survols distants ;
- utiliser des listes de définitions.

Mise en forme de base des listes

La mise en forme de base des listes est très simple. Supposons que vous commençiez par cette simple liste de tâches à faire :

```
<ul>
  <li>Lire les e-mails</li>
  <li>Écrire un chapitre</li>
  <li>Faire les courses</li>
  <li>Faire à manger</li>
  <li>Regarder Lost</li>
</ul>
```

Pour ajouter une puce personnalisée, vous pouvez utiliser la propriété `list-style-image`. Vous n'aurez cependant pas grand contrôle sur la position de la puce. Préférez plutôt désactiver les puces des listes et ajouter les vôtres sous la forme d'images d'arrière-plan.

Vous pourrez ensuite utiliser les propriétés de positionnement de l'image d'arrière-plan pour contrôler précisément l'alignement des puces.

Les versions plus anciennes d'Internet Explorer et d'Opera contrôlent le retrait des listes avec la marge gauche, alors que la plupart des navigateurs modernes, dont Firefox et Safari, utilisent le remplissage gauche. À cet égard, la première chose à faire est de supprimer ce retrait en ramenant à zéro la marge et le remplissage de la liste. Pour supprimer la puce par défaut, choisissez `none` (aucun) pour le type de style de la liste :

```
ul {
  margin: 0;
  padding: 0;
  list-style-type: none;
}
```

Rien de plus simple que d'ajouter une puce personnalisée. Le remplissage du côté gauche de l'élément de liste crée l'espace nécessaire pour la puce, laquelle est ensuite appliquée sous forme d'image d'arrière-plan à l'élément de liste. Si celui-ci doit couvrir plusieurs lignes, il faudra peut-être positionner la puce vers le haut ou près du haut de l'élément de liste. Si vous savez, au contraire, que le contenu des éléments de la liste ne s'étendra pas sur plus d'une ligne, vous pouvez centrer verticalement la puce en définissant la position verticale avec la valeur `middle` ou `50%` :

```
li {
  background: url(/img/bullet.gif) no-repeat 0 50%;
  padding-left: 30px;
}
```

La liste résultante est présentée à la Figure 6.1.

Figure 6.1

Une liste simple mise en forme avec des puces personnalisées.

- ✓ Lire les e-mails
- ✓ Écrire un chapitre
- ✓ Faire les courses
- ✓ Faire à manger
- ✓ Regarder Lost

Création d'une barre de navigation verticale

En combinant l'exemple précédent avec les techniques de mise en forme apprises au Chapitre 5, vous pouvez créer des barres de navigation verticales sophistiquées avec des survols CSS, comme celle présentée à la Figure 6.2.

Figure 6.2

Barre de navigation verticale mise en forme.



Comme toujours, vous devez commencer par un balisage sémantique de qualité :

```
<ul class="nav">
  <li><a href="home.htm">Accueil</a></li>
  <li><a href="about.htm">A propos</a></li>
  <li><a href="services.htm">Nos services</a></li>
  <li><a href="work.htm">Notre travail</a></li>
  <li><a href="news.htm">Actualités</a></li>
  <li><a href="contact.htm">Contact</a></li>
</ul>
```

La première chose à faire est de supprimer les puces par défaut et de ramener à zéro la marge et le remplissage :

```
ul.nav {
  margin: 0;
  padding: 0;
  list-style-type: none;
}
```

Vous pouvez ensuite commencer à disposer la mise en forme graphique. Ici, nous donnons au menu de navigation un arrière-plan vert clair et une bordure vert sombre. Nous définissons aussi la largeur de la liste de navigation en cadratins (em).

```
ul.nav {
  margin: 0;
  padding: 0;
  list-style-type: none;
  width: 8em;
  background-color: #8BD400;
  border: 1px solid #486B02;
}
```

Au lieu d'appliquer la mise en forme aux éléments de la liste, on l'applique aux liens d'ancre à l'intérieur, grâce à quoi on obtient une meilleure compatibilité entre les navigateurs. Pour créer une zone réactive analogue à un bouton, vous devez attribuer la valeur `block` à la propriété `display` des ancrés. Les liens d'ancre s'étirent alors de manière à couvrir tout l'espace disponible, qui correspond ici à la largeur de la liste. Vous pouvez définir la largeur des ancrés explicitement, mais j'ai constaté que le code était plus facile à gérer quand on définissait la largeur de la liste parente. Les deux dernières règles sont simplement des ajouts de mise en forme correspondant à la couleur du texte du lien et à la désactivation des soulignements.

```
ul.nav a {
  display: block;
  color: #2B3F00;
  text-decoration: none;
}
```

Pour créer l'effet de biseau sur les éléments de menu, vous devez attribuer à la bordure supérieure une couleur plus claire que la couleur d'arrière-plan et à la bordure inférieure une couleur plus sombre. À ce stade, vous pouvez aussi insérer une image d'arrière-plan à utiliser comme icône.

```
ul.nav a {
  display: block;
  color: #2B3F00;
  text-decoration: none;
  border-top: 1px solid #E4FFD3;
  border-bottom: 1px solid #486B02;
  background: url(/img/arrow.gif) no-repeat 5% 50%;
  padding: 0.3em 1em;
}
```

Idéalement, j'aurais défini le positionnement de la flèche à 10 pixels du bord gauche de l'ancre. La spécification CSS n'autorise cependant pas le mélange des unités. J'ai donc utilisé un pourcentage à la place. En vérité, la plupart des navigateurs acceptent les unités multiples, et il me semble que c'est un des rares points où la spécification a tort.

Avec toutes les bordures empilées les unes sur les autres, vous remarquerez que la bordure inférieure sur le dernier lien double la bordure inférieure de la liste. Dans le cas présent, je m'en tiens au plus simple et je supprime la bordure inférieure de la liste. Si ce n'est pas possible, on peut ajouter une classe sur le premier ou le dernier élément de la liste pour supprimer directement la bordure. À l'avenir, vous pourrez aussi utiliser la pseudo-classe `:last-child`, mais sa prise en charge par les navigateurs est à cette heure encore limitée.

```
ul.nav .last a {
  border-bottom: 0;
}
```

La liste ressemble maintenant à une barre de navigation verticale de fière apparence. Pour compléter l'effet, il ne reste qu'à appliquer les états `:hover`, `:focus` et `:selected`. Pour cela, changez les couleurs de texte et d'arrière-plan. Vous pouvez aussi changer celles des bordures afin de créer un effet d'enfoncement du bouton. Ces styles sont appliqués aux liens d'ancre lorsque l'utilisateur les survole. Ils sont également appliqués à toutes les ancrées dont l'élément de liste parent reçoit la classe `selected`.

```
ul.nav a:hover,
ul.nav a:focus,
ul.nav .selected a {
  color: #E4FFD3;
  background-color: #6DA203;
}
```

Cette technique devrait maintenant fonctionner dans tous les principaux navigateurs à l'exception d'Internet Explorer 6 et de ses versions antérieures pour Windows. Internet Explorer 6 ajoute inexplicablement un espace supplémentaire au-dessus et en dessous des éléments de liste. Pour corriger ce bogue, vous devez définir la propriété `display` des éléments de liste en lui attribuant la valeur `inline` :

```
ul.nav li {
  display: inline; /* :KLUDGE: Supprime l'espace incongru dans IE/Win */
}
```

Et voilà : vous avez maintenant une barre de navigation verticale mise en forme, complète, avec ses effets de survol.

Signaler la page courante dans la barre de navigation

Dans le précédent exemple de barre de navigation verticale, j'ai utilisé une classe pour signaler la page courante. Pour les petits sites, où la navigation est incorporée dans la page, vous pouvez ajouter simplement la classe dans chaque page. Pour les grands sites, il y a toutes les chances que la navigation se construise de manière dynamique, auquel cas la classe peut être ajoutée à l'arrière-plan. Pour les sites de taille moyenne, où la navigation principale ne change pas, il est cependant courant d'inclure la navigation sous forme de fichier externe. Dans ces cas, ne serait-il pas intéressant de pouvoir signaler la page sur laquelle on se trouve sans avoir à ajouter dynamiquement une classe au menu ? C'est précisément ce que permettent les CSS.

Cette technique fonctionne par l'ajout d'un ID ou d'un nom de classe à l'élément body de chaque page, qui indique dans quelle page ou quelle section se trouve l'utilisateur. Ensuite, vous pouvez ajouter un ID ou un nom de classe correspondant à chaque élément de la liste de navigation. La combinaison unique de l'ID de la balise body et de la classe ou de l'ID de la liste peut être utilisée pour marquer la page ou la section courante dans la barre de navigation du site.

Considérez l'exemple de code HTML suivant. La page courante est la page d'accueil, comme l'indique l'ID home de la balise body. Chaque élément de liste dans la navigation principale se voit attribuer un nom de classe construit à partir du nom de la page à laquelle est lié l'élément de la liste.

```
<body id="home">
  <ul class="nav">
    <li class="home"><a href="home.htm">Accueil</a></li>
    <li class="about"><a href="about.htm">A propos</a></li>
    <li class="services"><a href="services.htm">Nos services</a></li>
    <li class="products"><a href="work.htm">Notre travail</a></li>
    <li class="news"><a href="news.htm">Actualités</a></li>
    <li class="contact"><a href="contact.htm">Contact</a></li>
  </ul>
</body>
```

Pour signaler la page courante, vous pouvez alors simplement cibler la combinaison suivante d'ID et de noms de classes :

```
#home .nav .home a,
#about .nav .about a ,
#news .nav .news a,
#products .nav .products a,
#services .nav .services a {
  background-position: right bottom;
  color: #fff;
  cursor: default;
}
```

Lorsque l'utilisateur se trouve sur la page d'accueil, l'élément de navigation possédant la classe home affiche l'état sélectionné, alors que dans la page "news", c'est l'élément de

navigation qui possède la classe `news` qui affiche l'état sélectionné. Pour compléter l'effet, j'ai changé le style de curseur pour afficher le curseur de flèche par défaut. Si on survole le lien sélectionné, le curseur ne change pas d'état : on n'est donc pas tenté de cliquer sur le lien vers la page où l'on se trouve déjà.

Création d'une barre de navigation horizontale simple

Imaginez que vous ayez une page de résultats de recherche et que vous souhaitez créer une liste de navigation paginée simple comme celle de la Figure 6.3. Pour cela, vous devez commencer par créer une liste numérotée de vos options de navigation.

```
<ol class="pagination">
  <li><a href="search.htm?page=1" rel="prev">Préc</a></li>
  <li><a href="search.htm?page=1">1</a></li>
  <li class="selected">2</li>
  <li><a href="search.htm?page=3">3</a></li>
  <li><a href="search.htm?page=4">4</a></li>
  <li><a href="search.htm?page=5">5</a></li>
  <li><a href="search.htm?page=3" rel="next">Suiv</a></li>
</ol>
```

Figure 6.3

La barre de navigation horizontale des résultats de recherche.



Vous remarquerez que j'ai utilisé l'attribut `rel` pour signaler les pages précédente et suivante du jeu de résultats. C'est un excellent moyen, qui va se révéler pratique par la suite, lorsque nous souhaiterons donner une autre apparence à ces liens en particulier.

Comme avec les autres exemples de liste du chapitre, vous devez commencer par supprimer les styles de marge, de remplissage et de liste par défaut du navigateur. Bien des développeurs – et j'en fais partie – préfèrent utiliser une réinitialisation globale au début de leur feuille de styles. Si vous le faites, sautez cette première étape.

```
ol.pagination {
  margin: 0;
  padding: 0;
  list-style-type: none;
}
```

Pour que les éléments de la liste s'alignent horizontalement et non verticalement, attribuez la valeur `inline` à leur propriété `display`. Pour la mise en forme des listes horizontales plus complexes, vous aurez cependant un meilleur contrôle si vous faites flotter les éléments puis si vous utilisez des marges pour les espacer les uns des autres.

```
ol.pagination li {
  float: left;
  margin-right: 0.6em;
}
```

Les éléments de la liste s'affichent maintenant tous horizontalement et vous pouvez commencer à leur appliquer un traitement graphique. Dans cet exemple, nous souhaitons que tous les numéros de page apparaissent dans un carré avec un arrière-plan gris. Lorsque l'utilisateur survole ces liens, l'arrière-plan doit passer en bleu et le texte du lien devenir blanc.

```
ol.pagination a,  
ol.pagination li.selected {  
    display: block;  
    padding: 0.2em 0.5em;  
    border: 1px solid #ccc;  
    text-decoration: none;  
}  
ol.pagination a:hover,  
ol.pagination a:focus,  
ol.pagination li.selected {  
    background-color: blue;  
    color: white;  
}
```

Voilà qui est parfait pour les numéros de page, mais nous voulons donner une mise en forme légèrement différente aux liens Précédent et Suivant. Pour cela, nous allons cibler leur attribut `rel` avec des sélecteurs d'attributs. Pour commencer, je ne souhaite pas que ces liens aient un effet de bordure. Je le désactive donc.

```
ol.pagination a[rel="prev"],  
ol.pagination a[rel="next"] {  
    border: none;  
}
```

Je voudrais aussi ajouter une flèche de présentation au début et à la fin de la liste. On peut les intégrer directement dans le code HTML, mais il est aussi possible de les injecter avec des CSS, afin de pouvoir les modifier ou les supprimer par la suite. Pour utiliser des CSS, servez-vous des pseudo-sélecteurs `:before` et `:after` en combinaison avec la propriété `content`.

```
ol.pagination a[rel="prev"]::before {  
    content: "\00AB";  
    padding-right: 0.5em;  
}  
  
ol.pagination a[rel="next"]::after {  
    content: "\00BB";  
    padding-left: 0.5em;  
}
```

La première déclaration cible le lien d'ancre au début de la liste et ajoute, avant lui, une double flèche vers la gauche avec le code de caractère "00AB". La seconde déclaration cible le dernier lien d'ancre et ajoute une double flèche droite à la fin.

Et voilà le résultat : une barre de navigation paginée horizontale simple mais flexible.

Création d'une barre de navigation graphique

Les barres de navigation simples sont excellentes pour le contenu paginé, mais vous souhaiterez sans doute créer des menus plus élaborés sur le plan graphique pour votre navigation principale. Dans cet exemple, vous allez voir comment créer une barre de navigation graphique comme celle de la Figure 6.4.



Figure 6.4

Barre de navigation horizontale.

Comme dans le précédent exemple, vous devez commencer par une liste à puces simple :

```
<ul class="nav">
  <li><a href="home.htm">Accueil</a></li>
  <li><a href="about.htm">A propos</a></li>
  <li><a href="news.htm">Actualités</a></li>
  <li><a href="products.htm">Produits</a></li>
  <li><a href="services.htm">Services</a></li>
  <li><a href="clients.htm">Clients</a></li>
  <li><a href="case-studies.htm">Etudes de cas</a></li>
</ul>
```

Ensuite, vous devez réduire à zéro le remplissage et les marges, ainsi que supprimer les puces par défaut. Pour cet exemple, il faut que la barre de navigation horizontale fasse 72 cadratins de large et possède un arrière-plan dégradé orange.

```
ul. nav {
  margin: 0;
  padding: 0;
  list-style: none;
  width: 72em;
  background: #FAA819 url(img/mainNavBg.gif) repeat-x;
}
```

Pour le moment, la liste s'affiche verticalement. Pour l'afficher horizontalement, faites flotter à gauche les éléments de liste.

```
ul. nav li {
  float: left;
}
```

Rappelez-vous que, lorsqu'un élément flotte, il ne prend plus d'espace dans le flot du document. Du coup, la liste parente n'a plus de contenu ; elle s'effondre et l'arrière-plan de la liste disparaît. Comme vous l'avez appris au Chapitre 3, il existe plusieurs moyens d'amener les éléments parents à contenir les enfants flottants. L'un d'eux consiste à ajouter un élément de dégagement. Malheureusement, cela nécessite d'ajouter une balise inutile à la page, ce qu'il vaut mieux éviter. L'une des autres méthodes consiste à faire flotter l'élément parent également et à le dégager plus bas, par exemple en utilisant le pied de page du site. La troisième méthode, que j'adopte habituellement, passe par l'utilisation d'overflow:hidden :

```
ul.nav {
  margin: 0;
  padding: 0;
```

```

list-style: none;
width: 72em;
overflow: hidden;
background: #FAA819 url(img/mainNavBg.gif) repeat-x;
}

```

Comme avec le précédent exemple de navigation, chacun des liens dans cette barre de navigation horizontale est créé de manière à se comporter comme un bouton, car la valeur `block` a été attribuée à sa propriété `display`. Pour que chaque bouton ait une taille fixe, vous pouvez fixer explicitement sa hauteur et sa largeur, mais ce choix peut poser des problèmes en termes de facilité de maintenance. Laissez plutôt la largeur de chacun d'eux se caler sur la taille du texte d'ancre. Au lieu de définir une largeur explicite, j'applique donc 3 cadratins de remplissage aux côtés gauche et droit de chaque lien d'ancre. Comme dans l'exemple précédent, on a centré le texte du lien verticalement en utilisant le paramètre de hauteur de ligne. Les soulignements de lien sont ensuite désactivés, et la couleur du lien passée en blanc :

```

ul.nav a {
  display: block;
  padding: 0 3em;
  line-height: 2.1em;
  text-decoration: none;
  color: #fff;
}

```

Je souhaite créer des séparateurs entre les liens dans la barre de navigation. Il est possible de définir des bordures horizontales sur l'élément de liste ou les ancrés mais, par souci de simplicité, je préfère appliquer une image d'arrière-plan aux liens d'ancre.

```

ul.nav a {
  display: block;
  padding: 0 2em;
  line-height: 2.1em;
  background: url(img/divider.gif) repeat-y left top;
  text-decoration: none;
  color: #fff;
}

```

Le premier lien dans la barre de navigation possède du coup un séparateur indésirable. On peut s'en débarrasser en ajoutant une classe au premier élément de la liste et en supprimant l'image d'arrière-plan :

```

ul.nav .first a {
  background-image: none;
}

```

Si vous ne vous préoccupez pas spécifiquement de prendre en charge Internet Explorer 6, vous pouvez aussi renoncer à la classe additionnelle et utiliser la pseudo-classe `:first-child` à la place.

```

ul.nav li:first-child a {
  background: none;
}

```

Pour finir, l'état de survol de cet exemple correspond à un simple changement de la couleur du lien :

```
ul.nav a:hover,
ul.nav a:active {
  color: #333;
}
```

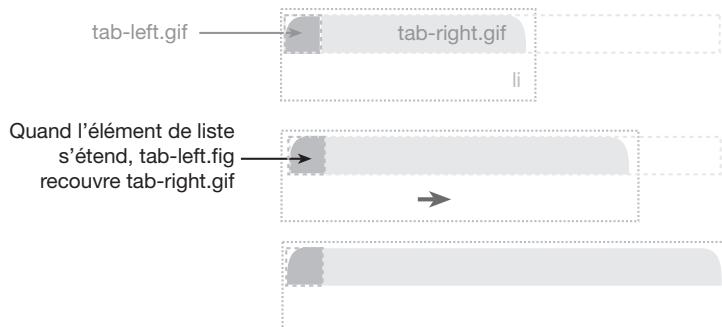
Et voilà le travail : une barre de navigation horizontale entièrement mise en forme et bien prise en charge par les différents navigateurs.

Système de navigation élastique à onglets

Au Chapitre 4, vous avez découvert la technique des portes coulissantes popularisée par Douglas Bowman et vu de quelle manière elle pouvait être utilisée pour créer des boîtes flexibles à bords arrondis. Elle peut également l'être pour créer un système de navigation extensible à onglets. Ces derniers sont créés à partir d'une grande image et d'une image de côté. Lorsque le texte dans les onglets s'étend, une partie plus importante de la grande image est révélée. La petite image reste calée à gauche, afin de couvrir le côté à angle droit de l'image plus grande et de compléter l'effet (voir Figure 6.5).

Figure 6.5

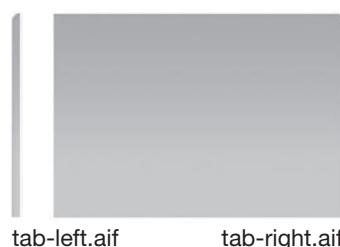
Exemple de la technique des "portes coulissantes".



La Figure 6.6 présente les deux images utilisées pour créer les onglets. Elles sont très grandes. On peut ainsi augmenter largement le pourcentage de la taille de la police sans que les onglets se dénaturent.

Figure 6.6

Les deux images qui composent les onglets.



Le code de cet exemple est exactement le même que celui du précédent exemple de barre de navigation horizontale :

```

<ul class="nav">
  <li><a href="home.htm">Accueil</a></li>
  <li><a href="about.htm">A propos</a></li>
  <li><a href="news.htm">Actualités</a></li>
  <li><a href="products.htm">Produits</a></li>
  <li><a href="services.htm">Services</a></li>
  <li><a href="clients.htm">Clients</a></li>
  <li><a href="case-studies.htm">Etudes de cas</a></li>
</ul>

```

Comme pour les exemples précédents, les marges et le remplissage sont ramenés à zéro, les puces des listes sont supprimées et une largeur est définie pour la barre de navigation. La valeur `hidden` est attribuée à la propriété `overflow` de la liste de navigation afin d'ouvrir un dégagement pour tous les éléments flottants qui y sont inclus.

```

ul.nav {
  margin: 0;
  padding: 0;
  list-style: none;
  width: 72em;
  overflow: hidden;
}

```

Ici aussi, les éléments de la liste sont rendus flottants à gauche afin qu'ils s'affichent horizontalement et non verticalement. Cette fois, la plus grande des deux images qui composent l'onglet est cependant appliquée sous forme d'image d'arrière-plan à l'élément de liste. Comme elle forme la partie droite de l'onglet, elle est positionnée à droite.

```

ul.nav li {
  float: left;
  background: url(img/tab-right.gif) no-repeat right top;
}

```

L'affichage des ancrés est aussi défini comme de niveau bloc (`display: block`) afin que l'utilisateur puisse cliquer n'importe où dans la zone. La largeur de chaque onglet est à nouveau contrôlée par la largeur du contenu. La valeur de hauteur de ligne contrôle également la hauteur. Pour compléter l'effet des onglets, la partie gauche de l'onglet est appliquée sous forme d'arrière-plan à l'ancre et alignée à gauche. Lorsque l'onglet change de taille, cette image reste alignée à gauche, par-dessus l'image plus grande et en couvrant son bord gauche à angle droit. Enfin, pour s'assurer que cette technique fonctionne sous Internet Explorer 5.2 sur Mac, les ancrés sont rendues flottantes également.

```

ul.nav li a {
  display: block;
  padding: 0 2em;
  line-height: 2.5em;
  background: url(img/tab-left.gif) no-repeat left top;
  text-decoration: none;
  color: #fff;
  float: left;
}

```

Pour créer l'effet de survol, on change simplement la couleur du lien :

```
ul.nav a:hover,
ul.nav a:focus {
    color: #333;
}
```

Le système de navigation résultant est présenté à la Figure 6.7.

Figure 6.7

Le système de navigation à portes coulissantes à sa taille normale.



Si vous augmentez la taille du texte dans votre navigateur, les onglets se redimensionnent parfaitement (voir Figure 6.8).

Figure 6.8

Le système de navigation à portes coulissantes, lorsque le texte est agrandi plusieurs fois.



Cette méthode offre un moyen simple et aisément accessible de créer des barres de navigation à onglets attrayantes et accessibles.

Menus déroulants Suckerfish

Malgré les problèmes qu'ils posent en termes d'utilisabilité, les menus déroulants continuent d'être un élément d'interface populaire sur Internet. Les solutions JavaScript abondent, mais nombre d'entre elles génèrent de fait des difficultés d'accessibilité, notamment parce qu'elles ne fonctionnent pas dans les navigateurs où le JavaScript est désactivé. Voilà pourquoi plusieurs pionniers ont exploré la voie des menus déroulants en CSS uniquement. Patrick Griffiths, l'un d'eux, a mis au point une technique de menus déroulants appelée Suckerfish (<http://www.alistapart.com/articles/dropdowns/>).

Cette technique est incroyablement simple : il suffit d'imbriquer la sous-navigation sur une liste à puces, de la positionner hors écran, puis de la repositionner lorsque l'élément parent de liste est survolé. Le résultat final est présenté à la Figure 6.9.

Figure 6.9

Un menu déroulant Suckerfish entièrement réalisé en CSS.



Commençons par créer la liste de navigation multiniveau.

```

<ul class="nav">
  <li><a href="/home/">Accueil</a></li>
  <li><a href="/products/">Produits</a>
    <ul>
      <li><a href="/products/silverback/">Silverback</a></li>
      <li><a href="/products/fontdeck/">Font Deck</a></li>
    </ul>
  </li>
  <li><a href="/services/">Services</a>
    <ul>
      <li><a href="/services/design/">Conception</a></li>
      <li><a href="/services/development/">Développement</a></li>
      <li><a href="/services/consultancy/">Consulting</a></li>
    </ul>
  </li>
  <li><a href="/contact/">Contact</a></li>
</ul>

```

Comme pour tous les exemples de navigation de ce chapitre, il faut d'abord ramener à zéro les marges et le remplissage et supprimer les puces par défaut. Il faut ensuite donner une largeur aux éléments de liste et les faire flotter à gauche. Pour la mise en forme, je souhaite donner aux listes de navigation une bordure et une couleur d'arrière-plan. Toutefois, comme les éléments de liste incorporés sont tous flottants, ils ne prennent pas d'espace et les listes s'effondrent sur elles-mêmes. Pour contourner ce problème, j'ai décidé de rendre les listes flottantes également.

```

ul.nav, ul.nav ul {
  margin: 0;
  padding: 0;
  list-style-type: none;
  float: left;
  border: 1px solid #486B02;
  background-color: #8BD400;
}

ul.nav li {
  float: left;
  width: 8em;
  background-color: #8BD400;
}

```

Pour être sûr que les éléments dans les menus déroulants s'empilent verticalement, il faut définir la largeur de la liste en lui attribuant la même valeur que la largeur des éléments de liste incorporés. Le menu déroulant commence maintenant à prendre forme.

Pour masquer les menus déroulants jusqu'à ce qu'ils soient activés, il faut les positionner de manière absolue (`position: absolute`) puis les masquer en les sortant à gauche de l'écran.

```

ul.nav li ul {
  width: 8em;
  position: absolute;
  left: -999em;
}

```

Le tour de magie se produit maintenant. En ajoutant une pseudo-classe `:hover` à l'élément de liste parent, on peut faire réapparaître la liste en changeant sa position pour la ramener à son emplacement de départ.

```
.nav li:hover ul {  
    left: auto;  
}
```

Les styles suivants obligent les liens de navigation à se comporter comme des éléments de niveau bloc et changent l'apparence de la liste en leur attribuant des couleurs d'arrière-plan et des bordures biseautées.

```
ul.nav a {  
    display: block;  
    color: #2B3F00;  
    text-decoration: none;  
    padding: 0.3em 1em;  
    border-right: 1px solid #486B02;  
    border-left: 1px solid #E4FFD3;  
}  
  
ul.nav li li a {  
    border-top: 1px solid #E4FFD3;  
    border-bottom: 1px solid #486B02;  
    border-left: 0;  
    border-right: 0;  
}  
  
/*suppression des bordures indésirables sur les derniers éléments de liste*/  
ul.nav li:last-child a {  
    border-right: 0;  
    border-bottom: 0;  
}  
ul a:hover,  
ul a:focus {  
    color: #E4FFD3;  
    background-color: #6DA203;  
}
```

Et le tour est joué : vous venez de créer une barre de navigation à menus déroulants simple, entièrement en CSS. Cette technique fonctionne dans la plupart des navigateurs web modernes, mais pas dans les anciennes versions d'Internet Explorer, qui ne reconnaissent pas la pseudo-classe `:hover` pour les éléments qui ne sont pas des ancrés. Pour contourner ce problème, vous pouvez utiliser quelques lignes de JavaScript ou un fichier de comportement .htc qui active cette fonctionnalité.

Le code JavaScript pour le correctif des menus déroulants sous Internet Explorer n'est pas du ressort de ce livre, mais vous trouverez plus de détails à ce sujet à la page (en anglais) <http://htmldog.com/articles/suckerfish/dropdowns/>.

Cartes-images CSS

Les cartes-images permettent aux développeurs web de spécifier des zones réactives dans une image. Ultra-populaires il y a quelques années, elles sont beaucoup moins courantes aujourd’hui. C’est en partie dû à la popularité de Flash et à la tendance qui a conduit à privilégier la simplicité d’un code moins orienté sur la présentation. Si les cartes-images sont toujours parfaitement valides en HTML, elles ont en effet le défaut de mélanger présentation et contenu. Il est cependant possible d’en créer avec une combinaison de listes, d’ancres et de styles CSS avancés.

Dans cet exemple, nous allons utiliser une photographie de quelques membres de l’équipe Clearleft posant à la manière d’un groupe de rock indépendant devant un mur graffité situé près des locaux (voir Figure 6.10). Lorsque l’utilisateur survole chaque personne, une boîte rectangulaire doit apparaître. S’il clique dessus, il est conduit vers le site web de la personne concernée.

Figure 6.10

Rich, Sophie, Cath, James et Paul posent devant le mur graffité à l’extérieur du bureau.



La première chose à faire est d’ajouter l’image à la page, dans une div nommée :

```
<div class="imagemap">
  
</div>
```

Ensuite, ajoutez une liste de liens vers le site web de chaque personne après l'image. Chaque élément de liste doit se voir attribuer une classe pour identifier la personne correspondante. Vous pouvez donner à chaque lien un attribut de titre (`title`) contenant le nom de la personne. Dans la plupart des navigateurs, le nom de la personne apparaît dans une info-bulle quand l'utilisateur survole le lien.

```
<div id="imagemap">
  
  <ul>
    <li class="rich">
      <a href="http://www.clagnut.com/" title="Richard Rutter">Richard Rutter</a>
    </li>
    <li class="sophie">
      <a href="http://www.wellieswithwings.org/" title="Sophie Barrett">Sophie
      ↪Barrett</a>
    </li>
    <li class="cath">
      <a href="http://www.electricelephant.com/" title="Cathy Jones">Cathy Jones</a>
    </li>
    <li class="james">
      <a href="http://www.jeckecko.net/blog/" title="James Box">James Box</a>
    </li>
    <li class="paul">
      <a href="http://twitter.com/nicepaul" title="Paul Annett">Paul Annett</a>
    </li>
  </ul>
</div>
```

Fixez la largeur et la hauteur de la div en les faisant correspondre aux dimensions de l'image. Ensuite, attribuez-lui un positionnement relatif (`position: relative`). Cette dernière étape est la clé de cette technique, car elle permet aux liens enclos d'être positionnés de manière absolue, par rapport aux bords de la div et donc de l'image.

```
.imagemap {
  width: 333px;
  height: 500px;
  position: relative; /* La clé de cette technique */
}
```

Il ne faut pas que les puces s'affichent. Faites-les donc disparaître en attribuant la valeur `none` à la propriété `list-style`. Pour bien faire, ramenez aussi à zéro les marges et le remplissage de la liste :

```
.imagemap ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
```

La prochaine étape vise à mettre en forme les liens. En positionnant les liens d'ancre de manière absolue, on les déplace vers le coin supérieur gauche de la div conteneur. Ils peuvent ensuite être positionnés de manière individuelle au-dessus de chaque personne, afin de former les zones réactives. Vous devez cependant d'abord définir leurs largeurs et leurs

hauteurs pour créer les zones réactives désirées. Le texte du lien est toujours affiché ; il est donc nécessaire de le masquer en le sortant de l'écran grâce à un grand retrait négatif :

```
.imagemap a {  
    position: absolute;  
    display: block;  
    width: 50px;  
    height: 60px;  
    text-indent: -1000em;  
}
```

Les liens individuels peuvent maintenant être positionnés au-dessus des personnes correspondantes :

```
.imagemap .rich a {  
    top: 50px;  
    left: 80px;  
}  
  
.imagemap .sophie a {  
    top: 90px;  
    left: 200px;  
}  
  
.imagemap .cath a {  
    top: 140px;  
    left: 55px;  
}  
.imagemap .james a {  
    top: 140px;  
    left: 145px;  
}  
  
.imagemap .paul a {  
    top: 165px;  
    left: 245px;  
}
```

Enfin, pour créer l'effet de survol, on applique une bordure aux liens en cas de survol :

```
.imagemap a:hover,  
imagemap a:focus {  
    border: 1px solid #fff;  
}
```

La base de la technique est terminée. Si vous survolez les images, vous obtenez maintenant le résultat présenté à la Figure 6.11.

Tout cela suppose néanmoins que vous possédez un navigateur bien en jambes, comme Safari ou Firefox. Si vous utilisez Internet Explorer, il ne se passe rien ! Internet Explorer n'aime pas afficher les liens dont le contenu est masqué hors écran, même si vous définissez explicitement des largeurs et des hauteurs. Mais il existe une solution.

Figure 6.11

La carte-image CSS pendant le survol.



Si vous donnez un arrière-plan aux liens d'ancre, vous parviendrez à tromper Internet Explorer afin qu'il se comporte correctement. Le problème est que nous ne souhaitons pas donner d'arrière-plan aux liens, puisqu'ils sont censés être masqués ! Vous pourriez tenter de définir un arrière-plan transparent, mais cela ne paraît pas fonctionner. Pourquoi donc ne pas utiliser une image PNG ou GIF transparente ?

```
.imagemap a {  
    position: absolute;  
    display: block;  
    background-image: url(/img/shim.gif);  
    width: 60px;  
    height: 80px;  
    text-indent: -1000em;  
}
```

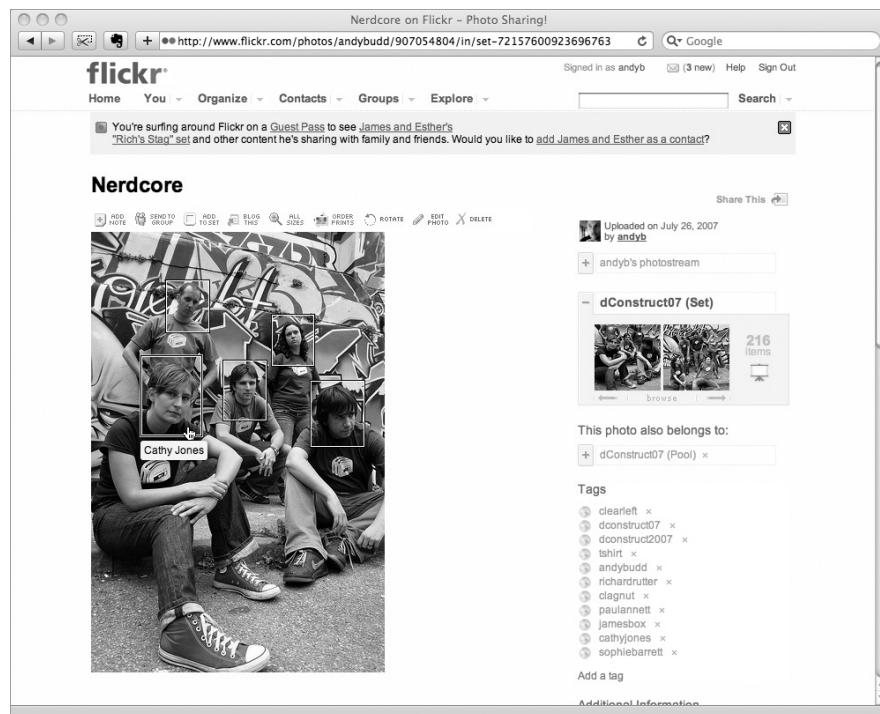
Bizarrement, il n'y a même pas besoin de pointer sur une véritable image ! Vous pouvez vous contenter de spécifier une URL inexistante et tromper ainsi Internet Explorer. Enfin, quand même, créer un lien vers une URL qui n'existe pas, c'est mal, même si c'est pour corriger les défauts d'un navigateur bogué ! J'utilise donc pour ma part une véritable image.

Cartes-images à la mode Flickr

Si vous avez déjà utilisé le service de partage de photos Flickr, vous avez sans doute vu une technique similaire qui est utilisée pour l'annotation des images (voir Figure 6.12). Lorsque vous survolez une image annotée, une boîte à double bordure apparaît sur la zone qui contient chaque note. Lorsque vous survolez l'une de ces boîtes, elle passe en surbrillance et affiche la note. Moyennant quelques ajustements, il est possible de produire un résultat analogue avec la précédente technique.

Figure 6.12

Notes sur les images
Flickr.



Pour créer la boîte à double bordure, vous devez placer deux balises span supplémentaires à l'intérieur de chaque lien d'ancre et en ajouter une à la note aussi. Une fois que ces balises span ont été ajoutées, la liste complétée doit ressembler à ceci :

```

<ul>
  <li class="rich">
    <a href="http://www.clagnut.com/">
      <span class="outer">
        <span class="inner">
          <span class="note">Richard Rutter</span>
        </span>
        </span>
      </a>
    </li>
  ...
</ul>

```

Le code CSS est au départ identique à celui de l'exemple précédent : vous donnez à la div conteneur les dimensions de l'image et spécifiez un positionnement relatif. Le remplissage et les marges de la liste sont ramenés à zéro et les puces supprimées :

```

.imagemap {
  width: 333px;
  height: 500px;
}

```

```
    position: relative;
}

.imagemap ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
```

Comme auparavant, les liens d'ancre incorporés sont positionnés de manière absolue. Cette fois, vous allez cependant définir les dimensions sur les balises `span` internes et laisser les balises `span` et les liens d'ancre externes prendre forme autour d'elles. Pour ma part, j'ai attribué une bordure foncée à la balise `span` externe et une bordure claire à la balise `span` interne, afin de marquer leurs positions dans l'image. Enfin, comme je ne souhaite pas masquer le texte à l'intérieur des liens d'ancre, mais l'afficher sous forme d'info-bulle, je lui donne une mise en forme simple.

```
.imagemap a {
  position: absolute;
  display: block;
  background-image: url(/img/shim.gif);
  color: #000;
  text-decoration: none;
  border: 1px solid transparent;
}

.imagemap a .outer {
  display: block;
  border: 1px solid #000;
}

.imagemap a .inner {
  display: block;
  width: 50px;
  height: 60px;
  border: 1px solid #fff;
}
```

Comme auparavant, vous devrez positionner les ancrés au-dessus de chaque personne :

```
.imagemap .rich a {
  top: 50px;
  left: 80px;
}

.imagemap .sophie a {
  top: 90px;
  left: 200px;
}

.imagemap .cath a {
  top: 140px;
  left: 55px;
}

.imagemap .james a {
```

```
    top: 140px;  
    left: 145px;  
}  
  
.imagemap .paul a {  
    top: 165px;  
    left: 245px;  
}
```

Vous pouvez ensuite appliquer l'effet de survol au lien d'ancre. Pour cela, changez la couleur de bordure de l'ancre, d'abord transparente, en la rendant jaune, pour les états `hover` et `focus` :

```
.imagemap a:hover,  
.imagemap a:focus {  
    border-color: #d4d82d;  
}
```

Pour afficher la note lorsque l'utilisateur survole la zone réactive, vous devez d'abord positionner le contenu de la balise `span` de note sous la zone réactive. Pour cela, définissez un positionnement absolu pour cette balise et donnez-lui une position négative en bas (`bottom`). Pour soigner les notes, définissez une largeur, un remplissage et une couleur d'arrière-plan, puis centrez le texte :

```
.imagemap a .note {  
    position: absolute;  
    bottom: -3em;  
    width: 7em;  
    padding: 0.2em 0.5em;  
    background-color:#ffc;  
    text-align: center;  
}
```

La Figure 6.13 présente le résultat dans un navigateur.

Figure 6.13

Les survols à la Flickr commencent à prendre forme.



Comme vous pouvez le voir, l'effet commence à prendre forme. Les notes ont bonne apparence, mais il serait préférable qu'elles soient centrées horizontalement sous la zone réactive, au lieu d'être ferrées à gauche. Vous pouvez le faire en positionnant le bord gauche de la balise `span` de note au milieu de la zone réactive. Ensuite, déplacez cette balise vers la gauche, à mi-largeur de la note, en utilisant des marges négatives. La zone réactive de cet exemple fait 50 pixels de large. J'ai donc défini la position `left` avec une valeur de 25 pixels. Les notes font 8 cadratins de large, remplissage inclus. Avec une marge négative de 4 cadratins, on parvient ainsi à les centrer horizontalement sous la zone réactive.

```
.imagemap a .note {
  position: absolute;
  bottom: -3em;
  width: 7em;
  padding: 0.2em 0.5em;
  background-color:#ffc;
  text-align: center;
  left: 25px;
  margin-left: -4em;
}
```

Maintenant que les notes sont centrées, il est temps de les rendre interactives. Elles doivent être masquées par défaut et ne s'afficher que lors du survol de la zone réactive. Pour cela, on pourrait attribuer la valeur `none` à la propriété `display` puis la ramener à `block` lors du survol de l'ancre, mais cela empêcherait les lecteurs d'écran d'accéder au contenu de la note. Je préfère donc masquer le texte en le déplaçant à gauche hors de l'écran et en le repositionnant lors du survol :

```
.imagemap a .note {
  position: absolute;
  bottom: -3em;
  width: 7em;
  padding: 0.2em 0.5em;
  background-color:#ffc;
  text-align: center;
  left: -1000em;
  margin-left: -5em;
}

.imagemap a:hover .note,
.imagemap a:focus .note {
  left: 25px;
}
```

Nous y sommes presque. Un dernier ajustement. Au lieu d'afficher tout le temps les bordures doubles de la zone réactive, on pourrait ne les faire apparaître que lors du survol de l'image. Il serait alors possible d'observer l'image normalement, sans les marques des zones réactives. Lorsque le curseur survolerait l'image, les zones réactives apparaîtraient, afin de signaler à l'utilisateur que d'autres informations peuvent être découvertes. Vous pouvez réaliser cela en rendant les bordures sur les `span` externe et interne transparentes par défaut, puis en leur attribuant une couleur pour le survol :

```
.imagemap a .outer {
  display: block;
```

```
border: 1px solid transparent;
}

.imagemap a .inner {
  display: block;
  width: 50px;
  height: 60px;
  border: 1px solid transparent;
}

.imagemap:hover a .outer,
.imagemap:focus a .outer {
  border-color: #000;
}

.imagemap:hover a .inner,
.imagemap:focus a .inner {
  border-color: #fff;
}
```

Malheureusement, comme vous le savez déjà, Internet Explorer 6 ne prend en charge que le survol des liens. Pour résoudre ce problème, il est aussi judicieux d'afficher les bordures lorsque l'utilisateur survole directement les zones réactives :

```
.imagemap:hover a .outer,
.imagemap:focus a .outer,
.imagemap a:hover .outer,
.imagemap a:focus .outer {
  border: 1px solid #000;
}

.imagemap:hover a .inner,
.imagemap:focus a .inner,
.imagemap a:hover .inner,
.imagemap a:focus .inner {
  border: 1px solid #fff;
}
```

Et voilà le travail : une carte-image CSS avancée à la mode Flickr (voir Figure 6.14).

Figure 6.14

La version terminée de notre carte-image à la mode Flickr.



Survol distant

Le *survol distant* est un événement de survol qui déclenche une modification d'affichage dans un autre endroit de la page. Il s'opère en imbriquant un élément, ou plusieurs, à l'intérieur d'un lien d'ancre. Les éléments imbriqués peuvent ensuite être positionnés un à un de manière absolue. Même s'ils s'affichent à différents endroits, ils sont tous contenus dans la même ancre parente et réagissent donc au même événement de survol. Le survol d'un élément peut ainsi affecter le style d'un autre élément.

Pour cet exemple, nous allons reprendre la technique de la carte-image CSS simple en plaçant une liste de liens à côté de l'image. Lorsque l'utilisateur survolera ces liens, il déclenchera l'affichage des zones réactives de l'image. Lorsqu'il survolera les zones réactives de l'image, il déclenchera l'affichage des liens texte.

Le code HTML de cet exemple est analogue à celui de l'exemple de carte-image CSS simple. Vous aurez cependant besoin de deux balises `span` supplémentaires : l'une entourant le texte du lien et l'autre, vide, qui doit agir comme zone réactive. Vous pourrez ainsi positionner les liens texte à côté de l'image et les zones réactives au-dessus de chacune des personnes.

```
<div class="remote">
  
  <ul>

    <li class="rich">
      <a href="http://www.clagnut.com/" title="Richard Rutter">
        <span class="hotspot"></span>
        <span class="link">&raquo; Richard Rutter</span>
      </a>
    </li>

    <li class="sophie">
      <a href="http://www.wellieswithwings.org/" title="Sophie Barrett">
        <span class="hotspot"></span>
        <span class="link">&raquo; Sophie Barrett</span>
      </a>
    </li>

    <li class="cath">
      <a href="http://www.electricelephant.com/" title="Cathy Jones">
        <span class="hotspot"></span>
        <span class="link">&raquo; Cathy Jones</span>
      </a>
    </li>

    <li class="james">
      <a href="http://www.jeckecko.net/blog/" title="James Box">
        <span class="hotspot"></span>
        <span class="link">&raquo; James Box</span>
      </a>
    </li>
```

```
<li class="paul">
  <a href="http://twitter.com/nicepaul" title="Paul Annett">
    <span class="hotspot"></span>
    <span class="link">&raquo; Paul Annett</span>
  </a>
</li>

</ul>
</div>
```

La mise en forme de base de la liste est la même que dans l'exemple de la carte-image :

```
.remote {
  width: 333px;
  height: 500px;
  position: relative;
}

.remote ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
```

La première chose à faire est de définir un positionnement absolu pour les zones réactives et de spécifier leurs dimensions. Dans cet exemple, trois des zones réactives font la même taille, les deux dernières étant légèrement plus grandes. J'ai donc défini les tailles par défaut d'abord, avant de les redéfinir en cas de besoin. Comme pour la technique précédente, toutes les ancrées se trouvent alors positionnées au niveau du coin supérieur gauche de l'image. Vous pouvez ensuite positionner chaque zone réactive sur la personne appropriée de l'image, avec les propriétés de positionnement `top` et `left`.

```
.remote a .hotspot {
  width: 50px;
  height: 60px;
  position: absolute;
}

.remote .rich a .hotspot {
  top: 50px;
  left: 80px;
}

.remote .sophie a .hotspot {
  top: 90px;
  left: 200px;
}

.remote .cath a .hotspot {
  top: 140px;
  left: 55px;
  width: 60px;
```

```
height: 80px;
}

.remote .james a .hotspot {
  top: 140px;
  left: 145px;
}

.remote .paul a .hotspot {
  top: 165px;
  left: 245px;
  width: 60px;
  height: 80px;
}
```

Les balises span qui contiennent le texte des liens sont aussi positionnées de manière absolue et ont une largeur de 15 cadratins. Elles sont positionnées par rapport à la liste conteneur également, ici à droite de l'image, à l'aide d'une position droite négative. Pour finir, les liens reçoivent un style de curseur pour s'assurer que l'icône appropriée s'affiche dans Internet Explorer.

```
.remote a .link {
  position: absolute;
  display: block;
  width: 10em;
  right: -11em;
  cursor: pointer;
}

.remote .rich a .link {
  top: 0;
}

.remote .sophie a .link {
  top: 1.2em;
}

.remote .cath a .link {
  top: 2.4em;
}

.remote .james a .link {
  top: 3.6em;
}

.remote .paul a .link {
  top: 4.8em;
}
```

Les zones réactives doivent maintenant se trouver au bon endroit, de même que les liens texte.

Pour créer l'effet de survol sur la zone réactive lorsque l'utilisateur passe sur la zone réactive elle-même ou le texte, vous devez appliquer une bordure à la balise span de la zone réactive lors du survol de l'ancre parente :

```
.remote a:hover .hotspot,  
.remote a:focus .hotspot {  
  border: 1px solid #fff;  
}
```

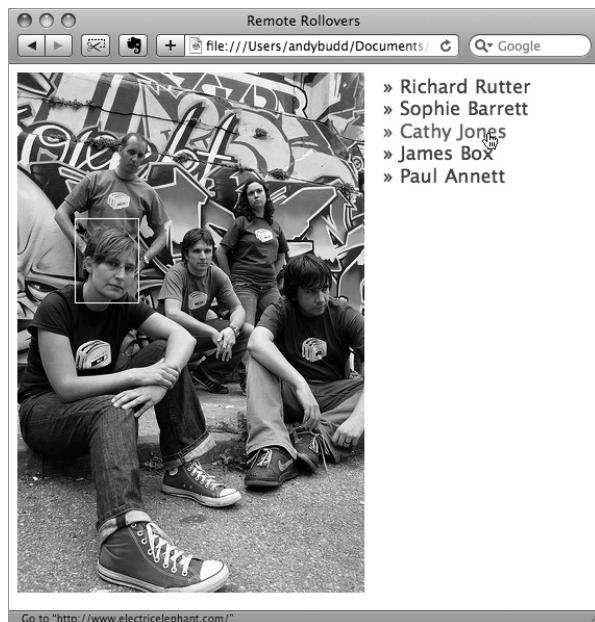
De la même manière, pour changer la couleur du texte lorsque l'utilisateur survole le texte ou la zone réactive, vous devez changer le style de la balise `span` lorsque l'ancre parente est survolée ou activée :

```
.remote a:hover .link ,  
.remote a:focus .link {  
  color: #0066FF;  
}
```

Si vous testez cet exemple, vous verrez qu'il fonctionne parfaitement dans Safari et Firefox (voir Figure 6.15). Lorsque vous survolez le nom d'une personne, le texte du lien change de couleur, et une boîte apparaît au-dessus de la personne dans l'image. Le même événement se produit quand vous survolez la personne dans l'image.

Figure 6.15

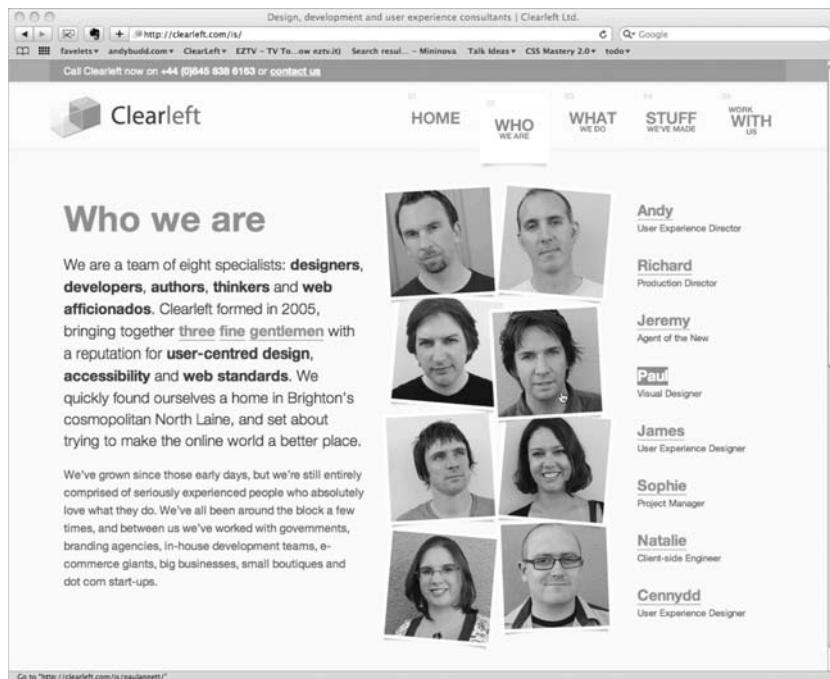
Démonstration du survol distant. Lorsque l'utilisateur survole le texte du lien à côté de l'image, un contour apparaît sur la personne correspondante dans l'image.



La mise en forme de cet exemple est assez simple, mais rien ne vous limite pourtant, hormis votre imagination. Nous avons d'ailleurs utilisé une version légèrement modifiée de cette technique dans la section "Qui sommes-nous ?" du site Clearleft (<http://clearleft.com/is/>) [voir Figure 6.16].

Figure 6.16

Lorsque vous survolez les photos de l'équipe Clearleft, le nom de la personne passe en surbrillance sur la liste à droite.



Note à propos des listes de définitions

Au cours de ce chapitre, vous avez vu comment les listes à puces (et par extension, les listes numérotées) pouvaient être utilisées pour créer différents effets. Il existe cependant un troisième type de liste, souvent négligé, mais qui connaît un récent regain d'intérêt : la liste de définitions. Elle est constituée de deux composants-clés : un terme de définition `<dt>` et une ou plusieurs descriptions de définition `<dd>`.

```
<dl>
  <dt>Apple</dt>
    <dd>Red, yellow or green fruit</dd>
    <dd>Computer company</dd>
  <dt>Bananna</dt>
    <dd>Curved yellow fruit</dd>
</dl>
```

Comme son nom le suggère, son principal but est de créer des balises de définitions. La spécification HTML reste toutefois vague et suggère que ce type de liste pourrait être utilisé pour d'autres applications, comme des propriétés de produits ou des conversations. Voilà qui rend le concept de définition plutôt flou, mais c'est relativement logique dans le contexte historique du HTML qui est avant tout un langage simple de formatage du texte.

Des pionniers du Web ont profité du fait que les listes de définitions pouvaient être utilisées pour regrouper structurellement une série d'éléments liés et ont commencé à les utiliser pour créer toutes sortes de choses, des catalogues de produits aux galeries d'images en passant

par des formulaires ou même des mises en page complètes. Si ces techniques sont souvent très astucieuses, elles ont cependant le tort de tirer un peu trop sur la corde du concept de définition, au point d'en rompre le sens proprement naturel.

L'un des arguments en leur faveur tient à ce qu'aucun autre élément HTML ne permet ce type d'association. Ce n'est pas exactement vrai, car le but de l'élément div est justement de regrouper un document en sections logiques. Plus inquiétant, c'est exactement le même type d'argument que l'on utilise quand on veut justifier le recours aux tableaux pour la mise en page. C'est le signe que les listes de définitions commencent à être utilisées de manière inappropriée.

Pour plus d'informations sur les listes de définitions, consultez l'excellent article (en anglais) de Mark Norman Francis sur le site 24 Ways (<http://24ways.org/2007/my-other-christmas-present-is-a-definition-list>).

En résumé

Vous avez découvert à quel point les listes pouvaient être flexibles. Vous avez vu comment créer des barres de navigation verticales et horizontales, et notamment des systèmes à onglets accessibles. Pour finir, vous avez appris à utiliser le positionnement pour créer des menus déroulants, des cartes-images et des survols distants entièrement en CSS.

Au chapitre suivant, vous apprendrez à créer des structures de formulaire et des tableaux de données accessibles et à les mettre en forme en CSS.

7

Mise en forme des formulaires et des tableaux de données

À l'heure du tout interactif, les formulaires sont devenus un composant incontournable des applications web modernes. Ils permettent aux utilisateurs d'interagir avec les systèmes, de poster des commentaires ou d'effectuer des réservations complexes auprès d'agences de voyages. Ils peuvent être très simples, n'incluant qu'un champ d'adresse e-mail et un champ de message, ou formidablement complexes, s'étendant sur plusieurs pages. La mise en forme des formulaires s'est traditionnellement effectuée à l'aide de tableaux, mais, ici, vous allez découvrir qu'elle peut se faire, y compris pour les formulaires complexes, à l'aide de CSS.

Les tableaux retrouvent peu à peu leur fonction première, qui était d'afficher des données tabulaires et non de structurer des maquettes de pages. Les applications web doivent capturer les données des utilisateurs, mais elles doivent aussi de plus en plus souvent les afficher dans des formats utilisables et faciles à comprendre. Les formulaires et les tableaux de données ont été relativement négligés au profit d'un travail plus approfondi sur la conception des sites web. Cependant, une bonne approche de la gestion et l'interaction de l'information peut faire pencher la balance sur la réussite ou non d'un site web.

Au cours de ce chapitre, vous apprendrez à :

- créer des tableaux de données attrayants et accessibles ;
- créer des mises en page simples et complexes de formulaires ;
- mettre en forme différents types d'éléments de formulaire ;
- fournir des retours d'information accessibles pour vos formulaires.

Mise en forme des tableaux de données

Beaucoup de développeurs ont pris conscience des inconvénients des maquettes tabulaires et évitent maintenant dès qu'ils le peuvent de les utiliser pour leurs mises en page. Un petit groupe d'individus a poussé le vice jusqu'à tenter de se débarrasser complètement des tableaux, en créant tout en CSS, même les calendriers. Cela part sans doute d'une bonne intention, mais il faut admettre que les calendriers proposent par nature un contenu tabulaire. Il ne s'agit ni plus ni moins que de lignes de semaines et de colonnes de jours. C'est précisément pourquoi les tableaux ont parfaitement leur place sur le Web.

Même relativement simples, les tableaux de données peuvent être difficiles à lire quand ils contiennent plus de quelques lignes et colonnes. Sans séparation entre les cellules de données, les informations se fondent les unes dans les autres et produisent un assemblage confus et surchargé (voir Figure 7.1).

Figure 7.1

Les tableaux de données compacts peuvent être très déroutants au premier coup d'œil.

≤ Janvier 2008 ≥							
D	L	M	M	J	V	S	
30	31	1	2	3	4	5	
6	7	8	9	10	11	12	
13	14	15	16	17	18	19	
20	21	22	23	24	25	26	
27	28	29	30	31	1	2	

À l'inverse, les tableaux exagérément aérés peuvent être aussi difficiles à lire, car les colonnes et les cellules perdent alors leur structuration visuelle. C'est particulièrement délicat lorsque vous essayez de suivre des lignes d'informations dans les tableaux où l'espacement des colonnes est très important, comme celui de la Figure 7.2. Si vous n'y prêtez attention, vous risquez de tomber dans la mauvaise ligne en passant d'une colonne à l'autre. On le remarque particulièrement au milieu du tableau, là où l'on est le plus éloigné des bords haut et bas qui offrent un ancrage visuel.

≤ Janvier 2008 ≥							
D	L	M	M	J	V	S	
30	31	1	2	3	4	5	
6	7	8	9	10	11	12	
13	14	15	16	17	18	19	
20	21	22	23	24	25	26	
27	28	29	30	31	1	2	

Figure 7.2

Les tableaux très espacés peuvent aussi être difficiles à comprendre instantanément.

En passant quelques minutes à concevoir vos tableaux de données, vous pouvez considérablement améliorer leur compréhension et la vitesse à laquelle les informations seront récupérées. Par exemple, les dates à la Figure 7.3 sont espacées confortablement avec un léger remplissage vertical et horizontal. Elles ont aussi été mises en valeur avec un subtil effet de biseau, suggérant à l'utilisateur qu'il peut cliquer dessus. Les principaux titres de colonne ont été distingués des données à l'aide de couleurs d'arrière-plan légèrement différentes, d'une bordure inférieure et d'un traitement typographique particulier. Au final, on obtient un widget de calendrier facile à utiliser.

Figure 7.3

Le tableau de données mis en forme.

Janvier 2008						
Dim	Lun	Mar	Mer	Jeu	Ven	Sam
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2

Éléments spécifiques des tableaux

Si les tableaux de données peuvent être difficiles à lire pour les utilisateurs qui possèdent une bonne vue, imaginez combien la tâche est compliquée et frustrante pour ceux qui se servent de lecteurs d'écran. Heureusement, la spécification HTML inclut un certain nombre d'éléments et d'attributs destinés à améliorer l'accessibilité des tableaux de données pour ces périphériques. Actuellement, tous ces éléments ne sont pas pris en charge par les lecteurs d'écran, mais il est bon de les utiliser chaque fois que c'est possible.

Résumé et légende

Le premier de ces éléments est la légende de tableau, qui, au fond, fait office de titre. Bien que cet élément ne soit pas requis, l'utiliser si l'occasion le permet est toujours judicieux. Ici, je m'en sers pour indiquer aux utilisateurs le mois qu'ils observent. L'autre ajout intéressant est le résumé de tableau. L'attribut `summary` peut être appliqué à la balise `table`. Il est utilisé pour décrire le contenu du tableau. À la manière de l'attribut `alt` des images, il doit résumer les données. Un résumé bien écrit peut par lui-même épargner la lecture du contenu du tableau.

```
<table class="cal" summary="Un calendrier sélecteur de date">
  <caption>
    <a href="cal.php?month=dec08" rel="prev">&lt;</a> Janvier 2008
    ↵ <a href="cal.php?month=feb09" rel="next">&gt;</a>
  </caption>
</table>
```

thead, tbody et tfoot

`thead`, `tfoot` et `tbody` permettent de décomposer les tableaux en sections logiques. Par exemple, vous pouvez placer tous les titres de colonne dans l'élément `thead`, qui permet d'appliquer une mise en forme particulière à cette zone. Si vous choisissez d'utiliser un élément `thead` ou `tfoot`, vous devez avoir au moins un élément `tbody`. Vous ne pouvez employer qu'un seul élément `thead` et `tfoot` par tableau, mais vous pouvez avoir plusieurs éléments `tbody` pour faciliter la décomposition des tableaux complexes en portions plus faciles à gérer.

Les titres de ligne et de colonne doivent être marqués avec les balises `th` plutôt que `td`, sauf pour les éléments qui correspondent à la fois à un titre et à des données, auquel cas, il convient de conserver `td`. Les titres de tableau peuvent être associés à un attribut `scope` avec la valeur `row` ou `col` pour indiquer s'il s'agit de titres de ligne ou de colonne. Ils peuvent également avoir la valeur `rowgroup` ou `colgroup` s'ils sont liés à plusieurs lignes ou colonnes.

```
<thead>
  <tr>
    <th scope="col">Dim</th>
    <th scope="col">Lun</th>
    <th scope="col">Mar</th>
    <th scope="col">Mer</th>
    <th scope="col">Jeu</th>
    <th scope="col">Ven</th>
    <th scope="col">Sam</th>
  </tr>
</thead>
```

col et colgroup

L'élément `tr` permet aux développeurs d'appliquer des styles à des lignes entières, mais il est bien plus difficile d'appliquer un style à une colonne entière. Pour résoudre ce problème, le W3C a introduit les éléments `colgroup` et `col`. Un `colgroup` est utilisé pour définir et regrouper une ou plusieurs colonnes utilisant l'élément `col`. Malheureusement, peu de navigateurs prennent en charge leur mise en forme.

```
<colgroup>
  <col id="sun" />
  <col id="mon" />
  <col id="tue" />
  <col id="wed" />
  <col id="thu" />
  <col id="fri" />
  <col id="sat" />
</colgroup>
```

Balisage des tableaux de données

En combinant tous ces éléments et attributs HTML, vous pouvez créer le quadrillage simple du tableau de calendrier présenté à la Figure 7.3.

```
<table class="cal" summary="Un calendrier sélecteur de date">
  <caption>
    <a href="#" rel="prev">&lt;</a> Janvier 2008 <a href="#" rel="next">&gt;</a>
  </caption>
  <colgroup>
    <col id="sun" />
    <col id="mon" />
    <col id="tue" />
    <col id="wed" />
    <col id="thu" />
    <col id="fri" />
```

```
<col id="sat" />
</colgroup>

<thead>
  <tr>
    <th scope="col">Dim</th>
    <th scope="col">Lun</th>
    <th scope="col">Mar</th>
    <th scope="col">Mer</th>
    <th scope="col">Jeu</th>
    <th scope="col">Ven</th>
    <th scope="col">Sam</th>
  </tr>
</thead>

<tbody>
  <tr>
    <td class="null">30</td>
    <td class="null">31</td>
    <td><a href="#">1</a></td>
    <td><a href="#">2</a></td>
    <td><a href="#">3</a></td>
    <td><a href="#">4</a></td>
    <td><a href="#">5</a></td>
  </tr>
  <tr>
    <td><a href="#">6</a></td>
    <td><a href="#">7</a></td>
    <td class="selected"><a href="#">8</a></td>
    <td><a href="#">9</a></td>
    <td><a href="#">10</a></td>
    <td><a href="#">11</a></td>
    <td><a href="#">12</a></td>
  </tr>
  ...
</tbody>
</table>
```

Mise en forme du tableau

La spécification CSS possède deux modèles de bordure de tableau : `separate` et `collapsed`. Dans le modèle `separate`, les bordures sont placées autour des cellules individuelles. Dans le modèle `collapsed`, les cellules partagent leurs bordures. La plupart des navigateurs utilisent par défaut le modèle `separate`, mais le modèle `collapsed` est généralement plus utile. L'une des premières choses à faire est donc d'attribuer la valeur `collapse` à la propriété `border-collapse` du tableau. Pour les besoins de la démonstration, je vais cependant conserver ici les doubles bordures afin de créer un effet biseauté. Je commence donc par attribuer la valeur `separate` à la propriété `border-collapse`. Ensuite, par souci esthétique, je centre tout le texte dans le tableau et supprime le remplissage et les marges par défaut.

```
table.cal {
  border-collapse: separate;
  border-spacing: 0;
  text-align: center;
  color: #333;
}

.cal th, .cal td {
  margin: 0;
  padding: 0;
}
```

La spécification CSS inclut une propriété `border-spacing` qui permet de contrôler l'espacement entre les cellules. Malheureusement, Internet Explorer 7 et ses versions antérieures ne la comprennent pas. Vous devez donc vous rabattre sur l'ancien attribut `cellspacing` qui a au moins le mérite d'être fiable. Cet attribut est à strictement parler un attribut de présentation. Il s'agit cependant encore de HTML valide, et c'est le seul moyen de contrôler l'espacement des cellules dans Internet Explorer 6 et 7.

```
<table cellspacing="0" class="cal" summary="Un calendrier sélecteur de date">
```

Ajout de style visuel

Le travail de fond est réalisé. Il est maintenant temps d'ajouter le style visuel. Pour faire en sorte que la légende de tableau ressemble un peu plus à un titre standard, vous pouvez augmenter la taille de la police et la passer en gras. Vous pouvez également donner à la légende un peu d'espace en appliquant un remplissage vertical.

```
.cal caption {
  font-size: 1.25em;
  padding-top: 0.692em;
  padding-bottom: 0.692em;
  background-color: #d4dde6;
}
```

Pour positionner les liens Précédent et Suivant des deux côtés du mois courant, donnez-leur une marge horizontale et faites-les respectivement flotter à gauche et à droite. Vous pouvez ensuite leur attribuer une zone réactive plus importante en appliquant un remplissage. Pour mettre en forme ces liens, j'ai choisi le sélecteur d'attribut afin de cibler leurs attributs `rel`. Pour une prise en charge par les anciens navigateurs, vous pouvez cependant ajouter une classe à chaque lien. Une fois que vous avez positionné ces liens, mettez-les en forme à votre idée. Dans cet exemple, je me contente de changer leur couleur d'arrière-plan lorsque l'utilisateur les survole.

```
.cal caption [rel="prev"] {
  float: left;
  margin-left: 0.2em;
}

.cal caption [rel="next"] {
```

```

    float: right;
    margin-right: 0.2em;
}

.cal caption a:link,
.cal caption a:visited {
    text-decoration: none;
    color: #333;
    padding: 0 0.2em;
}

.cal caption a:hover,
.cal caption a:active,
.cal caption a:focus {
    background-color: #6d8ab7;
}

```

Pour distinguer la ligne initiale de titres de tableau, je lui attribue un arrière-plan un peu plus clair que le reste du tableau, avec un soulignement subtil. Je réduis aussi légèrement la taille du texte par rapport au reste du tableau.

```

.cal thead th {
    background-color: #d4dde6;
    border-bottom: 1px solid #a9bacb;
    font-size: 0.875em;
}

```

Par défaut, je souhaite que le texte dans le corps du tableau soit grisé, afin d'indiquer qu'il ne peut pas être sélectionné. Vous remarquerez que j'ai également appliqué au texte une ombre subtile.

```

.cal tbody {
    color: #a4a4a4;
    text-shadow: 1px 1px 1px white;
    background-color: #d0d9e2;
}

```

Pour donner aux cellules du tableau un effet biseauté, vous devez définir des couleurs différentes de chaque côté (des couleurs plus claires en haut et à gauche et des couleurs plus sombres en bas et à droite). Vous devez ensuite mettre en forme les liens d'ancre. Ici, je les passe en mode bloc (`display: block`) et j'applique un remplissage pour créer une zone réactive analogue à un bouton. Je passe également en gras les polices et choisis un arrière-plan légèrement plus foncé.

```

.cal tbody td {
    border-top: 1px solid #e0e0e1;
    border-right: 1px solid #9f9fa1;
    border-bottom: 1px solid #acacac;
    border-left: 1px solid #dfdfc0;
}

.cal tbody a {
    display: block;
}

```

```
text-decoration: none;
color: #333;
background-color: #c0c8d2;
font-weight: bold;
padding: 0.385em 0.692em 0.308em 0.692em;
}
```

Pour finir, définissez un état de survol pour les liens d'ancre. Les dates précédemment sélectionnées hériteront aussi de ce style par l'inclusion d'une classe sélectionnée. Ici, je passe les liens en blanc sur fond bleu et je leur attribue une ombre de texte subtile.

```
.cal tbody a:hover,
.cal tbody a:focus,
.cal tbody a:active,
.cal tbody .selected a:link,
.cal tbody .selected a:visited,
.cal tbody .selected a:hover,
.cal tbody .selected a:focus,
.cal tbody .selected a:active {
    background-color: #6d8ab7;
    color: white;
    text-shadow: 1px 1px 2px #22456b;
}
```

Vous remarquerez que les dates conservent leur apparence biseautée lors du survol. Si vous souhaitez donner l'impression que la case de la date s'enfonce, changez la couleur des bordures de cellule de manière à assombrir les bordures supérieure et gauche et à éclaircir les bordures inférieure et droite. Notez bien que, comme ce style utilise un pseudo-sélecteur `hover` sur un élément qui n'est pas une ancre, il ne s'affiche pas dans Internet Explorer 6. Pour qu'il s'y affiche, ajoutez des bordures aux liens à la place.

```
.cal tbody td:hover,
.cal tbody td.selected {
    border-top: 1px solid #2a3647;
    border-right: 1px solid #465977;
    border-bottom: 1px solid #576e92;
    border-left: 1px solid #466080;
}
```

Et voilà le travail : vous avez maintenant un superbe sélecteur de date analogue à celui de la Figure 7.3.

Mise en forme simple de formulaires

Les formulaires courts et assez simples sont plus faciles à remplir lorsque les étiquettes apparaissent au-dessus des éléments de formulaire associés. Les utilisateurs peuvent alors descendre progressivement, en lisant chaque étiquette et en complétant l'élément de formulaire qui la suit. Cette méthode fonctionne bien avec les formulaires courts qui collectent des informations relativement simples et prévisibles, comme des informations de contact (voir Figure 7.4).

Figure 7.4

Mise en forme simple de formulaire.

The image shows a screenshot of a web form. It consists of two main sections, each enclosed in a light gray box with a thin black border. The top section is labeled 'Vos coordonnées' and contains three input fields: 'Nom : (Obligatoire)' with a text input box, 'Adresse e-mail :' with a text input box, and 'Adresse Internet :' with a text input box. The bottom section is labeled 'Commentaires' and contains a single input field labeled 'Message : (Obligatoire)' with a large text area for input.

Éléments de formulaire utiles

Le HTML propose un certain nombre d'éléments utiles qui aident à structurer les formulaires et leur confèrent un sens. Le premier est l'élément `fieldset` (ensemble de champs). Il est utilisé pour regrouper des blocs liés d'informations. À la Figure 7.4, deux ensembles de champs sont utilisés : l'un pour les informations de contact et l'autre pour les commentaires. La plupart des agents utilisateurs les encadrent d'une fine bordure, qui peut être désactivée si on attribue la valeur `none` à la propriété `border`.

Un élément `legend` peut être utilisé pour identifier le rôle de chaque ensemble de champs. Les légendes agissent à la manière d'un titre. Elles sont généralement centrées en haut de l'ensemble et légèrement en retrait vers la droite. Malheureusement, elles sont particulièrement difficiles à mettre en forme à cause du manque de cohésion entre les navigateurs pour ce qui concerne leur positionnement. Certains navigateurs, comme Firefox et Safari, utilisent un remplissage pour créer un léger retrait. D'autres, comme Opera et Internet Explorer, ont de grands retraits par défaut qui ne peuvent se contrôler à l'aide du remplissage, des marges ou même du positionnement. Si vous utilisez des légendes, vous devrez donc accepter certaines variations entre navigateurs.

Étiquettes de formulaire

L'élément `label` est extrêmement important parce qu'il contribue à structurer les formulaires et à améliorer leur utilisabilité et leur accessibilité. Il permet d'ajouter une étiquette descriptive à chaque élément de formulaire et, dans de nombreux navigateurs, d'activer l'élément de formulaire associé quand on clique dessus. Le véritable intérêt des étiquettes tient à ce qu'elles rendent les formulaires plus accessibles pour les utilisateurs équipés de périphériques d'assistance. Quand un formulaire les utilise, les lecteurs d'écran associent correctement chaque élément de formulaire à son étiquette. Sans cela, ces périphériques

doivent "deviner" à quel texte se rapporte tel ou tel élément et, parfois, se trompent. Leurs utilisateurs peuvent aussi afficher une liste de toutes les étiquettes du formulaire afin de l'écouter d'un trait comme d'autres balayeraient le formulaire des yeux.

Les étiquettes s'associent très facilement aux contrôles de formulaire. Il existe deux méthodes possibles : soit implicite, dans laquelle l'élément de formulaire est imbriqué dans l'élément `label` :

```
<label>E-mail<input name="email" type="text" /><label>
```

soit explicite, dans laquelle on donne à l'attribut `for` de l'élément `label` la valeur de l'`id` de l'élément de formulaire associé :

```
<label for="email">E-mail<label>
<input name="email" id="email" type="text" />
```

Vous remarquerez que cet élément `input`, comme tous les contrôles de formulaire de ce chapitre, contient à la fois un attribut `id` et un attribut `name`. Le premier est requis pour créer l'association entre le champ de formulaire et l'étiquette, tandis que le second l'est pour retransmettre au serveur les données du formulaire. Ces deux éléments ne doivent pas nécessairement être identiques, même si je préfère, pour ma part, qu'ils le soient chaque fois que c'est possible, par souci de cohérence.

Les étiquettes associées aux contrôles de formulaire qui utilisent l'attribut `for` n'ont pas besoin de jouxter le contrôle dans le code source. Elles peuvent se trouver dans un emplacement complètement différent du document. D'un point de vue structurel, il n'est cependant pas judicieux de séparer les étiquettes des contrôles. Mieux vaut l'éviter si possible.

Disposition de base

Avec ces trois éléments structurels, vous pouvez commencer à disposer votre formulaire en marquant le contenu du premier ensemble de champs. La Figure 7.5 présente le formulaire sans mise en forme.

```
<fieldset>
  <legend>Vos coordonnées</legend>
  <div>
    <label for="author">Nom :</label>
    <input name="author" id="author" type="text" />
  </div>
  <div>
    <label for="email">Adresse e-mail :</label>
    <input name="email" id="email" type="text" />
  </div>
  <div>
    <label for="url">Adresse Internet :</label>
    <input name="url" id="url" type="text" />
  </div>
</fieldset>
```

Figure 7.5

Le formulaire sans mise en forme.

Vos coordonnées

Nom : (Obligatoire)

Adresse e-mail :

Adresse Internet :

Pour commencer, vous allez définir les styles généraux pour l'ensemble de champs et les éléments de légende. Le contenu peut être aéré avec un léger remplissage, tandis que les ensembles de champs doivent être séparés verticalement à l'aide de marges. Pour marquer ceux-ci, vous pouvez leur attribuer un léger arrière-plan avec une bordure d'un pixel légèrement plus sombre. Tâchez cependant de ne pas trop assombrir l'arrière-plan, car une mise en forme très marquée rendrait le formulaire difficile à comprendre. Le fait de passer les légendes en gras peut aussi contribuer à décomposer les informations pour les rendre plus digestes.

```
fieldset {
    margin: 1em 0;
    padding: 1em;
    border: 1px solid #ccc;
    background: #f8f8f8;
}

legend {
    font-weight: bold;
}
```

Les étiquettes peuvent être facilement positionnées verticalement au-dessus des éléments de formulaire. Par défaut, l'étiquette est un élément incorporé. Si vous attribuez la valeur `block` à sa propriété `display`, elle génère cependant sa propre boîte de bloc, ce qui force les éléments `input` à passer à la ligne suivante. La largeur des champs de saisie varie d'un navigateur à l'autre. Par souci d'uniformité, il est donc préférable d'en définir explicitement la largeur. Dans cet exemple, j'utilise des cadratins pour créer une disposition plus simple à réadapter.

```
label {
    display: block;
    cursor: pointer;
}

input {
    width: 20em;
}
```

Il est judicieux de changer le style du curseur de l'étiquette en un pointeur, car on signale ainsi qu'il est possible d'interagir avec.

Autres éléments

Cette structuration fonctionne aussi bien pour les autres éléments de formulaire, comme les zones de texte :

```
<fieldset>
  <legend>Commentaires</legend>
  <div>
    <label for="text">Message : </label>
    <textarea name="text" id="text">
    </textarea>
  </div>
</fieldset>
```

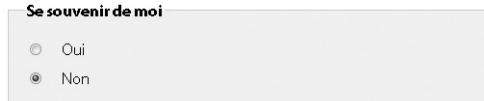
Les dimensions des zones de texte varient également d'un navigateur à l'autre. Il est donc judicieux de leur donner aussi une largeur et une hauteur explicites. Ici, j'ai attribué une largeur de 100 %, afin qu'elle soit définie par son élément parent. Cette manière de définir les largeurs rend les mises en page plus flexibles et plus indépendantes.

```
textarea {
  width: 100%;
  height: 10em;
}
```

Les boutons radio et les cases à cocher doivent être gérés autrement que les zones de texte et les champs de saisie. Leur légende est placée généralement à leur droite. Lorsqu'ils sont empilés, tous les éléments sont alignés à gauche, ce qui crée un bon repère visuel vertical et les rend plus faciles à sélectionner (voir Figure 7.6).

Figure 7.6

Disposition des boutons radio.



Auparavant dans cet exemple, nous avons défini la largeur des zones de texte en appliquant une largeur à l'élément `input`. Cependant, celui-ci peut correspondre à toutes sortes de composants de formulaire, comme les cases à cocher, les boutons radio et les boutons d'envoi, ainsi que les champs de saisie, plus courants. En lui donnant une largeur de 20 cadratins, on donne cette largeur à tous ces éléments de formulaire.

L'une des solutions possibles consiste à utiliser le sélecteur d'attribut pour cibler les différents types d'éléments de formulaire. Au lieu de fixer la largeur des éléments `input` à 20 cadratins, vous pouvez ainsi cibler spécifiquement les éléments `input` de type `text` :

```
input[type="text"] {
  width: 20em;
}
```

Malheureusement, le sélecteur d'attribut n'est pris en charge que par les navigateurs plus récents et ne fonctionne pas sous Internet Explorer 6 et ses versions antérieures. En attendant que la situation change, le meilleur moyen de distinguer les différents éléments `input` consiste à leur attribuer une classe.

Par exemple, vous pouvez donner aux boutons radio le nom de classe `radio` :

```
<fieldset>
  <legend>Se souvenir de moi</legend>
  <div>
    <label for="remember-yes"><input id="remember-yes" class="radio"
      name="remember" type="radio" value="yes" />Oui</label>
  </div>

  <div>
    <label for="remember-no"><input id="remember-no" class="radio"
      name="remember" type="radio" value="no" checked="checked" />Non</label>
  </div>
</fieldset>
```

Vous pouvez ensuite redéfinir la largeur des éléments `input` précédemment définie en fixant celle des boutons radio à `auto`. De même pour les cases à cocher et les boutons d'envoi :

```
input.radio, input.checkbox, input.submit {
  width: auto;
}
```

Vous aurez remarqué à cette occasion de quelle manière j'ai enveloppé les éléments de formulaire avec les étiquettes. Vous vous rappelez que j'ai précédemment configuré toutes les étiquettes dans ce formulaire de manière qu'elles se comportent comme des éléments de niveau bloc, en forçant leurs contrôles de formulaire associé à s'afficher dans une ligne séparée. Il n'est évidemment pas souhaitable d'en faire de même avec les étiquettes des boutons radio. En enveloppant les contrôles de formulaire avec les étiquettes, j'évite ce problème.

La dernière chose à faire est d'ajouter une petite marge droite aux boutons radio, afin de créer un espacement entre les étiquettes.

```
#remember-me .radio {
  margin-right: 1em;
}
```

Embellissements

La disposition de base est maintenant terminée, mais vous pouvez opérer quelques intéressants ajouts pour les navigateurs plus avancés. Par exemple, vous pouvez aider les utilisateurs à mieux repérer le champ de formulaire qu'ils remplissent en modifiant la couleur d'arrière-plan de l'élément lorsqu'il devient actif :

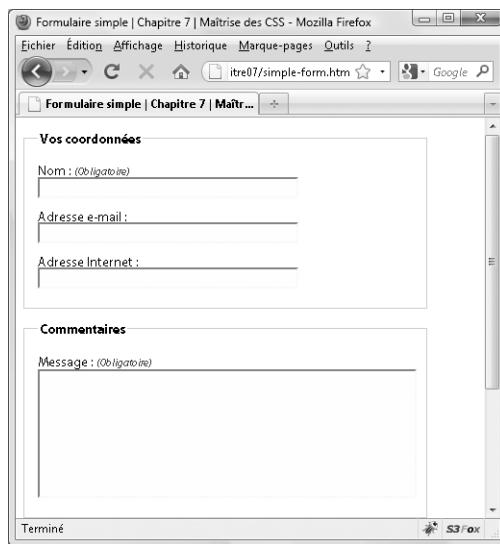
```
input[type="text"]:focus, textarea:focus {
  background: #ffc;
}
```

Vous pouvez aussi harmoniser l'apparence des éléments de champ texte et de zone de texte en leur donnant des bordures personnalisées. C'est particulièrement utile pour Firefox, qui

affiche les bordures du bas et de droite en blanc pour ces éléments, ce qui les fait disparaître sur fond blanc (voir Figure 7.7).

Figure 7.7

Les bordures du bas et de droite des champs de saisie et des zones de texte sont affichées en blanc dans Firefox, ce qui les fait disparaître sur fond blanc.



Ici, nous utilisons un sélecteur d'attribut pour cibler les champs de saisie, car ce style est tout particulièrement destiné à Firefox, qui comprend ce sélecteur.

```
input[type="text"], textarea {
    border-top: 2px solid #999;
    border-left: 2px solid #999;
    border-bottom: 1px solid #ccc;
    border-right: 1px solid #ccc;
}
```

Nous n'avons utilisé aucun champ de mot de passe. Toutefois, si nous devions créer un style de formulaire générique pour le site entier, nous aurions dû aussi inclure [type="password"] dans les deux précédents exemples.

Champs obligatoires

De nombreux formulaires contiennent des champs obligatoires. Vous pouvez les signaler en plaçant un texte mis en forme ou un astérisque à côté. Comme ces informations signalent le statut obligatoire du champ, l'élément le plus approprié pour ces informations est l'élément em ou l'élément strong :

```
<div>
<label for="author">Nom :<em class="required">(obligatoire)</em>/label>
<input name="author" id="author" type="text" />
</div>
```

Vous pouvez aussi appliquer la mise en forme de votre choix à ces informations. Dans cet exemple, j'ai réduit la taille de police et passé le texte en rouge :

```
.required {
```

```

font-size: 0.75em;
color:#760000;
}

```

Et voilà le travail : une mise en forme de formulaire simple et attrayante entièrement réalisée en CSS.

Mise en forme complexe de formulaire

Parcourir en un clin d'œil les formulaires plus longs et plus compliqués est difficile, et l'espace vertical commence à devenir un problème. Pour améliorer leur appréhension immédiate et réduire la quantité d'espace vertical utilisé, il est judicieux de positionner les étiquettes et les éléments de formulaire horizontalement au lieu de les empiler. Il est en fait très simple de créer un formulaire comme celui de la Figure 7.8, qui utilise exactement le même code que celui du précédent exemple.

Figure 7.8

Alignement horizontal du formulaire.

The form consists of three horizontal sections. The first section, 'Vos coordonnées', contains three input fields: 'Nom : (Obligatoire)' with a text input, 'Adresse e-mail:' with a text input, and 'Adresse Internet:' with a text input. The second section, 'Commentaires', contains a label 'Message : (Obligatoire)' and a large text area for input. The third section, 'Se souvenir de moi', contains two radio buttons: 'Oui' and 'Non', with 'Oui' being the selected option. A 'Valider' button is located at the bottom left of the form.

La seule différence entre cet exemple et le précédent tient à ce que, au lieu de définir les étiquettes comme un élément de niveau bloc, il suffit de les faire flotter à gauche. Vous devez en outre fixer leur largeur afin que tous les éléments s'alignent harmonieusement :

```

label {
  float: left;
  width: 10em;
  cursor: pointer;
}

```

Si les étiquettes de formulaire doivent s'étendre sur plusieurs lignes, vous avez intérêt à dégager les div conteneurs également. Vous les empêcherez ainsi d'interférer avec la prochaine série d'étiquettes et de rompre cette harmonie si bien préparée :

```
form div {
  clear: left;
}
```

Les formulaires sont rarement aussi simples que celui de la Figure 7.8. Bien souvent, vous devrez créer des exceptions à vos règles de mise en forme de base afin de gérer les composants multiples sur une même ligne ou les colonnes de cases à cocher et de boutons radio (voir Figure 7.9). Les deux sections suivantes expliquent comment traiter ces types d'exceptions.

Figure 7.9

Une mise en forme de formulaire plus complexe.

The screenshot shows a 'Personnel Information' form with the following fields:

- Lieu de naissance :** A dropdown menu set to "Etats-Unis".
- Date de naissance :** A date input field showing "24 Mars 1972".
- Couleur favorite :** A list of color names with checkboxes:
 - rouge (unchecked)
 - orange (unchecked)
 - jaune (unchecked)
 - violet (unchecked)
 - rose (unchecked)
 - bleu (unchecked)
 - vert (unchecked)
 - autre (unchecked)

Champ de date accessible

Comme vous venez de le voir, les étiquettes jouent un rôle important dans l'accessibilité des formulaires. Il arrivera pourtant parfois que vous ne souhaitiez pas afficher d'étiquette pour chaque élément. La Figure 7.9 présente ainsi un groupe d'éléments de formulaire qui sert à collecter des informations de date. Ici, il serait excessif d'afficher chaque étiquette, car la date de naissance se trouverait décomposée en trois entités séparées au lieu d'être appréhendée comme une entité unique. Quand on ne veut pas afficher les étiquettes, il peut malgré tout être utile de les faire figurer dans le code source et de les rendre ainsi disponibles pour les lecteurs d'écran.

```
<div>
  <label for="dateOfBirth">Date de naissance :</label>
  <input name="dateOfBirth" id="dateOfBirth" type="text" />
  <label id="monthOfBirthLabel" for="monthOfBirth">Mois de naissance :</label>
  <select name="monthOfBirth" id="monthOfBirth">
    <option value="1">Janvier</option>
    <option value="2">Février</option>
    <option value="3">Mars</option>
  </select>
  <label id="yearOfBirthLabel" for="yearOfBirth">Année de naissance :</label>
  <input name="yearOfBirth" id="yearOfBirth" type="text" />
</div>
```

Pour créer cette disposition, vous devez d'abord masquer les étiquettes "Mois de naissance" (monthOfBirth) et "Année de naissance" (yearOfBirth). Si vous leur attribuez la valeur `none` à leur propriété `display`, vous les empêcheriez de s'afficher, mais vous empêcheriez aussi les lecteurs d'écran d'y accéder. Au lieu de cela, vous pouvez les positionner hors écran avec un retrait de texte négatif important. Dans le style de formulaire générique que

nous avons créé précédemment, on définissait la largeur des étiquettes. Pour empêcher ces dernières d'affecter la mise en forme, il faut aussi ramener à zéro leur largeur :

```
#monthOfBirthLabel, #yearOfBirthLabel {
    text-indent: -1000em;
    width: 0;
}
```

Les différents contrôles de formulaire peuvent être redimensionnés individuellement et recevoir des marges pour contrôler leur espacement horizontal :

```
input#dateOfBirth {
    width: 3em;
    margin-right: 0.5em;
}

select#monthOfBirth {
    width: 10em;
    margin-right: 0.5em;
}

input#yearOfBirth {
    width: 5em;
}
```

Cases à cocher multicolonnes

Créer une disposition à deux colonnes pour les groupes importants de cases à cocher ou de boutons radio est un petit peu plus compliqué. Les étiquettes ne fonctionnent que pour les éléments individuels et non pour les groupes d'éléments. Idéalement, il faudrait envelopper l'ensemble du groupe dans un ensemble de champs et utiliser la légende comme étiquette du groupe. Malheureusement, cette solution n'est pas pratique pour l'instant, car les navigateurs ne gèrent pas uniformément le positionnement des légendes. En attendant que cela soit le cas, le mieux est d'utiliser un élément de titre à la place.

Pour créer l'effet de colonne, les cases à cocher sont divisées en deux ensembles et chacun d'eux est enveloppé dans une div à laquelle est attribuée la classe `col`. Ces éléments sont ensuite regroupés, enveloppés dans un ensemble de champs avec un ID descriptif :

```
<fieldset id="favoriteColor">
    <h2>Favorite Color:</h2>
    <div class="col">
        <div>
            <label><input class="checkbox" id="red" name="red" type="checkbox" value="red" />rouge</label>
            ...
        </div>
    </div>

    <div class="col">
```

```

<div>
  <label><input class="checkbox" id="orange" name="orange" type="checkbox"
    value="orange" />orange</label>
</div>
...
</div>
</fieldset>

```

Comme nous avons déjà créé un style d'ensemble de champs générique, la première chose à faire est de redéfinir ces styles, de ramener à zéro le remplissage et les marges, de supprimer les bordures et de définir une couleur d'arrière-plan transparente :

```

fieldset#favoriteColor {
  margin: 0;
  padding: 0;
  border: none;
  background: transparent;
}

```

Le titre doit agir à la manière d'une étiquette. Il doit donc flotter à gauche et avoir une largeur de 10 cadratins comme les autres étiquettes. Il doit aussi ressembler à une étiquette. Le style de police doit être normal et la taille de police réduite.

```

#favoriteColor h2 {
  width: 10em;
  float: left;
  font-size: 1em;
  font-weight: normal;
}

```

La disposition en deux colonnes peut ensuite être créée en donnant une largeur aux div et en les faisant flotter à gauche. Toutefois, comme toutes les div dans ce formulaire ont été dégagées à gauche, il faut redéfinir cette déclaration en utilisant `clear:none`.

```

#favoriteColor .col {
  width: 8em;
  float: left;
  clear: none;
}

```

Toutes les étiquettes dans ce formulaire flottent à gauche et sont définies avec une largeur de 10 cadratins. Les étiquettes pour les cases à cocher n'ont cependant pas besoin d'être flottantes. Cette déclaration peut ainsi être redéfinie ici.

```

#favoriteColor label {
  float: none;
}

```

Nous venons de réaliser une mise en forme de formulaire relativement complexe. Le style de formulaire simple se charge de la mise en forme générale et les exceptions sont gérées de manière individuelle par redéfinition des styles.

Boutons d'envoi

Les formulaires sont un excellent moyen de rendre un site interactif et de renvoyer des données au serveur. Pour les actionner, vous aurez besoin d'un contrôle de bouton. Normalement, les utilisateurs se servent d'un élément `input` dont l'attribut `type` possède la valeur `submit`. Les boutons d'envoi sont le moyen le plus courant de transmettre des données au serveur, mais ils ne sont pas sans poser de problème, notamment parce qu'ils ne peuvent être ciblés avec un simple sélecteur d'élément. Ils peuvent être ciblés avec un sélecteur d'attribut, mais celui-ci n'est pas pris en charge par les anciennes versions d'Internet Explorer. La seule possibilité qui reste est donc de les cibler directement avec un sélecteur d'ID ou de classe. Au lieu d'utiliser un élément `input`, pourquoi ne pas utiliser l'élément `button` ?

L'élément `button` s'est récemment acquis une certaine popularité, mais il reste toujours relativement méconnu et sous-utilisé. C'est un tort, car il offre une grande flexibilité. Pour commencer, vous pouvez utiliser des balises `button` pour englober une image et la transformer ainsi en contrôle (voir Figure 7.10).

```
<div>
<button type="submit">

</button>
</div>
```

Figure 7.10

Un élément `button` qui utilise une image de bouton.



Les boutons ont une mise en forme par défaut. Commencez par la désactiver.

```
button {
  border: none;
  background: none;
  cursor: pointer;
}
```

De nombreux systèmes d'exploitation, comme OS X, empêchent les auteurs de modifier le style de leurs éléments `input` de type `submit`, afin de préserver l'homogénéité d'affichage de l'ensemble du système. Mais l'élément `button` ne souffre pas de ces restrictions. On peut ainsi créer des styles de bouton avancés entièrement en CSS. Par exemple, supposons que vous commençiez par ce simple bouton d'envoi :

```
<p>
<button type="submit">Book Now</button>
</p>
```

Vous pouvez commencer par lui donner des dimensions explicites et une bordure colorée. Vous pouvez ensuite arrondir les bords avec la propriété `border-radius` et appliquer une élégante ombre de texte. Pour finir, vous pouvez appliquer un arrière-plan dégradé, soit

avec une image, soit avec les dégradés WebKit. Le résultat de cet exemple est présenté à la Figure 7.11.

```
button.two {
    width: 200px;
    height: 50px;
    border: 1px solid #989898;
    -moz-border-radius: 6px;
    -webkit-border-radius: 6px;
    border-radius: 6px;
    background: url(/img/button-bg.png) #c5e063 bottom left repeat-x;
    -moz-box-shadow: 2px 2px 2px #ccc;
    -webkit-box-shadow: 2px 2px 2px #ccc;
    box-shadow: 2px 2px 2px #ccc;
    color: #fff;
    font-size: 26px;

    font-weight: bold;
    text-shadow: 1px 1px 1px #666;
}
```

Figure 7.11

Un élément de bouton entièrement réalisé en CSS.

Book Now »

La principale limite concernant les éléments button tient à la manière dont Internet Explorer 6 et, dans une moindre mesure, Internet Explorer 7 gèrent leur envoi. Au lieu d'envoyer le contenu de l'attribut `value` comme le font les autres navigateurs, ils envoient le contenu de l'élément. En outre, si la page contient plusieurs boutons, Internet Explorer 6 transmet le contenu de tous les boutons au lieu de se limiter à celui sur lequel l'utilisateur a cliqué. Si vous souhaitez utiliser plusieurs boutons par page, assurez-vous donc qu'ils ont tous la même fonction, car il n'est pas possible de savoir lequel a été activé dans les anciennes versions d'Internet Explorer.

Retour de formulaire

Les formulaires requièrent généralement un message informatif qui signale les champs qui ont été omis ou mal remplis. Bien souvent, on affiche un message d'erreur à côté du champ approprié (voir Figure 7.12).

Figure 7.12

Exemple de retour de formulaire.

Vos coordonnées

Nom : (Obligatoire)	<input type="text"/>
Adresse e-mail :	<input type="text"/> <small>⚠ Adresse e-mail incorrecte. Essayez à nouveau.</small>
Adresse Internet :	<input type="text"/>

Pour réaliser cet effet, vous pouvez envelopper le texte de retour dans une balise `em` et la placer après le champ de saisie dans le code source. Pour que tout s'aligne correctement,

il faut cependant que la balise `em` et le champ de saisie qui la précède soient flottants. Cela a un effet sur le comportement du paragraphe enclos, qui à son tour a un effet sur la mise en forme globale. En outre, bien des lecteurs d'écran ignorent le texte entre les éléments de formulaire, à moins qu'il ne soit englobé dans une étiquette. Pour éviter ces problèmes, la meilleure solution consiste à inclure le texte du message d'erreur dans l'étiquette de formulaire, puis à le positionner à l'aide des CSS :

```
<div>
  <label for="email">Adresse e-mail :
  <em class="feedback">Adresse e-mail incorrecte. Essayez à nouveau. </em>
  </label>
  <input name="email" id="email" type="text" />
</div>
```

Pour positionner l'élément `em` du message informatif, vous devez attribuer la valeur `relative` à la propriété `position` de tous les paragraphes dans le formulaire, afin de fournir un nouveau contexte de positionnement. Vous pouvez ensuite positionner le message de manière absolue, afin qu'il apparaisse à la droite du champ de saisie. Comme les étiquettes font 10 cadratins de large et que les champs de saisie en font 20, la position gauche de l'élément de message informatif peut être fixée à 30 cadratins.

```
form div {
  position: relative;
}

form .feedback {
  position: absolute;
  left: 30em;
  right: 0;
  top: 0.5em;
}
```

Malheureusement, Internet Explorer 6 et ses versions antérieures ne définissent pas correctement la largeur du message (classe `feedback`) en la rendant aussi petite que possible. Pour résoudre ce problème, vous devez définir une largeur explicite pour ce navigateur. L'une des solutions consiste à utiliser des commentaires conditionnels, comme indiqué au Chapitre 8 :

```
form .feedback{
  width: 10em;
}
```

Vous pouvez ensuite appliquer la mise en forme de votre choix pour vos messages. Ici, j'ai passé le texte en rouge et en gras, avec une image d'avertissement à gauche du message :

```
form div {
  position: relative;
}

.feedback {
  position: absolute;
  left: 30em;
```

```
right :0;
top: 0.5em;
font-weight: bold;
color: #760000;
padding-left: 18px;
background: url(/img/error.png) no-repeat left top;
}
```

Vous pouvez aussi utiliser cette technique pour des messages de réussite ou des conseils sur la manière de remplir des portions particulières du formulaire.

En résumé

Vous avez vu comment différentes mises en page de formulaire pouvaient convenir à différentes situations. Vous savez maintenant mettre en forme des formulaires complexes en CSS, sans utiliser le moindre tableau. Vous avez vu comment utiliser des tableaux (pour afficher des données et non pour structurer la mise en page) et avez constaté que cette mise en page pouvait ne pas être si compliquée.

Au chapitre suivant, vous exploitez tout ce que vous avez appris jusque-là pour créer des mises en page CSS.

8

Mise en page

L'un des principaux intérêts des CSS tient à ce qu'elles permettent de contrôler la mise en page sans utiliser de balise de présentation dans le code HTML. Les mises en page CSS ont cependant la réputation d'être difficiles, notamment pour les débutants. Cette réputation est due, en partie, au manque d'uniformité dans le traitement des navigateurs, mais surtout à la prolifération des différentes techniques de mise en page disponibles sur le Web. Chaque auteur CSS semble avoir adopté sa manière de faire pour les mises en page multicolonnes et les nouveaux développeurs CSS reprennent bien souvent des techniques sans vraiment comprendre ce qu'ils font. Cette situation a empiré avec l'arrivée des *frameworks CSS*, qui visent à faciliter la mise en page CSS en créant un couplage étroit entre le balisage et la présentation – le défaut même qui nous avait pourtant poussés à abandonner les mises en page à tableaux. Cette façon de traiter les CSS comme une boîte noire peut avoir son efficacité s'il s'agit d'obtenir des résultats rapides, mais elle empêche, en fin de compte, le développeur de comprendre le langage et d'appliquer des modifications.

Toutes ces techniques de mise en page CSS s'appuient sur trois concepts élémentaires : le positionnement, le flottement et la manipulation des marges. Ces techniques ne diffèrent pas tant que cela et, si vous saisissez les concepts essentiels, vous créerez assez facilement vos propres mises en page. En fait, la mise en page est bien souvent la partie la plus simple des CSS. Ce sont toutes les opérations d'ajustement qui prennent du temps.

Au cours de ce chapitre, vous apprendrez à :

- centrer horizontalement une mise en page dans le navigateur ;
- créer des mises en page à deux ou trois colonnes flottantes ;
- créer des mises en page à largeur fixe, liquides et élastiques ;
- créer des colonnes de hauteurs égales ;
- comparer les frameworks CSS et les systèmes CSS.

Planification de la mise en page

Au moment de transformer les maquettes graphiques en pages entièrement fonctionnelles, il peut être très tentant de se jeter directement à l'eau et de commencer à placer des balises dans une page ou à découper des images. On se retrouve cependant vite coincé. Il est préférable, en réalité, de prendre quelques instants pour se charger de la planification afin d'éviter bien des soucis par la suite. C'est le dicton des couturières : "On peut mesurer deux fois, mais on ne coupe qu'une seule fois."

La première étape pour créer un système CSS évolutif et facile à gérer consiste à examiner les maquettes graphiques afin d'y rechercher des motifs qui se répètent. Il peut s'agir

de motifs dans la structure de la page ou de la manière dont certains éléments se répètent dans l'ensemble du site. La représentation visuelle ne doit pas encore particulièrement vous préoccuper à ce stade. Intéressez-vous plutôt à la structure et à la signification. Pour ma part, j'imprime généralement chaque maquette de page et je repère les motifs puis j'annote chaque page (voir Figure 8.1). J'ai aussi vu certaines personnes annoter directement leurs fichiers Photoshop ou leurs structures en blocs.

Figure 8.1

Guides pour le balisage.



Commencez par décomposer vos pages en leurs principales zones structurelles, comme la zone conteneur, l'en-tête, la zone de contenu et le pied de page. Ces zones restent généralement uniformes dans l'ensemble du site et ne changent que rarement. Il s'agit en quelque sorte des murs extérieurs de votre bâtiment.

Ensuite, portez votre attention sur la zone de contenu elle-même et commencez à construire la structure de grille. Combien de zones de contenu différentes y a-t-il et en quoi diffèrent-elles ? Les zones de contenu sont-elles si différentes ou peuvent-elles être traitées de la même manière du point de vue de la mise en page ? La plupart des mises en page ne contiennent que deux zones de contenu uniques : concentrez-vous donc plutôt sur les points communs que sur les différences visuelles. Vous pouvez considérer ces zones de contenu comme les murs porteurs à l'intérieur de votre bâtiment.

Pour finir, examinez les différentes structures de mise en page qui apparaissent dans les différentes zones de contenu. Devez-vous présenter certains types d'informations dans deux, trois ou quatre colonnes ? À la différence des précédentes, ces structures de mise en page tendent à être très flexibles et changent de page en page. Vous pouvez les considérer comme les murs de cloisonnement. Combinés avec l'étape précédente, ils aident à former le plan des salles d'un étage de l'immeuble. À ce stade, vous pouvez vous saisir du crayon et des marqueurs et commencer à matérialiser plus précisément la structure et les dimensions (voir Figure 8.2).

Figure 8.2

Calcul des dimensions sur du papier à carreaux.



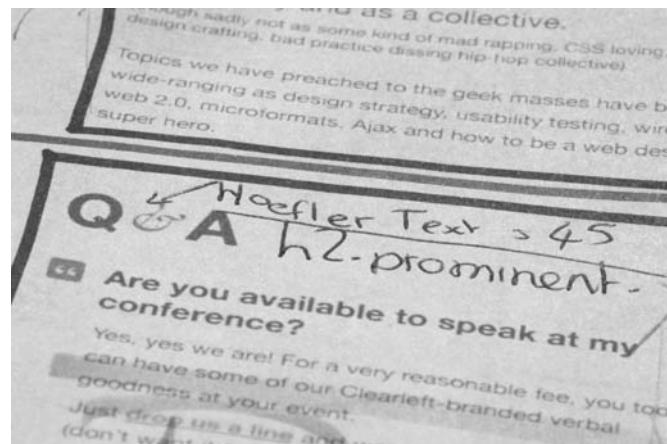
Une fois la structure en place, vous pouvez tourner votre attention vers les différents types de contenu. S'agit-il d'un article de news, d'un document ou d'un communiqué de presse ? Donnez à chaque bloc un nom descriptif et voyez quels rapports les uns entretiennent avec les autres. Il se pourrait que seules des différences minimales distinguent les articles de news des communiqués de presse, auquel cas, il serait judicieux de les combiner comme un seul type de contenu.

Voyez comment chaque bloc de contenu est structuré et vérifiez si des motifs se remarquent entre les différents types. Par exemple, vous remarquerez que les articles et les brèves d'actualité utilisent tous deux un titre bien marqué et un pied de page. Identifiez-les comme tels. Peu importe que les titres et les pieds de page possèdent une apparence différente : vous pourrez parfaitement les mettre en forme différemment selon le contexte. Il en va de même pour les messages d'erreur, les champs de recherche et les éléments de menu. Essayez de vous en tenir à des noms de classes aussi génériques que possible et mettez-les en forme en fonction du contexte.

Une fois que les motifs et les conventions de noms sont triés, vous pouvez commencer à définir les éléments que vous utiliserez. Par exemple, une liste de liens peut être une liste à puces, alors qu'un article correspondra à une div avec un élément h2, un paragraphe et un élément d'ancre. Il est bien plus facile de prévoir cela, avec quelques collaborateurs, que de le faire à la volée. J'ai aussi remarqué qu'il était utile de noter les codes de couleur, les dimensions et tous les autres renseignements nécessaires en cours de production. Vous pouvez cette fois encore placer ces annotations sur une impression papier de vos mises en page (voir Figure 8.3).

Figure 8.3

Les caractéristiques de détail sont définies pour les différents types de contenu.



Les premières fondations

Supposons que nous souhaitions créer un modèle de blog classique à trois colonnes, comme celui de la Figure 8.4.

Figure 8.4

Mise en page classique à trois colonnes.

Si on analyse la structure de page, il apparaît clairement qu'il faudra un élément conteneur pour centrer la mise en page, ainsi qu'un en-tête, une zone de contenu et un pied de page. Le code HTML ressemblera donc à ceci :

```
<body>
  <div class="wrapper">

    <div class="header">
      <!-- Le contenu de votre en-tête vient ici -->
    </div>

    <div class="content">
      <!-- Le contenu de votre page vient ici -->
    </div>

    <div class="footer">
      <!-- Le contenu de votre pied de page vient ici -->
    </div>

  </div>
</body>
```

Comme trois zones au moins sont encloses dans le conteneur (wrapper), commençons par appliquer les styles à celui-ci.

Centrage d'une mise en page à l'aide des marges

Les longues lignes de texte peuvent être difficiles à lire et déplaisantes à l'œil. Les moniteurs modernes ne cessant de s'agrandir, la question de la lisibilité à l'écran devient de plus en plus importante. L'un des moyens par lesquels les concepteurs ont tenté de résoudre ce problème consiste à centrer la mise en page. Au lieu de couvrir toute la largeur de l'écran, les mises en page centrées n'en occupent qu'une partie, ce qui crée des lignes plus courtes et plus faciles à lire.

Supposons que vous utilisez une mise en page classique où vous souhaitez centrer la div conteneur horizontalement à l'écran :

```
<body>
  <div class="wrapper">
    </div>
  </body>
```

Pour cela, vous devez simplement définir sa largeur et attribuer la valeur `auto` à ses marges horizontales :

```
.wrapper {
  width: 920px;
  margin: 0 auto;
}
```

Dans cet exemple, j'ai décidé de fixer la largeur de la div conteneur en pixels, afin qu'elle tienne parfaitement dans un écran de 1 024 × 768 pixels. Vous pouvez cependant tout aussi aisément définir la largeur sous la forme d'un pourcentage de l'élément body ou par rapport à la taille du texte en utilisant des cadratins (em).

Cette technique fonctionne sur tous les navigateurs modernes. Internet Explorer 5.x et Internet Explorer 6 en mode Quirks n'interprètent pas la déclaration `margin: auto`. Par chance, Internet Explorer interprète `text-align: center` étrangement, en centrant tous les éléments et pas seulement le texte. Vous pouvez donc exploiter cette bizarrerie en centrant tout dans la balise `body`, et notamment la div conteneur, puis en réalignant le contenu du conteneur à gauche :

```
body {  
    text-align: center;  
}  
  
.wrapper {  
    width: 920px;  
    margin: 0 auto;  
    text-align: left;  
}
```

Cette manière d'utiliser la propriété `text-align` est un *hack* (un bidouillage en CSS), mais elle est relativement inoffensive car elle n'a pas d'impact négatif sur le site. Le conteneur apparaît maintenant centré dans les versions plus anciennes d'Internet Explorer comme dans les navigateurs modernes compatibles avec les standards (voir Figure 8.5).

Figure 8.5

Centrage d'une mise en page avec `margin: auto`.



Mises en page flottantes

Il existe plusieurs moyens de créer des mises en page CSS, notamment avec un positionnement absolu et des marges négatives. La méthode des mises en page flottantes est cependant la plus simple et la plus fiable. Elle consiste à définir la largeur des éléments à positionner, puis à les faire flotter à gauche ou à droite.

Comme les éléments flottants ne prennent pas d'espace dans le flux du document, ils ne semblent pas exercer d'influence sur les boîtes de bloc alentour. Pour résoudre ce problème, vous devez dégager les éléments flottants à différents emplacements de la mise en page. Au lieu de faire continuellement flotter les éléments et de les dégager, on peut aussi tout faire flotter puis dégager un ou deux éléments à des emplacements stratégiques du document, comme au niveau du pied de page. Sinon, il est possible d'utiliser la propriété `overflow` pour dégager le contenu d'éléments particuliers. C'est ma méthode de choix et c'est donc celle que j'adopterai dans le reste des exemples.

Mise en page flottante à deux colonnes

Pour créer une mise en page à deux colonnes dans la zone de contenu, vous devez d'abord créer la structure HTML de base.

```
<div class="content">

  <div class="primary">
    <!-- Le contenu principal vient ici -->
  </div>

  <div class="secondary">
    <!-- La navigation et le contenu secondaire viennent ici -->
  </div>

</div>
```

La zone de contenu secondaire pour cette mise en page (qui inclut le système de navigation du site) doit apparaître à gauche de la page, le contenu principal étant pour sa part placé à droite. Toutefois, j'ai choisi de placer la zone de contenu principal au-dessus de la zone de contenu secondaire dans l'ordre du code source, pour des raisons liées à l'utilisabilité et à l'accessibilité. Tout d'abord, le contenu principal est la partie la plus importante de la page et doit donc venir en premier dans le document. Ensuite, tant que ce n'est pas nécessaire, il est inutile de forcer les utilisateurs de lecteurs d'écran à parcourir tous les liens de navigation et le contenu moins important, comme les suggestions de sites, avant qu'ils n'aient atteint le contenu principal.

Normalement, lorsque les utilisateurs créent des mises en page flottantes, ils font flotter les deux colonnes à gauche et, entre elles, ils créent une gouttière à l'aide des propriétés de marge et de remplissage. Avec cette méthode, les colonnes sont étroitement liées dans l'espace disponible, sans intervalle qui les sépare. Ce ne serait pas vraiment un problème si les navigateurs savaient se tenir, mais les bogues de quelques-uns d'entre eux peuvent dénaturer

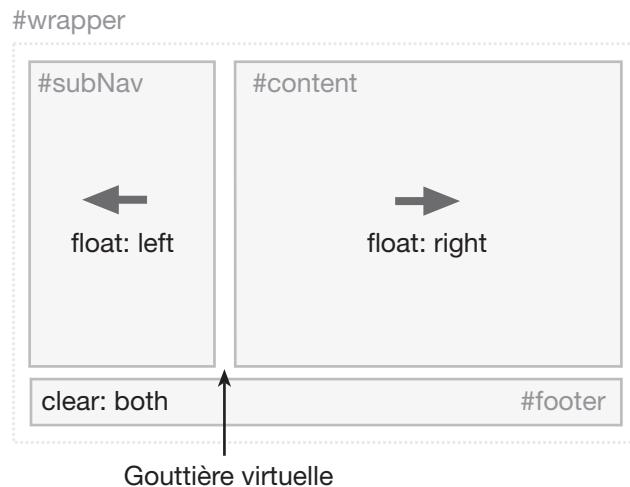
les mises en page cousues au pixel près et contraindre les colonnes à s'empiler les unes sur les autres au lieu de se serrer côte à côte.

Ce phénomène peut surgir dans Internet Explorer, qui respecte la taille du contenu des éléments plus que celle des éléments eux-mêmes. Dans les navigateurs conformes aux standards, si le contenu d'un élément devient trop grand, il s'étend tout simplement hors de la boîte. Sous Internet Explorer, dans un tel cas, c'est l'élément tout entier qui s'étend. Cet effet indésirable peut être provoqué par les plus infimes détails, comme une portion de texte en italique. Quand ce problème survient dans une mise en page ciselée au millimètre, il ne reste plus d'espace pour que les éléments restent côte à côte et l'un d'entre eux finit par disparaître en dessous. D'autres bogues d'Internet Explorer, comme celui du décalage de texte de 3 pixels et celui de la double marge des éléments flottants (voir le Chapitre 9), ainsi que différentes erreurs d'arrondi des navigateurs peuvent provoquer l'effondrement des mises en page flottantes.

Pour empêcher vos mises en page de se défaire, vous devez éviter de les engoncer à l'intérieur de leurs éléments conteneurs. Au lieu d'utiliser des marges et des remplissages horizontaux pour créer des gouttières, créez une gouttière virtuelle en faisant flotter un élément à gauche et un autre à droite (voir Figure 8.6). Si la taille d'un élément augmentait accidentellement de quelques pixels, il grignoterait un bout de la gouttière virtuelle au lieu de manquer d'espace et de glisser sous l'élément précédent.

Figure 8.6

Une mise en page à deux colonnes avec des éléments flottants.



Le code CSS pour cette mise en page est très simple. Vous devez fixer la largeur désirée de chaque colonne puis faire flotter le contenu secondaire à gauche et le contenu principal à droite. Vous devez aussi ajouter un léger remplissage au contenu principal afin d'empêcher que le texte enclos vienne toucher le bord droit de l'élément. Vous remarquerez que j'ai aussi ajouté `display: inline` à tous les éléments flottants. Il s'agit d'une mesure défensive contre le bogue de la double marge des éléments flottants d'Internet Explorer (nous reviendrons sur ce point au chapitre suivant).

```
.content .primary {  
    width: 650px;  
    padding-right: 20px;  
    float: right;  
    display: inline;  
}  
  
.content .secondary {  
    width: 230px;  
    float: left;  
    display: inline;  
}
```

Comme la largeur totale disponible est de 920 pixels, on obtient une gouttière virtuelle de 20 pixels de large entre les éléments flottants. Ainsi, la mise en page ne s'effondrera pas si le contenu déborde de manière accidentelle.

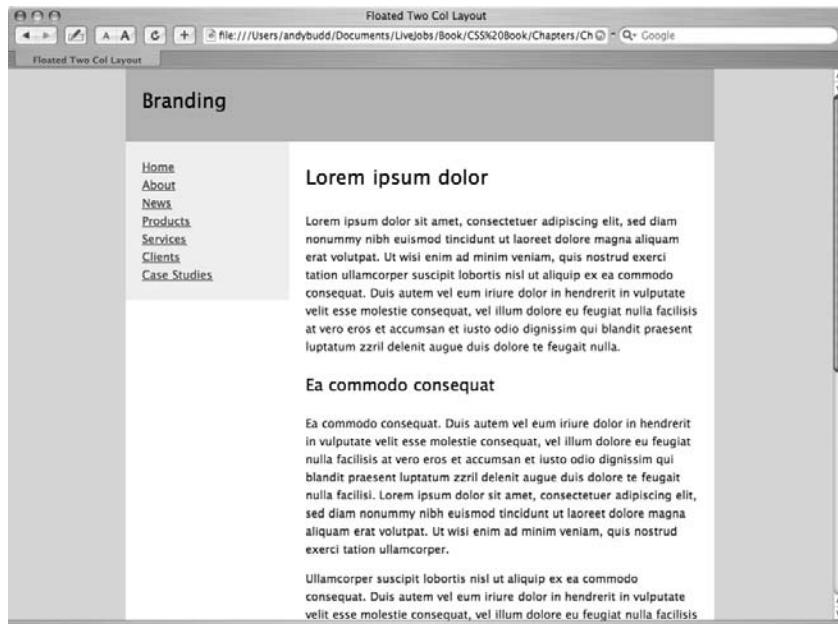
Comme ces éléments sont flottants, ils ne prennent plus d'espace dans le flot du document et le pied de page remonte tout en haut. Pour éviter cela, vous devez dégager les éléments flottants en appliquant la méthode `overflow` à leur élément parent soit, ici, la div `content`.

```
.content {  
    overflow: hidden;  
}
```

Et voilà le travail : une mise en page CSS simple à deux colonnes (voir Figure 8.7).

Figure 8.7

Mise en page flottante à deux colonnes.



Vous remarquerez qu'au lieu de créer deux éléments séparés appelés `primary-content` et `secondary-content`, j'ai simplement utilisé les termes `primary` et `secondary`. Je me suis ensuite servi du fait que les deux éléments sont imbriqués dans l'élément `content` pour créer l'association. Cette méthode a plusieurs avantages. D'une part, vous n'avez pas à créer des noms de classes pour chacun des éléments que vous souhaitez mettre en forme. Au lieu de cela, vous pouvez tirer parti de la cascade. D'autre part, et c'est un argument bien plus important, vous pouvez utiliser les mêmes classes `primary` et `secondary` plusieurs fois, en créant un système de nommage flexible. Par exemple, supposons que vous souhaitez créer une mise en page à trois colonnes au lieu de deux.

Mise en page flottante à trois colonnes

Le code HTML requis pour créer une mise en page à trois colonnes est très proche de celui utilisé pour la mise en page à deux colonnes. La seule différence tient à l'ajout de deux nouvelles div à l'intérieur de la div de contenu principale : l'une pour le contenu principal et l'autre pour le contenu secondaire. Les noms de classes flexibles `primary` et `secondary` peuvent ainsi être utilisés à nouveau.

```
<div class="content">

    <div class="primary">
        <div class="primary">
            <!-- Votre contenu principal vient ici -->
        </div>
        <div class="secondary">
            <!-- Votre contenu secondaire vient ici -->
        </div>
    </div>

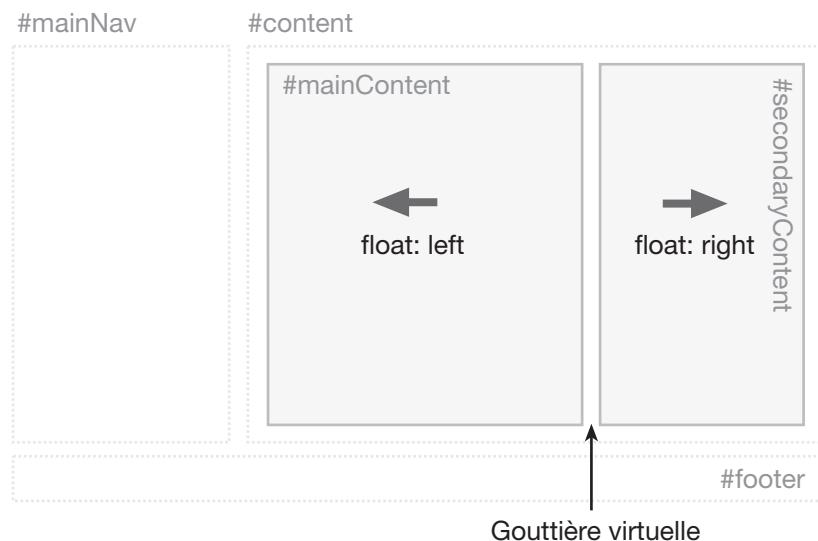
    <div class="secondary">
        <!-- La navigation et le contenu secondaire viennent ici -->
    </div>

</div>
```

Avec le même code CSS que pour la technique à deux colonnes, vous pouvez faire flotter le contenu secondaire à gauche et le contenu principal à droite. Ensuite, à l'intérieur de la div de contenu principal, vous pouvez faire flotter la div principale à gauche et la div secondaire à droite (voir Figure 8.8). Cette opération divise en fait la zone de contenu principal en deux sous-zones, ce qui crée une mise en page à trois colonnes.

Figure 8.8

Nous créons la mise en page à trois colonnes en divisant la colonne de contenu en deux colonnes.



Comme auparavant, le code CSS est très simple. Vous devez simplement fixer les largeurs désirées puis faire flotter la div principale à gauche et la div secondaire à droite, en créant un interstice de 20 pixels au milieu :

```
.content .primary .primary {
  width: 400px;
  float: left;
  display: inline;
}

.content .primary .secondary {
  width: 230px;
  float: right;
  display: inline;
}
```

Vous remarquerez que le remplissage côté droit que nous avons attribué à la div principale dans le premier exemple est maintenant appliqué à la nouvelle div principale dans le second exemple. Nous devons donc supprimer le remplissage du style plus général et l'appliquer directement au style spécifique.

```
.content .primary {
  width: 670px; /* largeur augmentée et remplissage supprimé */
  float: right;
  display: inline;
}

.content .secondary {
  width: 230px;
  float: left;
```

```

display: inline;
}

.content .primary .primary {
width: 400px;
float: left;
display: inline;
}

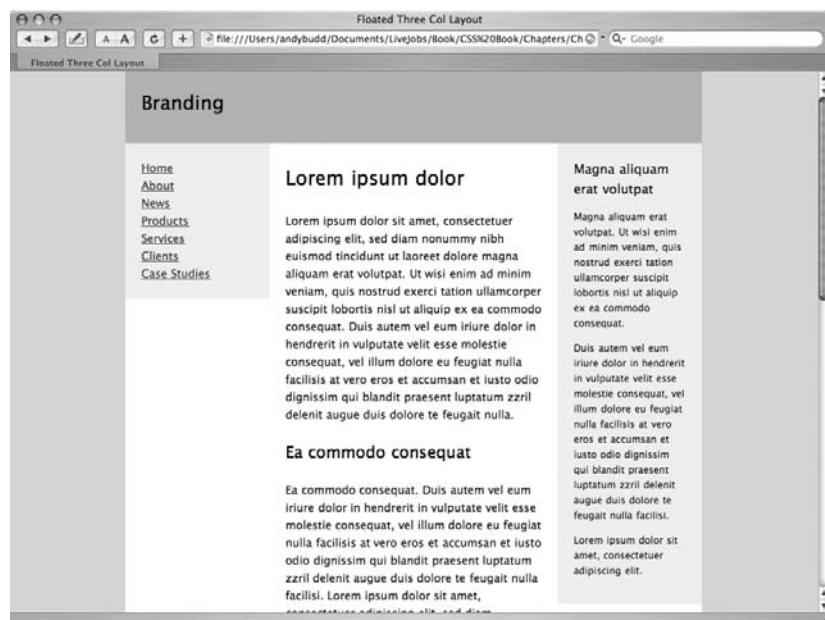
.content .primary .secondary {
width: 230px;
padding-right: 20px; /* le remplissage est appliqué ici à la place */
float: right;
display: inline;
}

```

Voilà qui produit une élégante et robuste mise en page à trois colonnes (voir Figure 8.9).

Figure 8.9

Mise en page à trois colonnes réalisée à l'aide d'éléments flottants.



Mises en page à largeur fixe, liquide et élastique

Jusqu'ici, tous les exemples utilisaient des largeurs définies en pixels. Ce type de mise en page, dit *à largeur fixe*, est très courant, car il offre au développeur un contrôle complet sur la mise en page et le positionnement. Si vous fixez la largeur de la maquette à 960 pixels de large, elle fera toujours 960 pixels de large. Si vous souhaitez ensuite placer un logo en haut de la page, vous savez qu'il devra faire 960 pixels pour y tenir. Le fait de connaître la largeur exacte de chaque élément permet de les disposer précisément et de savoir où chaque partie

vient se placer. La méthode de la largeur fixe est en conséquence la plus simple à utiliser et la plus couramment employée.

Les mises en page à largeur fixe ont cependant leurs inconvénients. Comme elles sont fixes, elles conservent la même taille, quelle que soit la taille de la fenêtre du navigateur. Elles utilisent donc mal l'espace disponible. Sur les écrans à haute résolution, les mises en page créées pour les écrans à $1\ 024 \times 760$ pixels peuvent sembler menues, comme perdues au milieu de l'écran. À l'inverse, une mise en page créée pour une résolution de $1\ 024 \times 760$ pixels contraindra l'utilisateur à faire défiler l'écran horizontalement avec les résolutions d'écran plus petites. Alors que la variété des tailles d'écran ne cesse d'augmenter, ces types de mises en page ont le défaut de mal s'adapter à la nature flexible du Web. Ils constituent à cet égard un compromis quelque peu maladroit.

Leur autre problème concerne les longueurs des lignes et la lisibilité du texte. Si elles fonctionnent bien avec la taille de texte par défaut des navigateurs, il suffit en revanche d'augmenter cette taille de quelques points pour que les colonnes latérales se mettent à manquer d'espace et que les longueurs de ligne soient insuffisantes pour une lecture confortable.

Pour contourner ces problèmes, vous pouvez choisir une mise en page liquide ou élastique plutôt qu'une structure à largeur fixe.

Mises en page liquides

Dans les mises en page liquides, les dimensions sont définies à l'aide de pourcentages et non de pixels. Ces mises en page peuvent ainsi se redimensionner en fonction de la fenêtre du navigateur. Plus cette dernière s'agrandit, plus les colonnes s'élargissent. Inversement, plus la fenêtre se rétrécit, plus les colonnes se réduisent en largeur. Les mises en page liquides exploitent efficacement l'espace et, si le travail est bien adapté, les mieux conçues ne se remarquent même pas.

Cependant, elles ont aussi leurs problèmes. Lorsque la largeur de la fenêtre du navigateur est étroite, les longueurs de lignes peuvent être très faibles et le texte difficile à lire, notamment avec les mises en page multicolonnes. Il peut alors être intéressant de spécifier une largeur minimale (`min-width`) en pixels ou en cadratins pour empêcher la mise en page de devenir trop étroite. Mais si la valeur de `min-width` est trop grande, la mise en page hérite alors des mêmes défauts que celles à largeur fixe.

À l'inverse, si la mise en page s'étend sur la largeur entière de la fenêtre du navigateur, les lignes peuvent devenir trop longues et difficiles à lire. Il existe plusieurs solutions pour éviter ce problème. Tout d'abord, au lieu de couvrir la totalité de la largeur, vous pouvez amener le conteneur à n'en couvrir qu'un certain pourcentage (par exemple, 85 %). Vous pouvez envisager de définir le remplissage et les marges internes sous forme de pourcentage également. Le remplissage et les marges augmenteront ainsi proportionnellement à la taille de la fenêtre, en empêchant les colonnes de s'élargir trop rapidement. Enfin, vous pouvez définir une largeur maximale pour le conteneur afin d'éviter que le contenu ne devienne excessivement large sur les moniteurs de grande taille.

Ces techniques peuvent être utilisées pour transformer la précédente mise en page à largeur fixe sur trois colonnes en une mise en page fluide à trois colonnes. Commençons par définir la largeur du conteneur sous la forme d'un pourcentage de la largeur totale de la fenêtre. La plupart des utilisateurs choisissent une taille arbitraire calculée en fonction de l'apparence qui leur convient à l'écran, et c'est bien ainsi. Mais si vous souhaitez être plus précis, examinez vos statistiques de navigateur pour calculer la taille de fenêtre la plus courante et choisissez un pourcentage de conteneur qui correspond à ce que la largeur fixe donnerait à cette taille. Liquid Fold (<http://liquidfold.net/>) est un excellent outil pour cela. Par exemple, si votre graphiste a utilisé une largeur de 960 pixels et que la fenêtre de navigateur de la majorité de vos utilisateurs est définie à 1 250 pixels, le pourcentage à utiliser est $(960/1\ 250) \times 100 = 76,8\%$.

Ensuite, fixez la largeur des zones de contenu principale et secondaire sous la forme d'un pourcentage de la largeur du conteneur. Dans l'exemple précédent, la largeur de la div de contenu principal était de 670 pixels. Comme la largeur totale était de 920 pixels, cela correspond à 72,82 %. De la même manière, la largeur de la div de contenu secondaire correspond à 25 % précisément. Ce qui laisse 2,18 % de gouttière virtuelle entre la navigation et le conteneur pour intégrer les erreurs d'arrondi et les irrégularités susceptibles de se produire :

```
.wrapper {  
    width: 76.8%;  
    margin: 0 auto;  
    text-align: left;  
}  
  
.content .primary {  
    width: 72.82%;  
    float: right;  
    display: inline;  
}  
.content .secondary {  
    width: 25%;  
    float: left;  
    display: inline;  
}
```

Vous devez ensuite définir les largeurs des colonnes à l'intérieur de la zone de contenu principale. Là, les choses se compliquent un peu, car les largeurs des div de contenu dépendent de la largeur de l'élément de contenu principal et non du conteneur global. Cette fois, la largeur de la div principale est de 400 pixels, ce qui correspond à 59,7 % de l'élément parent. De la même manière, la largeur de la div secondaire correspond à 34,33 % de l'élément parent. Pour finir, il faut encore une gouttière de 20 pixels, ce qui correspond à 2,63 % de l'élément parent.

```
.content .primary .primary {  
    width: 59.7%;  
    float: left;  
    display: inline;  
}  
  
.content .primary .secondary {
```

```

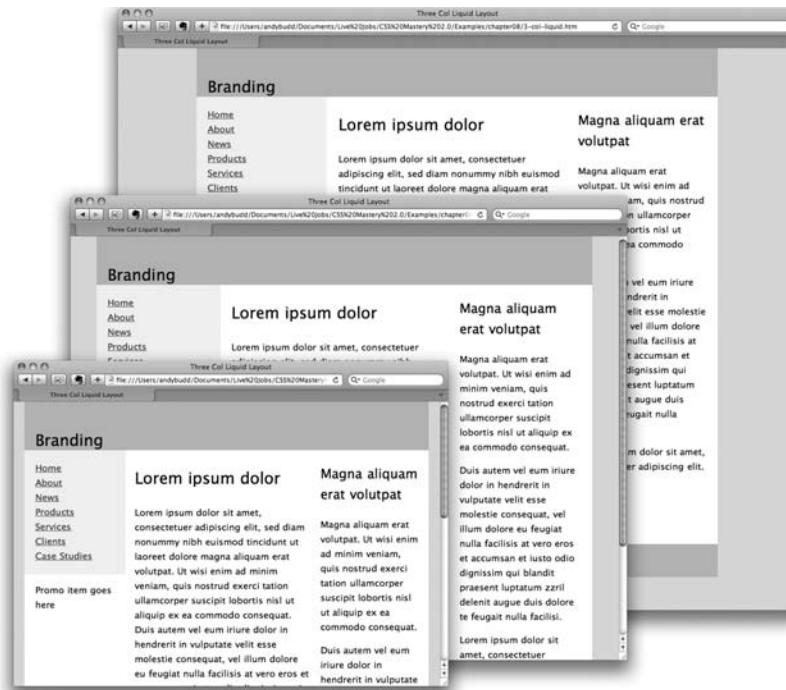
width: 34.33%;
padding-right: 2.63%;
float: right;
display: inline;
}

```

Ces paramètres produisent une mise en page liquide optimale pour une fenêtre de 1 250 pixels, mais aussi confortable pour la lecture avec des résolutions d'écran plus hautes ou plus restreintes (voir Figure 8.10).

Figure 8.10

Mise en page liquide à trois colonnes à 800 × 600, 1 024 × 768 et 1 250 × 900 pixels.



Comme cette mise en page s'adapte très agréablement, il n'est pas nécessaire d'ajouter une propriété `max-width`. Pour s'assurer que les lignes de texte conservent une longueur qui les rende lisibles, il est cependant toujours judicieux d'ajouter un réglage `max-width` défini en cadratins (em). La mise en page est un peu étriquée avec les fenêtres de petite taille : j'ajoute donc aussi un réglage `min-width` en cadratins.

```

.wrapper {
  width: 76.8%;
  margin: 0 auto;
  text-align: left;
  max-width: 125em;
  min-width: 62em;
}

```

Et voilà le travail : une mise en page agréable, flexible et liquide.

Mises en page élastiques

Si les mises en page liquides sont utiles pour tirer le meilleur parti de l'espace disponible dans la page, il peut néanmoins arriver que les longueurs de lignes soient excessives sur les moniteurs à haute résolution. À l'inverse, les lignes peuvent devenir trop courtes et fragmentées dans les fenêtres étroites ou lorsque la taille du texte est augmentée de quelques points. Si cela pose un problème, la solution peut se trouver dans une mise en page élastique.

Les mises en page élastiques définissent la largeur des éléments par rapport à la taille de la police au lieu de la largeur du navigateur. Avec des largeurs en cadratins, vous avez la garantie que, lorsque la taille de police augmente, toute la mise en page s'ajuste en fonction. Les longueurs de lignes conservent ainsi une taille lisible, un ajustement particulièrement utile pour les personnes à vision réduite.

Comme toutes les techniques de mise en page, celles-ci ont leurs inconvénients. Elles ont quelques défauts en commun avec les mises en page à largeur fixe, dont celui de ne pas exploiter pleinement l'espace disponible. En outre, elles peuvent devenir beaucoup plus larges que la fenêtre du navigateur – comme l'ensemble de la mise en page s'agrandit quand la taille du texte augmente – ce qui contraint à faire apparaître les barres de défilement horizontales. Pour éviter cela, on peut ajouter une propriété `max-width` fixée à 100 % de la div conteneur. `max-width` n'est pas pris en charge par Internet Explorer 6 et ses versions antérieures, mais il l'est par les nouvelles versions. Si vous souhaitez rendre `max-width` opérant dans Internet Explorer 6, vous pouvez utiliser du JavaScript également.

Les mises en page élastiques sont plus simples à créer que les mises en page liquides car tous les éléments HTML restent au même endroit les uns par rapport aux autres, se contentant de changer de taille. Transformer une mise en page à largeur fixe en mise en page élastique est assez simple. L'astuce consiste à définir la taille de police de base de manière qu'un cadratin (1 em) corresponde approximativement à 10 pixels.

Dans la plupart des navigateurs, la taille de police par défaut est de 16 pixels. 10 pixels correspondent à 62,5 % de 16 pixels. Il suffit alors de définir une taille de police pour l'élément `body` fixée à 62,5 % :

```
body {  
    font-size: 62.5%;  
    text-align: center;  
}
```

Comme 1 cadratin vaut maintenant 10 pixels à la taille de police par défaut, la mise en page à largeur fixe peut assez facilement être convertie en mise en page élastique. Dans les précédentes éditions de ce livre, j'ai recommandé de définir toutes les largeurs en cadratins. Ma collègue et relectrice Natalie Downe a cependant suggéré de conserver les largeurs internes sous forme de pourcentages et de ne définir que la largeur du conteneur en cadratins. De cette manière, les largeurs internes se redimensionnent toujours en fonction de la taille de police. Il est ainsi possible de modifier la taille globale de la mise en page sans avoir à changer la largeur de chaque élément individuel. Ce système est donc plus flexible et plus simple à gérer.

```
.wrapper {  
    width: 92em;  
    max-width: 95%;
```

```

margin: 0 auto;
text-align: left;
}

.content .primary {
width: 72.82%;
float: right;
display: inline;
}

.content .secondary {
width: 25%;
float: left;
display: inline;
}

.content .primary .primary {
width: 59.7%;
float: left;
display: inline;
}

.content .primary .secondary {
width: 34.33%;
padding-right: 2em;
float: right;
display: inline;
}

```

Avec ce code, on obtient une mise en page identique à la mise en page à largeur fixe dans le cas d'une taille de police standard (voir Figure 8.11), mais qui se réajuste élégamment lorsque la taille du texte est augmentée (voir Figure 8.12).

Figure 8.11

La mise en page élastique à la taille de texte par défaut.

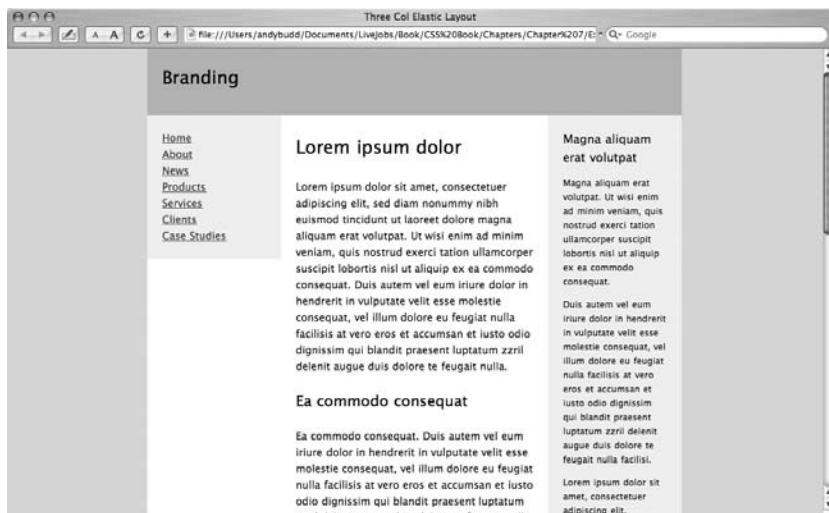




Figure 8.12

La mise en page élastique une fois la taille du texte augmentée de plusieurs crans.

Alors que presque tous les navigateurs modernes proposent des outils de zoom, on peut se demander si les mises en page élastiques avaient encore le moindre intérêt. Oui, tant que tous les navigateurs ne proposeront pas cette fonctionnalité, il sera intéressant de développer des mises en page élastiques pour les navigateurs plus anciens.

Images liquides et élastiques

Dans une mise en page liquide ou élastique, vous constaterez peut-être que les images à largeur fixe ont un effet dévastateur sur la structure des pages. Lorsque la largeur de la mise en page est réduite, les images se déplacent en fonction et peuvent interférer les unes avec les autres. Certaines créent des largeurs minimales naturelles, qui empêchent des éléments de diminuer de taille. D'autres sortent de leurs éléments conteneurs et chamboulent des mises en page pourtant ajustées au millimètre.

L'augmentation de la largeur de la mise en page peut avoir des conséquences dramatiques en créant des interstices indésirables et en déséquilibrant les proportions. Pas de panique : il existe quelques moyens d'éviter ces problèmes.

Pour les images qui doivent couvrir une grande zone, comme celles que l'on trouve dans les en-têtes des sites ou les logos, envisagez d'utiliser une image d'arrière-plan plutôt qu'un élément `img`. Ainsi, lorsque l'élément est mis à l'échelle, c'est une partie plus ou moins importante de l'image d'arrière-plan qui se révèle :

```
#branding {
    height: 171px;
    background: url(/img/branding.png) no-repeat left top;
}



</div>


```

Si l'image doit se trouver dans la page sous forme d'élément `img`, essayez de fixer la largeur de l'élément conteneur à 100 % et attribuez la valeur `hidden` à la propriété `overflow`. L'image sera détournée du côté droit de manière à tenir à l'intérieur de l'élément `branding`, mais elle se mettra à l'échelle en même temps que la mise en page :

```
#branding {  
    width: 100%;  
    overflow: hidden;  
}  
  
<div id="branding">  
      
</div>
```

Pour les images normales de contenu, vous voudrez sans doute qu'elles se redimensionnent verticalement et horizontalement, afin d'éviter tout détourage. Vous pouvez le faire en ajoutant un élément `img` à la page sans en spécifier les dimensions. Ensuite, définissez la largeur en pourcentage de l'image et ajoutez un paramètre `max-width` de la même taille que l'image pour éviter la pixellisation.

Par exemple, supposons que vous souhaitiez créer un style d'articles d'actualités avec une colonne d'image étroite à gauche et une colonne de texte plus large à droite. L'image doit faire approximativement le quart de la largeur de la boîte conteneur, le texte prenant le reste de la place. Vous pouvez le faire en fixant la largeur de l'image à 25 % et en définissant une valeur `max-width` égale à la taille de l'image, soit ici 200 pixels de large :

```
.news img {  
    width: 25%;  
    max-width: 200px;  
    float: left;  
    display: inline;  
    padding: 2%;  
}  
  
.news p {  
    width: 68%;  
    float: right;  
    display: inline;  
    padding: 2% 2% 2% 0;  
}
```

Lorsque l'élément `news` s'étend ou se contracte, l'image et les paragraphes s'étendent ou se contractent également, en conservant leur équilibre visuel (voir Figure 8.13). Dans les navigateurs compatibles avec les standards, l'image ne dépasse cependant jamais sa taille actuelle.

Figure 8.13

En donnant aux images une largeur en pourcentage, vous leur offrez la possibilité de se redimensionner également en fonction du contexte.



Lore ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim.



Lorem ipsum dolor sit amet, consecetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut labore et dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim.



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim.

Fausses colonnes

Vous aurez peut-être remarqué que les zones de navigation et de contenu secondaire dans toutes ces mises en page présentent un arrière-plan gris clair. Idéalement, l'arrière-plan devrait s'étendre sur toute la hauteur de la mise en page, afin de créer un effet de colonne. Pourtant, comme les zones de navigation et de contenu secondaire ne courent pas sur toute la hauteur, leurs arrière-plans s'arrêtent avant d'atteindre le bas.

Pour créer l'effet de colonne, vous pouvez fabriquer de fausses colonnes en appliquant une image d'arrière-plan répétée à un élément qui s'étend sur toute la hauteur de la mise en page, comme la div conteneur. Dan Cederholm a forgé l'expression anglaise de "faux column" (colonne factice) pour décrire cette technique.

Avec la mise en page en deux colonnes à largeur fixe, appliquez tout simplement à l'élément conteneur une image d'arrière-plan répétée verticalement et de même largeur que la zone de navigation (voir Figure 8.14) :

```
#wrapper {  
  background: #fff url(/img/nav-bg-fixed.gif) repeat-y left top;  
}
```

Figure 8.14

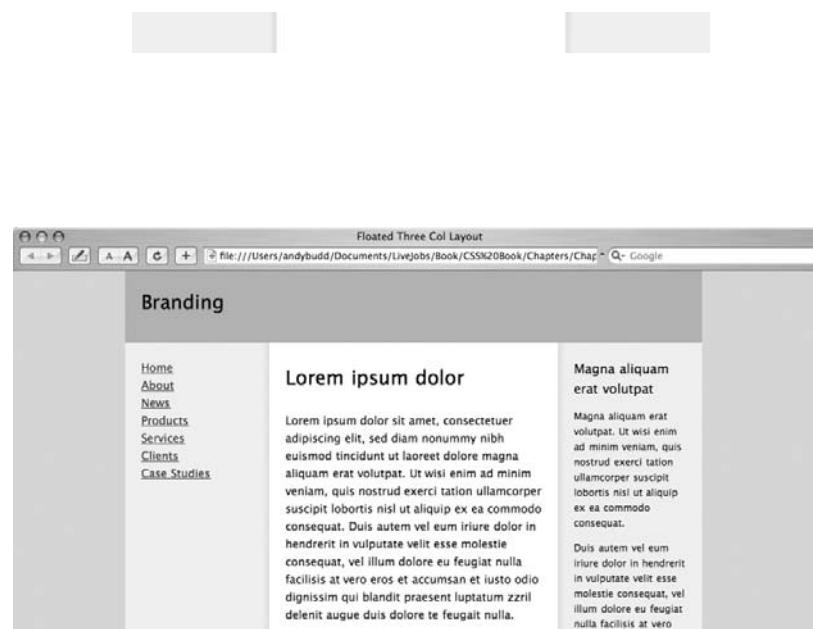
Colonne factice à largeur fixe.



Pour la mise en page en trois colonnes à largeur fixe, vous pouvez utiliser une méthode similaire. Cette fois, l'image d'arrière-plan répétée doit cependant couvrir toute la largeur du conteneur et inclure les deux colonnes (voir Figure 8.15). En appliquant cette image comme dans l'exemple précédent, vous créez un élégant effet de colonne factice (voir Figure 8.16).

Figure 8.15

Image d'arrière-plan utilisée pour créer l'effet des trois colonnes factices.

**Figure 8.16**

Effet des trois colonnes factices.

Il est assez facile de créer des colonnes factices pour les mises en page à largeur fixe, car vous connaissez nécessairement la taille des colonnes et leurs positions. Pour les mises en page fluides, l'affaire se complique quelque peu. Les colonnes changent de forme et de position

lorsque la fenêtre du navigateur est redimensionnée. L'astuce consiste alors à utiliser des pourcentages pour positionner l'image d'arrière-plan.

Si vous définissez une position d'arrière-plan à l'aide de pixels, le coin supérieur gauche de l'image est positionné par rapport au coin supérieur gauche de l'élément et décalé du nombre de pixels spécifié. Avec le positionnement en pourcentage, c'est le point correspondant dans l'image qui est positionné. Ainsi, si vous définissez une position verticale et horizontale de 20 %, vous positionnez en fait un point qui se trouve à 20 % du coin supérieur gauche de l'image et vous le calez à 20 % du coin supérieur gauche de l'élément parent (voir Figure 8.17).

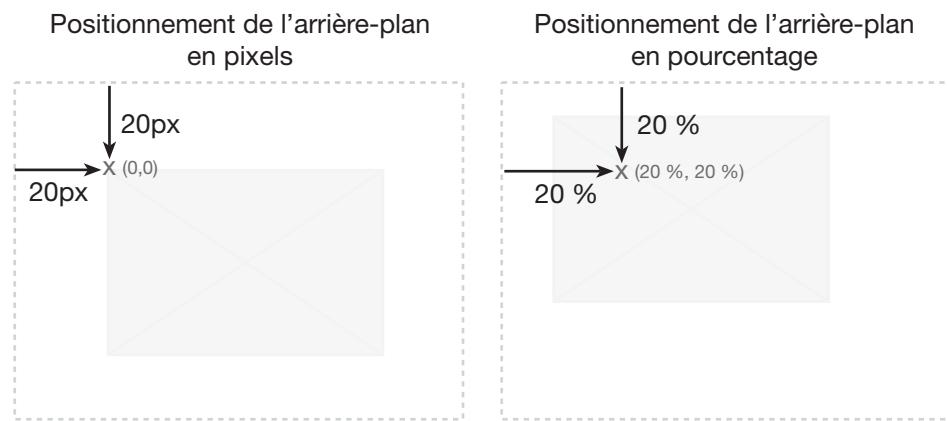


Figure 8.17

Lorsque le positionnement est spécifié en pourcentage, c'est la position correspondante de l'image qui est utilisée.

Le positionnement des images en pourcentage peut être très utile, car il permet de créer des images d'arrière-plan avec les mêmes proportions horizontales que la mise en page, puis de les positionner là où vous souhaitez que les colonnes apparaissent.

Pour créer une colonne factice pour la zone de contenu secondaire, commencez par créer une image d'arrière-plan très large. Dans cet exemple, j'ai créé une image de 4 000 pixels de large et 5 pixels de haut. Ensuite, vous devez créer une zone sur l'image d'arrière-plan qui agira comme une fausse colonne. La zone de contenu secondaire a été fixée à 25 % de la largeur du conteneur. Vous devez donc créer une zone correspondante sur l'image d'arrière-plan, qui atteint 25 % de sa largeur. Pour une image d'arrière-plan de 4 000 pixels de large, cela représente une colonne de 1 000 pixels de large. Exportez cette image au format GIF, en vous assurant que la zone non couverte par la colonne factice est transparente.

Le bord droit de la colonne factice se trouve maintenant à 25 % du côté gauche de l'image. Le bord droit de la zone de contenu secondaire est à 25 % du bord gauche de l'élément conteneur. Cela signifie que, si vous appliquez l'image comme arrière-plan à l'élément conteneur et fixez la position horizontale à 25 %, le bord droit de la colonne factice s'aligne parfaitement avec le bord droit de l'élément de navigation.

```
.wrapper {  
    background: #fff url(/img/secondary-faux-column.gif) repeat-y 25% 0;  
}
```

Vous pouvez créer l'arrière-plan pour la zone de contenu principale à l'aide d'une méthode similaire. Le bord gauche de cette colonne factice doit commencer à 72,82 % du bord gauche de l'image pour se caler sur la position de l'élément de contenu principal par rapport au conteneur. Comme l'élément conteneur possède déjà une image d'arrière-plan, vous devez ajouter un second élément conteneur à l'intérieur du premier. Vous pouvez ensuite appliquer la seconde image d'arrière-plan de colonne factice à ce nouvel élément conteneur.

```
.inner-wrapper {  
    background: url(/img/primary-faux-column.gif) repeat-y 72.82% 0;  
}
```

Si vous avez bien calculé les proportions, vous devriez ainsi obtenir une magnifique mise en page liquide à trois colonnes qui s'étire sur toute la hauteur du conteneur (voir Figure 8.18).



Figure 8.18

Mise en page à trois colonnes factices.

Colonnes de hauteurs égales

En plus des colonnes que vous créerez pour la mise en page principale, vous pouvez créer des colonnes de hauteurs égales à d'autres endroits (voir Figure 8.19). Si le résultat est facile à obtenir avec des tableaux, il est plus difficile à réaliser en CSS.

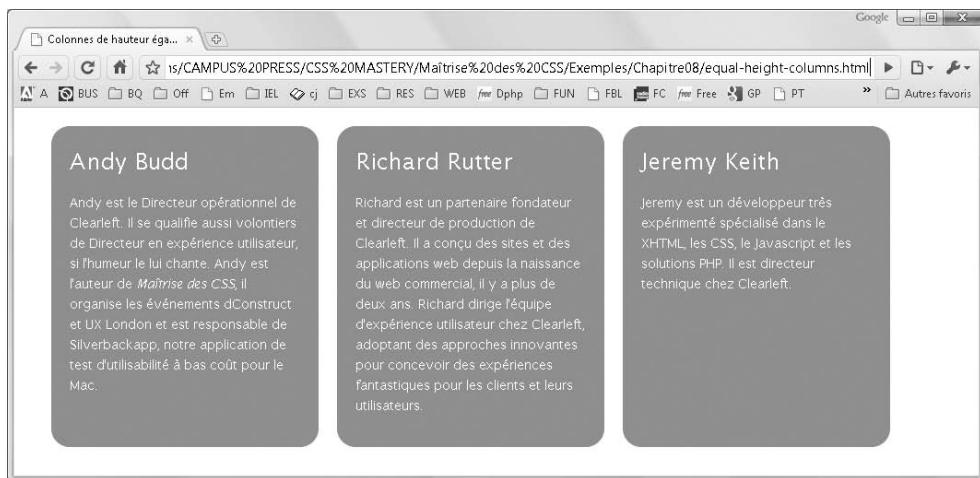


Figure 8.19

Trois colonnes de hauteurs égales.

Commençons par le code HTML.

```

<div class="wrapper">
  <div class="box">
    <h1>Andy Budd</h1>
    <p>...</p>
    <div class="bottom"></div>
  </div>

  <div class="box">
    <h1>Richard Rutter</h1>
    <p>...</p>
    <div class="bottom"></div>
  </div>

  <div class="box">
    <h1>Jeremy Keith</h1>
    <p>...</p>
    <div class="bottom"></div>
  </div>

```

Pour cet exemple, vous aurez besoin de trois div – une pour chacune des trois colonnes. À l'intérieur de chaque div, il faut un titre, du texte et une div vide à utiliser comme ancrage pour les bords arrondis du bas. Les trois div sont ensuite encloses dans une div conteneur (wrapper) que vous allez utiliser pour fixer la hauteur. Nous pouvons maintenant commencer à mettre en forme les boîtes.

```

.wrapper {
  width: 100%;
}

.box {

```

```
width: 250px;
margin-left: 20px;
float: left;
display: inline;
padding: 20px;
background: #89ac10 url(/img/top.gif) no-repeat left top;
}
```

Comme vous pouvez le voir à la Figure 8.20, on obtient ainsi trois colonnes de hauteurs inégales.

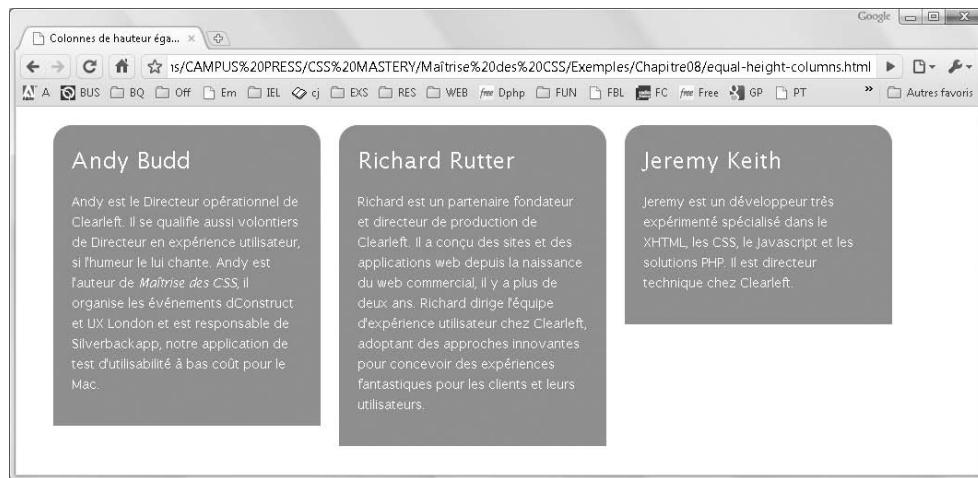


Figure 8.20

Les trois colonnes avant l'application de la technique principale.

L'astuce de cette technique est de donner à chaque boîte une grande quantité de remplissage inférieur puis à supprimer cette hauteur en excès à l'aide d'une valeur égale de marge négative. Chaque colonne se trouve ainsi forcée de déborder l'élément conteneur (voir Figure 8.21). Si vous attribuez ensuite la valeur `hidden` à la propriété `overflow` du conteneur, les colonnes sont tronquées au niveau de leur plus grande longueur. Dans cet exemple, je donne à chaque élément un remplissage inférieur de 520 pixels et une marge inférieure de 500 pixels. Les 20 pixels de différence forment le remplissage visible en bas de chaque boîte.

```
.wrapper {
  width: 100%;
  overflow: hidden;
}

.box {
  width: 250px;
  padding-left: 20px;
}
```

```

padding-right: 20px;
padding-top: 20px;
padding-bottom: 520px;
margin-bottom: 500px;
margin-left: 20px;
float: left;
display: inline;
background: url(/img/top.gif) #89ac10 top left no-repeat;
}

```

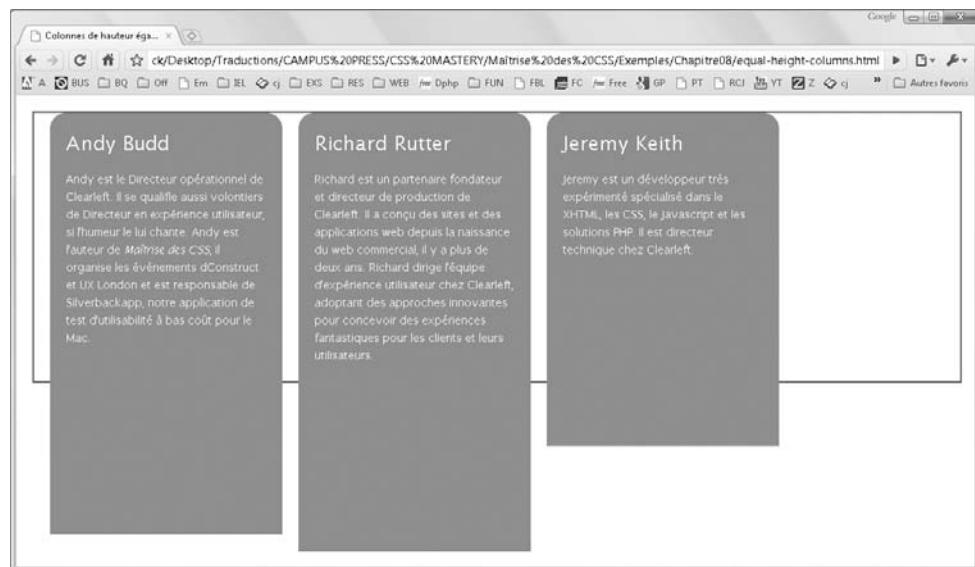


Figure 8.21

La bordure rouge délimite la div conteneur, afin que vous puissiez voir de quelle manière les trois colonnes débordent de cet élément.

Pour positionner le bas des colonnes au bon endroit, vous devez les aligner avec le bas de l'élément conteneur. Pour cela, vous devez d'abord définir le contexte de positionnement en attribuant au conteneur un positionnement relatif (`position: relative`). Vous pouvez ensuite attribuer un positionnement absolu aux div vides et la valeur `0` à leur propriété `bottom`. Il ne reste plus qu'à donner aux éléments la largeur et la hauteur adéquates puis à appliquer l'image du bas sous forme d'arrière-plan.

```

.wrapper {
  width: 100%;
  overflow: hidden;
  position: relative;
}

.box {
  width: 250px;
  padding-left: 20px;
  padding-right: 20px;
}

```

```

padding-top: 20px;
padding-bottom: 520px;
margin-bottom: 500px;
margin-left: 20px;
float: left;
display: inline;
padding: 20px;
background: url(/img/top.gif) #89ac10 top left no-repeat;
}

.bottom {
  position: absolute;
  bottom: 0;
  height: 20px;
  width: 290px;
  background: url(/img/bottom.gif) #89ac10 bottom left no-repeat;
  margin-left: -20px;
}

```

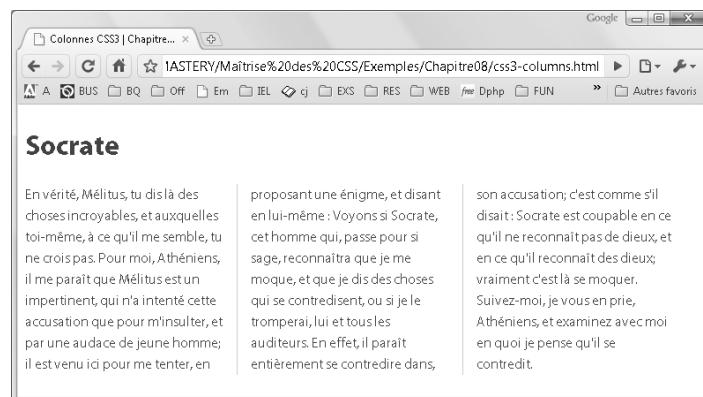
Résultat : une mise en page à trois colonnes qui conserve la hauteur de sa colonne la plus longue, comme le montre la Figure 8.19. Pas mal, non ?

Colonnes CSS 3

La spécification CSS 3 permet aussi de créer des colonnes de hauteurs égales (voir Figure 8.22), grâce aux propriétés `column-count`, `column-width` et `column-gap`.

Figure 8.22

Ces colonnes de texte ont été créées avec les propriétés de colonne de la spécification CSS 3.



Commencez par le balisage suivant :

```

<h1>Socrate</h1>
<div class="col">
  <p>Après avoir philosophé un moment...</p>
</div>

```

En appliquant ces règles, vous allez créer une mise en page à trois colonnes, chacune large de 14 cadratins et écartée des autres d'un intervalle de 2 cadratins. L'intérêt des colonnes CSS tient à leur gestion lorsque l'espace disponible est moindre que la largeur des colonnes définies. Au lieu que les colonnes soient chassées à la ligne comme cela se produit avec les éléments flottants, leur nombre se réduit. Si vous n'avez pas assez d'espace pour trois colonnes, vous revenez à deux colonnes.

```
.col {  
  -moz-column-count: 3;  
  -moz-column-width: 14em;  
  -moz-column-gap: 2em;  
  -moz-column-rule: 1px solid #ccc;  
  -webkit-column-count: 3;  
  -webkit-column-width: 14em;  
  -webkit-column-gap: 2em;  
  -webkit-column-rule: 1px solid #ccc;  
  column-count: 3;  
  column-width: 14em;  
  column-gap: 2em;  
  column-rule: 1px solid #ccc;  
}
```

Comme vous pouvez probablement le voir dans ce code, les colonnes CSS ne sont pas encore largement prises en charge. Vous devez donc flanquer le code standard d'extensions spécifiques des navigateurs.

Frameworks CSS et systèmes CSS

Dans l'univers de la programmation, les frameworks, comme Rails ou Django, reprennent les patterns de programmation courants du développement web, comme l'ajout d'enregistrements à une base de données, et en proposent une abstraction par le biais d'un jeu simple de composants réutilisables. Cette abstraction permet aux développeurs de construire des applications relativement sophistiquées sans devoir recréer ces fonctions de toutes pièces. À la différence des bibliothèques de fonctions autonomes, les frameworks ont tendance à être extrêmement intégrés. À cet égard, ils sont si abstraits qu'il est possible, encore que ce ne soit pas souhaitable, de créer des applications entières sans même comprendre le langage parent.

Au cours des deux dernières années sont lentement apparus ce qu'on a appelé des frameworks CSS. Censés réduire la corvée de programmation des CSS et aider les utilisateurs à créer des mises en page courantes sans éditer le code sous-jacent, ils encouragent en fait les développeurs à utiliser un certain nombre de procédés de balisage et de conventions de noms en laissant la mise en page se gérer à l'arrière-plan. Les trois frameworks les plus connus sont YUI Grids, Blueprint et 960 (voir Figure 8.23), mais il en existe quelques autres.

Ces frameworks offrent plusieurs avantages en termes de productivité, dont les réinitialisations de styles globaux, la gestion typographique de niveau site et le traitement uniformisé des formulaires – autant d'aspects dont vous aurez besoin pour la majorité de vos projets.

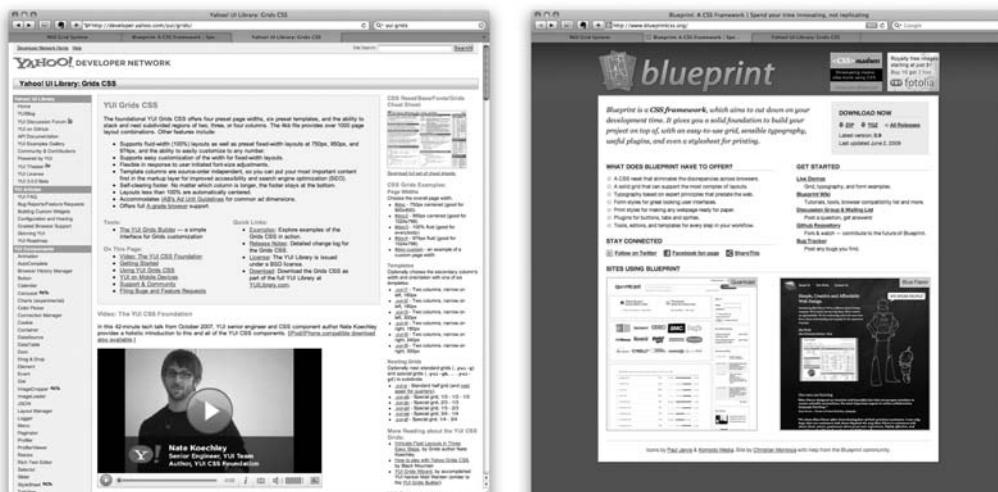


Figure 8.23

Les sites web YUI, Blueprint et 960.

Malheureusement, ils changent également votre manière de programmer le code HTML et brouillent la séparation entre présentation et signification. Par exemple, le balisage utilisé dans le framework Blueprint est par nature un balisage de présentation, qui s'exprime en termes de colonnes et d'étendue de colonnes :

```
<div class="column span-24">
  <!-- en-tête -->
</div>
<div class="column span-4">
  <!-- barre latérale gauche -->
</div>
<div class="column span-16">
  <!-- contenu principal -->
</div>
<div class="column span-4 last">
  <!-- barre latérale droite -->
</div>
```

Quand les développeurs emploient des frameworks pour contrôler la mise en page, ils se trouvent contraints d'utiliser un style de balisage axé sur la présentation qui ressemble aux anciennes maquettes à tableaux. On pourrait même avancer que les tableaux valent mieux que les frameworks CSS, car ils utilisent le même balisage axé sur la présentation mais sans les CSS en plus à charger. Les frameworks forcent en outre le développeur à connaître non seulement le langage sous-jacent mais aussi celui du framework. Le plus souvent, cela n'arrive même pas et le développeur se retrouve avec une connaissance superficielle des deux.

Les frameworks ont comme autre désavantage d'imposer une structure quadrillée particulière à vos mises en page. C'est parfait lorsque celles-ci concordent avec les largeurs et les marges définies par le framework. Mais tout comme il n'est pas acceptable que votre framework de programmation vienne imposer un fonctionnement type à votre site web, il est inacceptable que votre framework CSS en réglemente la présentation. Lorsque vous adoptez un framework spécifique, vous risquez de l'utiliser pour chaque projet et de vous retrouver dans une ornière. "Quand on n'a qu'un marteau, tout ressemble à un clou !"

Ces problèmes s'éclairent quand on comprend d'où viennent les frameworks. Au lieu d'être conçus de toutes pièces comme un système de mise en forme flexible pour toutes sortes de conceptions, ils ont été créés pour la plupart sur des sites comme Yahoo! ou le Laurence Kansas Journal. Ces sites possédaient déjà des structures quadrillées et des guides de styles bien définis. Les développeurs savaient donc que chaque nouvelle page obéirait aux mêmes principes. Au fil du temps, ils ont trouvé d'autres usages pour ces systèmes et en ont proposé des abstractions, puis les ont mis à disposition du public. Mais l'adaptation de ces frameworks à leurs sites d'origine reste évidente pour qui les utilise aujourd'hui.

Comment, dès lors, tirer parti des avantages en termes de productivité qu'offrent les frameworks CSS, sans pâtir de leurs inconvénients ? C'est tout l'intérêt des systèmes CSS. Un système CSS est une sorte de boîte à outils réutilisable de styles et de principes de balisage qui peut être utilisée pour développer des frameworks propres à tel ou tel site. Cette boîte à outils peut inclure vos redéfinitions globales, vos styles typographiques et vos traitements des formulaires, ainsi que des patterns de code HTML pour les composants HTML courants comme les formulaires d'inscription, les tableaux de calendrier et les listes de navigation. Vous pouvez ensuite utiliser les techniques apprises dans ce livre pour développer pour vos clients un système qui agit à la manière d'un framework personnalisé, avec toutes les options de mises en forme dont vous pourriez avoir besoin. Ce processus requiert au départ un petit effort supplémentaire de votre part, mais il offre tous les avantages d'un framework CSS sans ses inconvénients.

En résumé

Vous avez appris à créer des mises en page simples à deux et trois colonnes en largeur fixe à l'aide d'éléments flottants. Vous avez ensuite vu comment convertir ces mises en page en mises en page liquides et élastiques. Vous avez découvert certains des problèmes liés à ces types de structurations et vu quelles solutions étaient possibles en fixant des largeurs maximales en cadratins ou en pixels. Vous avez également vu comment créer des effets de colonne en pleine longueur avec les mises en page à largeur fixe ou flexible, en utilisant des images d'arrière-plan répétées verticalement. Ce chapitre a aussi fourni l'occasion d'aborder les techniques utilisées pour créer des mises en page CSS. Les techniques sont cependant innombrables et suffiraient à remplir un livre à elles seules. Enfin, vous avez découvert certains des dangers inhérents aux frameworks CSS et compris combien il était préférable de développer votre propre système CSS.

L'un des principaux écueils auxquels les développeurs se trouvent confrontés avec les mises en page CSS tient au manque d'uniformité dans le traitement des différents navigateurs. Pour résoudre les problèmes liés aux variations d'affichage des navigateurs, vous devez bien connaître les différents bogues et savoir les résoudre. Au chapitre suivant, vous allez découvrir certains des bogues les plus connus et vous familiariser avec les bases du débogage CSS.

9

Bogues et correctifs

Comparé à bien des langages de programmation, le langage des CSS est assez facile à apprendre. Sa syntaxe est simple et, parce qu'il est par nature axé sur la présentation, il n'exige pas de se familiariser avec une logique complexe. Tout se complique en fait lorsqu'il s'agit de tester le code dans différents navigateurs. Les bogues des navigateurs et leur manque d'uniformité dans l'affichage sont les principales embûches qui jalonnent le chemin des développeurs CSS. Telle mise en page peut avoir fière allure sur un navigateur et, sans raison apparente, se retrouver sens dessus dessous dans un autre.

La réputation qu'ont les CSS d'être difficiles ne provient pas du langage lui-même, mais des contorsions requises pour que les sites fonctionnent dans les anciens navigateurs. Il n'est pas facile de trouver des informations sur les bogues, mal documentés et parfois même mal compris. Les hacks sont souvent utilisés par les développeurs comme des formules magiques – incompréhensibles, mais qui miraculeusement, une fois appliquées au code, remettent tout en ordre. Ils font effectivement partie des outils dont il faut doter son arsenal, mais ils ne doivent être appliqués qu'avec parcimonie et généralement en dernier recours. Il est en réalité bien plus important de savoir pister, isoler et identifier les bogues. Ce n'est qu'une fois que l'on a précisément reconnu un bogue qu'il faut chercher des moyens de le résoudre.

Au cours de ce chapitre, vous en apprendrez plus sur :

- la manière de pister les bogues CSS ;
- la mystérieuse propriété `hasLayout` ;
- les hacks et les filtres ;
- les bogues de navigateur les plus courants et leurs correctifs ;
- la prise en charge échelonnée des navigateurs.

La chasse aux bogues

Tous les navigateurs sont bogués et certains plus que d'autres. Lorsqu'un développeur CSS rencontre un problème avec son code, il est même immédiatement tenté d'en conclure que le problème vient du navigateur et de chercher un hack ou une solution de rechange. Les bogues de navigateur ne sont pourtant pas si courants qu'on aime à le penser. Les problèmes CSS les plus courants proviennent non pas des bogues des navigateurs mais d'une compréhension incomplète de la spécification CSS. Pour éviter ces problèmes, il est préférable d'aborder les bogues CSS en présupposant d'abord qu'on a soi-même commis une erreur. Ce n'est qu'une fois que vous êtes sûr que vous n'avez pas fait d'erreur que vous devez considérer que le problème résulte d'un bogue de navigateur.

Problèmes CSS courants

Certains des problèmes CSS les plus simples sont causés par des erreurs typographiques et syntaxiques dans le code, par exemple si vous oubliez de terminer les déclarations par un point-virgule ou si vous tapez `font-face` au lieu de `font-family`. L'un des moyens simples d'éviter ces problèmes se trouve dans le choix d'un éditeur CSS comme SKEdit ou CSS Edit, qui inclut une mise en forme de la syntaxe et des fonctionnalités de saisie semi-automatique du code. Ces fonctionnalités évitent les erreurs de base mais ne remplacent pas une véritable validation. En analysant votre code avec un service comme le validateur CSS du W3C (<http://jigsaw.w3.org/css-validator/>), vous pourrez repérer toutes les erreurs grammaticales, connaître les lignes concernées par les erreurs et obtenir une brève description de chaque erreur (voir Figure 9.1).



Figure 9.1

Le site web Microsoft analysé par le validateur CSS du W3C.

L'extension Web Developer de Firefox (<https://addons.mozilla.org/en-US/firefox/addon/60>) inclut des raccourcis vers la version en ligne des validateurs HTML et CSS. Firefox possède lui-même un validateur HTML populaire (<http://users.skynet.be/mgueury/mozilla/>).

Lorsque vous validerez votre code HTML et CSS, vous tomberez peut-être sur une page pleine d'erreurs. Ne vous inquiétez pas. La plupart d'entre elles sont, en fait, le résultat d'une ou deux erreurs effectives. Si vous réparez la première erreur mentionnée et revalidez votre code, un grand nombre des erreurs suivantes disparaîtront. Procédez ainsi à quelques reprises et votre code devrait rapidement se retrouver sans erreurs.

Rappelez-vous que le validateur n'est qu'un outil automatisé et qu'il n'est pas infaillible. Un nombre sans cesse plus important de bogues ont été signalés dans le navigateur ; si vous pensez que votre code est correct alors que le validateur est d'un autre avis, vérifiez la dernière spécification CSS. Par exemple, à l'heure où ces lignes sont écrites, le validateur CSS affiche toujours des erreurs pour les extensions spécifiques des éditeurs comme `-moz-border-radius`, alors qu'elles sont effectivement autorisées dans la spécification CSS. En cas de doute, validez votre code en utilisant le profil CSS 3 et vérifiez la spécification.

Problèmes liés à la spécificité et à l'ordre de tri

En plus des erreurs syntaxiques, l'un des problèmes les plus courants concerne la spécificité et l'ordre de tri. Les problèmes de spécificité se manifestent généralement lorsque vous appliquez une règle à un élément et que vous constatez qu'elle reste sans effet. Vous appliquez d'autres règles, qui fonctionnent, mais certaines ne semblent pas fonctionner. Dans ce cas, le problème provient généralement du fait que vous avez déjà défini des règles pour cet élément autre part dans votre document, avec un sélecteur plus spécifique.

Dans l'exemple suivant, les développeurs ont passé en blanc la couleur d'arrière-plan de tous les paragraphes dans la zone de contenu (`content`). Ils souhaitent cependant que le paragraphe d'introduction soit orange et ont donc appliqué cette règle directement au paragraphe :

```
.content p {  
    background-color: white;  
}  
  
.intro {  
    background-color: orange;  
}
```

Si vous testez ce code dans un navigateur, vous verrez que le paragraphe d'introduction est toujours blanc. C'est que le sélecteur qui cible tous les paragraphes dans la zone de contenu est plus spécifique que celui qui cible le paragraphe d'introduction. Pour parvenir au résultat désiré, vous devez inverser ces propriétés. Ici, le meilleur moyen d'y parvenir est d'ajouter la classe de l'élément de contenu au début du sélecteur de paragraphe :

```
.content p {  
    background-color: white;  
}  
  
.content .intro {  
    background-color: orange;  
}
```

Évitez d'ajouter des sélecteurs spécifiques sans réfléchir, car vous risqueriez de provoquer des problèmes de spécificité dans d'autres parties du code. Au lieu de cela, il est souvent préférable de supprimer les sélecteurs superflus, de les rendre aussi génériques que possible et de n'ajouter de sélecteur spécifique que lorsque vous souhaitez un contrôle plus fin.

Comme indiqué au Chapitre 1, l'extension Firebug pour Firefox (<https://addons.mozilla.org/en-US/firefox/addon/1843>) est un outil très précieux pour le débogage des CSS.

L'une de ses fonctionnalités permet d'inspecter un élément afin de voir quels styles CSS sont redéfinis. Elle procède en barrant tous les styles redéfinis à d'autres endroits dans la feuille de styles (voir Figure 9.2).

Figure 9.2

Les styles sont barrés lorsqu'ils sont redéfinis dans d'autres parties de la feuille de styles.



Problèmes liés à l'effondrement des marges

L'effondrement des marges (voir le Chapitre 3) est une autre fonctionnalité CSS qui, lorsqu'elle est mal comprise, conduit les développeurs à s'arracher les cheveux. Considérez l'exemple simple d'un paragraphe imbriqué dans un élément div :

```
<div id="box">
  <p>Ce paragraphe possède une marge de 20px.</p>
</div>
```

La div box se voit attribuer une marge de 10 pixels et le paragraphe une marge de 20 pixels :

```
#box {
  margin: 10px;
  background-color:#d5d5d5;
}

p {
  margin: 20px;
  background-color:#6699FF;
```

On s'attendrait normalement à ce que le style résultant ressemble à la Figure 9.3, avec une marge de 20 pixels entre le paragraphe et la div et une marge de 10 pixels autour de la div.

Figure 9.3

Voici le résultat auquel on s'attendrait avec le style précédent.

Ce paragraphe a une marge de 20px.

Pourtant, le style donne le résultat présenté à la Figure 9.4.

Figure 9.4

Voici le résultat que l'on obtient.

Ce paragraphe a une marge de 20px.

Deux choses se passent ici. Tout d'abord, les marges de 20 pixels du haut et du bas du paragraphe s'effondrent avec la marge de 10 pixels de la div, ce qui forme une marge verticale de 20 pixels. Ensuite, au lieu d'être encloses par la div, les marges semblent dépasser du haut et du bas de la div. C'est l'effet du calcul des hauteurs réalisé pour les éléments possédant des enfants de niveau bloc.

Si un élément ne possède pas de bordure verticale ou de remplissage vertical, sa hauteur est calculée comme la distance entre les bords supérieur et inférieur de ses enfants contenus. Du coup, les marges supérieure et inférieure des enfants contenus semblent dépasser de l'élément conteneur. Il existe un correctif simple. En ajoutant une bordure verticale ou un remplissage vertical, les marges ne s'effondrent plus et la hauteur de l'élément est calculée comme la distance entre les bords des marges supérieure et inférieure de ses enfants contenus.

Pour que l'exemple précédent produise le résultat de la Figure 9.3, vous devez simplement ajouter un remplissage ou une bordure autour de la div :

```
#box {  
    margin: 10px;  
    padding: 1px;  
    background-color:#d5d5d5;  
}  
  
p {  
    margin: 20px;  
    background-color:#6699FF;  
}
```

La plupart des problèmes liés à l'effondrement des marges peuvent être résolus en ajoutant un léger remplissage ou une bordure fine de la même couleur que l'arrière-plan de l'élément concerné.

La vue topographique de la barre d'outils Web Developer est un excellent outil pour voir comment les éléments interagissent les uns avec les autres. Lorsque vous activez cette option, chaque élément reçoit une couleur d'arrière-plan qui dépend de sa position dans le document. Il est alors facile de voir comment les éléments sont positionnés les uns par rapport aux autres dans le document (voir Figure 9.5).

Figure 9.5

Vue topographique du site des extensions Firefox.



La vue Layout de Firebug (voir Figure 9.6) est un autre outil utile, qui indique les différentes dimensions de l'élément inspecté.

Figure 9.6

La vue Layout de l'en-tête du site des extensions Mozilla.



Les fondamentaux de la chasse aux bogues

La première étape pour pister un bogue consiste à valider les codes HTML et CSS et à vérifier la présence ou pas d'erreurs typographiques ou syntaxiques. Certaines erreurs d'affichage sont provoquées par des navigateurs qui affichent les pages en mode Quirks.

Il est donc judicieux de vérifier que vous utilisez le DOCTYPE adéquat pour votre langage de balisage afin que vos pages s'affichent en mode Standards (voir le Chapitre 1). La barre d'outils Web Developer indique le mode dans lequel la page s'affiche. Si c'est en mode Quirks, la coche en haut à droite de la barre d'outils est grise. En mode Standards, elle est verte. Cliquez dessus pour obtenir plus d'informations concernant la page, ainsi que pour définir explicitement le mode de rendu (voir Figure 9.7).

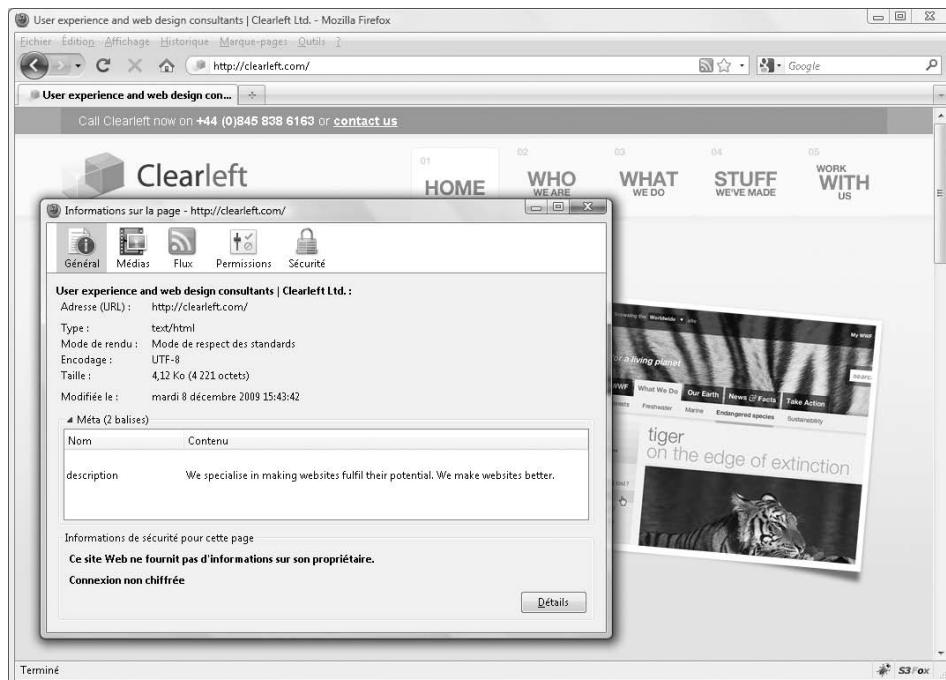


Figure 9.7

La barre d'outils Web Developer de Firefox indique si votre page s'affiche en mode Standards ou en mode Quirks.

De nombreux développeurs Windows avaient pour habitude de développer leurs pages en utilisant principalement Internet Explorer. Chaque fois qu'ils opéraient un changement, ils vérifiaient le résultat dans Internet Explorer pour voir si tout fonctionnait. Une fois que les pages étaient presque prêtes, ils testaient le résultat dans une série de navigateurs et tentaient de résoudre les incohérences. Cette procédure est malheureusement très dangereuse et peut poser de nombreux problèmes à long terme.

Internet Explorer 6 est un navigateur bogué, de notoriété publique, avec plusieurs défaillances CSS importantes. En l'utilisant comme navigateur de développement, certains développeurs ont par erreur interprété ce comportement comme s'il correspondait à la norme et se sont demandé pourquoi les navigateurs modernes dénaturaient leurs mises en page soigneusement

élaborées. En fait, les pages se trouvaient dénaturées dans Internet Explorer et s'affichaient conformément à la spécification dans les navigateurs plus compatibles avec les standards.

La méthode la plus sûre consiste à utiliser comme navigateur principal de développement un navigateur plus conforme aux standards, comme Firefox ou Safari. Si votre mise en page fonctionne alors, elle a toutes les chances de fonctionner dans l'ensemble des navigateurs conformes aux standards, ce qui garantit que vous procédez correctement. Vous pouvez ensuite tester vos pages dans les navigateurs moins compatibles et trouver des solutions pour tous les problèmes d'affichage rencontrés. Ne remettez pas le test des navigateurs à la dernière phase du projet. Au lieu de cela, adoptez une méthodologie de test continue, en vérifiant vos pages dans tous les principaux navigateurs à mesure que vous travaillez. Vous éviterez ainsi toute mauvaise surprise à la fin du projet, au moment où vous pensez en avoir fini.

Essayer d'éviter les bogues par avance !

Ce conseil peut sembler évident, mais l'un des meilleurs moyens de se débarrasser des bogues consiste à les éviter à l'avance ! De nombreux bogues d'affichage sont provoqués par du code HTML ou CSS inutilement compliqué. Il est donc judicieux d'utiliser le code le plus simple possible pour obtenir le résultat désiré. Évitez les techniques excessivement astucieuses en leur préférant des méthodes éprouvées et tâchez de réduire autant que possible le nombre des hacks utilisés.

Il existe une multitude de moyens pour réaliser les mêmes effets, aussi envisagez d'abord d'utiliser une autre méthode plutôt que de passer des heures à déboguer ou à créer des hacks. Ce n'est qu'une fois que vous avez la certitude qu'il n'y a pas moyen d'y échapper que vous devez prendre cette peine.

Isoler le problème

Une fois que vous êtes sûr que vous avez un bogue, vous devez isoler le problème. En procédant ainsi et en identifiant les symptômes, vous pourrez avec un peu de chance déterminer ce qui cause le problème et le résoudre. L'un des moyens de le faire consiste à appliquer des bordures ou des contours aux éléments concernés et à voir comment ils interagissent les uns avec les autres.

```
.promo1 {  
    float: left;  
    margin-left: 5px;  
    border: 1px solid red;  
}  
  
.promo2 {  
    float: left;  
    border: 1px solid green;  
}
```

J'ai personnellement tendance à ajouter des bordures directement au code, mais vous pouvez utiliser l'option de contour dans la barre d'outils Web Developer de Firefox ou l'un des

nombreux bookmarklets pour afficher le contour des différents éléments. Parfois, le simple fait d'ajouter des bordures résout le problème, notamment avec les problèmes d'effondrement des marges.

Essayez de changer quelques propriétés pour voir si elles affectent le bogue et, si c'est le cas, de quelle manière. Il peut être utile de tenter d'aggraver le bogue. Par exemple, si dans Internet Explorer l'interstice entre deux boîtes est plus grand que prévu, augmentez la marge pour voir ce qui se passe. Si l'espace entre les boîtes dans Internet Explorer a doublé, vous êtes probablement victime de son bogue de la double marge des éléments flottants.

```
.promo1 {  
    float: left;  
    margin-left: 40px;  
    border: 1px solid red;  
}  
  
.promo2 {  
    float: left;  
    border: 1px solid green;  
}
```

Essayez quelques correctifs courants. Par exemple, de nombreux bogues d'Internet Explorer se résolvent lorsqu'on attribue une valeur négative à la propriété `position`, ou la valeur `inline` à la propriété `display` (sur les éléments flottants) ou en définissant explicitement une dimension, comme la largeur. Vous en apprendrez plus sur ces correctifs courants et leur fonctionnement au chapitre suivant.

Un grand nombre des problèmes CSS peuvent être détectés et résolus rapidement, moyennant peu d'effort. Si le problème persiste, vous devez envisager de créer un cas de test minimal.

Créer des cas de test minimaux

Un cas de test minimal correspond simplement à la plus petite portion de code HTML et CSS requise pour répliquer le bogue. En créant un cas de test minimal, vous mettez à l'écart une partie des variables et simplifiez le problème autant que possible.

Commencez par dupliquer les fichiers présentant des problèmes. Supprimez le code HTML superflu jusqu'à ce qu'il ne reste plus que l'essentiel. Ensuite, commentez vos feuilles de styles afin de retrouver celle qui provoque le problème. Accédez à ces feuilles de styles et commencez à supprimer ou à commenter des blocs de code. Si le bogue s'arrête soudain, vous savez que la dernière portion de code commentée le provoque. Poursuivez ainsi jusqu'à ce qu'il ne reste que le code responsable des problèmes.

À partir de ce point, vous pouvez commencer à enquêter plus en détail sur le bogue. Supprimez ou commentez les déclarations et voyez ce qui se passe. Comment le bogue s'en trouve-t-il modifié ? Changez les valeurs de propriété et voyez si le problème disparaît. Ajoutez des correctifs courants pour voir s'ils ont un effet. Éditez le code HTML pour voir si cela produit

un effet. Utilisez différentes combinaisons d'éléments HTML. Certains navigateurs possèdent d'étranges bogues d'espaces blancs ; essayez donc de supprimer les espaces blancs de votre code HTML. La liste des zones d'exploration possibles est presque infinie.

Résoudre le problème, pas le symptôme

Une fois que vous avez repéré la racine du problème, vous êtes en bien meilleure position pour appliquer la bonne solution. Comme il existe une multitude de manières d'habiller un site CSS, la solution la plus simple est d'éviter complètement le problème. Si les marges sont à l'origine de votre problème, pensez à utiliser un remplissage à la place. Si c'est une combinaison d'éléments HTML, essayez de changer cette combinaison.

Beaucoup de bogues CSS possèdent des noms très descriptifs, ce qui facilite la recherche de solutions sur Internet. Si vous avez remarqué qu'Internet Explorer double les marges sur tous les éléments flottants, cherchez "Internet Explorer bogue marges doubles éléments flottants" et vous serez immanquablement conduit à une solution.

Si vous constatez qu'il n'est pas possible d'éviter le bogue, vous devrez peut-être traiter les symptômes. Cela nécessite généralement de filtrer la règle dans une feuille de styles séparée et d'appliquer un correctif pour le navigateur concerné.

Demander de l'aide

Si vous avez créé un cas de test minimal, testé les solutions courantes, recherché des correctifs possibles et que vous ne trouviez toujours pas de solution, demandez de l'aide. Il existe des communautés consacrées aux CSS, dont CSS-Discuss (www.css-discuss.org/), le Web Standards Group (<http://webstandardsgroup.org/>) et Stackoverflow (<http://stackoverflow.com>). Ces communautés regorgent de développeurs CSS aguerris. Il est donc fort probable qu'une personne ait déjà rencontré votre bogue et sache comment le résoudre. Si vous tombez sur un nouveau bogue ou un bogue particulièrement étonnant, il est possible que des utilisateurs soient intrigués et proposent des suggestions ou même vous aident directement à élaborer un correctif.

Ce qu'il faut retenir en demandant de l'aide, c'est que la plupart des développeurs web sont extrêmement occupés. Si vous n'avez pas validé votre code ou si vous avez simplement posté un lien vers votre site complet en comptant qu'ils parcourront des centaines de lignes de HTML/CSS, ne vous attendez pas à recevoir des tas de messages d'aide. Le meilleur moyen de demander de l'aide à une liste de diffusion ou à un forum consiste à utiliser un titre qui décrit précisément le problème, à écrire un résumé succinct du problème et à coller le cas de test minimal ou, s'il fait plus de quelques lignes de code, à proposer un lien vers le cas de test sur votre site. Les captures d'écran annotées sont aussi utiles, car il n'est pas toujours évident de comprendre quel est le problème à partir d'une description écrite, notamment si elle n'affecte que certaines versions de navigateur.

Avoir ou ne pas avoir un "layout"...

Tous les navigateurs ont des bogues et Internet Explorer 6 semble être le champion en la matière. L'une des raisons pour lesquelles il se comporte différemment tient à ce que son moteur de rendu s'appuie sur un concept interne appelé "layout". Comme le layout (qu'on pourrait à peu près traduire par "agencement") est un concept propre au fonctionnement interne du moteur de rendu, il n'est généralement pas nécessaire de s'en préoccuper. Pourtant, les problèmes de layout sont à la source de nombreux bogues d'affichage d'Internet Explorer. Par la force des choses, il est donc utile de comprendre ce concept et de voir comment il affecte les CSS.

Qu'est-ce que le "layout" ?

Internet Explorer sous Windows s'appuie sur le concept de layout pour contrôler la taille et le positionnement des éléments. Les éléments qui sont dits "avoir un layout" sont chargés de se redimensionner et de positionner eux-mêmes et leurs enfants. Si un élément "n'a pas de layout", sa taille et sa position sont contrôlées par son ancêtre le plus proche qui possède un layout.

Le concept de layout est un hack utilisé par le moteur de rendu d'Internet Explorer pour réduire sa charge de traitement. Idéalement, tous les éléments devraient contrôler leur taille et leur positionnement, mais cela pose de multiples et importants problèmes de performances. Du coup, l'équipe de développement du logiciel a décidé de n'appliquer la gestion de l'agencement des éléments qu'à ceux qui en avaient effectivement besoin, considérant qu'il serait ainsi possible d'améliorer considérablement les performances.

Les éléments auxquels un layout est associé par défaut sont les suivants :

- `body`
- `html` (en mode Standards)
- `table`
- `tr` et `td`
- `img`
- `hr`
- `input`, `select`, `textarea` et `button`
- `iframe`, `embed`, `object` et `applet`
- `marquee`

Le concept de layout est spécifique d'Internet Explorer sous Windows. Il ne s'agit pas d'une propriété CSS. Il ne peut pas être explicitement défini en CSS, même si certaines propriétés CSS attribuent automatiquement un layout aux éléments. Il est possible de voir si un élément possède un layout en utilisant la fonction JavaScript `hasLayout`. Elle renvoie `true`

si l'élément en possède un et `false` sinon. `hasLayout` est une propriété en lecture seule. Sa valeur ne peut être modifiée avec du code JavaScript.

Le fait de définir les propriétés CSS suivantes attribue automatiquement un layout à l'élément concerné :

- `float: left` ou `right` ;
- `display: inline-block` ;
- `width: n'importe quelle valeur` ;
- `height: n'importe quelle valeur` ;
- `zoom: n'importe quelle valeur` (propriété Microsoft – considérée comme invalide) ;
- `writing-mode: tb-rl` (propriété Microsoft – considérée comme invalide).

Depuis Internet Explorer 7, les propriétés suivantes attribuent aussi un layout :

- `overflow: hidden, scroll` ou `auto` ;
- `min-width: n'importe quelle valeur` ;
- `max-width: n'importe quelle valeur sauf none`.

Quel effet résulte du layout ?

Le layout est la cause de nombreux bogues d'affichage d'Internet Explorer. Par exemple, si un paragraphe de texte se trouve à côté d'un élément flottant, le texte est supposé courir le long de l'élément et l'habiller. Toutefois, dans Internet Explorer 6 et ses versions antérieures, si le paragraphe possède un layout (par exemple, parce que vous avez défini sa hauteur), il est contraint à une forme rectangulaire, ce qui empêche le texte de l'habiller en l'enveloppant (voir Figure 9.8).

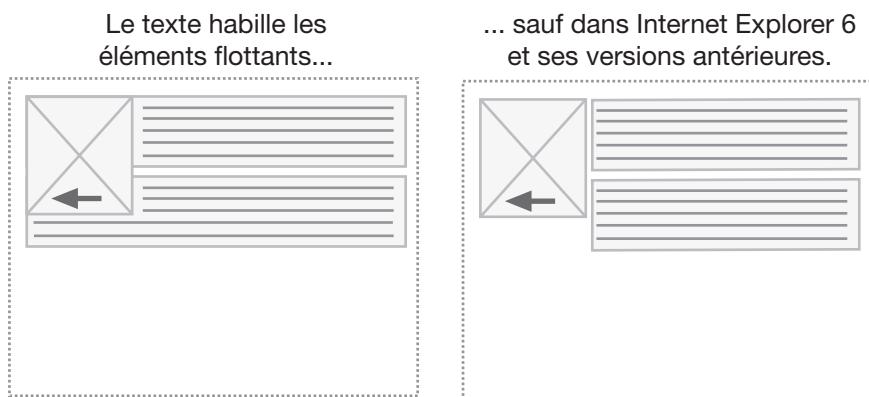


Figure 9.8

Le texte est censé courir le long des éléments flottants adjacents. Pourtant, dans Internet Explorer sous Windows, si l'élément texte possède un layout, cela ne se produit pas.

Les différences de rendu entre navigateurs peuvent poser toutes sortes de problèmes avec les mises en page flottantes.pire, bien des utilisateurs qui utilisent Internet Explorer comme navigateur principal presupposent par erreur qu'il s'agit du comportement normal et ne comprennent pas pourquoi les autres navigateurs traitent les éléments flottants différemment. En outre, le fait de donner à un élément un layout semble dégager tous les éléments flottants adjacents, comme si l'on choisissait `overflow:hidden`.

Il existe aussi un problème avec la manière dont les éléments avec layout se redimensionnent eux-mêmes. Selon la spécification CSS, le contenu d'un élément qui devient plus grand que cet élément est censé en déborder. Pourtant, dans Internet Explorer 6 et ses versions antérieures, les éléments avec layout s'agrandissent pour englober leur contenu (voir Figure 9.9).

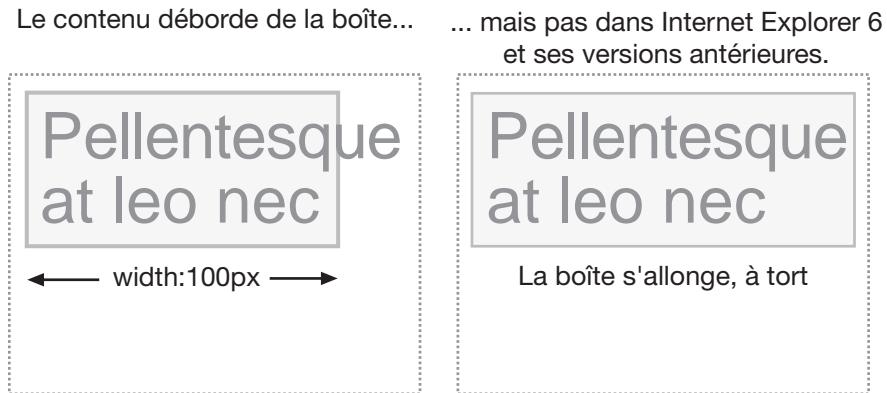


Figure 9.9

Les éléments dotés d'un layout s'agrandissent à tort pour inclure leur contenu.

Cette erreur de rendu signifie que la propriété `width` avec Internet Explorer sous Windows agit en réalité plutôt à la manière de `min-width`. Ce comportement est aussi à l'origine de nombreuses défaillances des mises en page flottantes. Lorsque le contenu d'une boîte flottante force à tort la largeur de la boîte à s'étendre, la boîte devient trop grande pour l'espace disponible et disparaît en dessous des autres éléments flottants.

Parmi les autres problèmes liés au layout, on peut citer :

- les éléments avec layout qui ne se réduisent pas en taille pour s'adapter à leur contenu ;
- les éléments flottants qui sont automatiquement dégagés par les éléments avec layout ;
- les éléments à positionnement relatif qui n'obtiennent pas de layout ;
- les marges qui ne s'effondrent pas entre les éléments avec layout ;
- la zone réactive des liens de niveau bloc sans layout qui ne couvre que le texte des liens ;
- les images d'arrière-plan sur les éléments de liste qui apparaissent et disparaissent par intermittence pendant le défilement.

Vous remarquerez que de nombreux correctifs pour Internet Explorer présentés plus loin imposent de définir des propriétés qui forcent l'élément à posséder un layout. En fait, si vous rencontrez un bogue avec Internet Explorer, l'une des premières choses à faire est de tenter d'appliquer les règles qui forcent l'élément à posséder un layout afin de voir si le problème peut ainsi se résoudre.

Si vous souhaitez en apprendre plus sur la propriété interne `hasLayout` d'Internet Explorer, je vous recommande de lire "On Having Layout" (en anglais) à l'adresse <http://www.satzansatz.de/cssd/onhavinglayout.html>.

Par chance, l'équipe Internet Explorer a résolu la plupart des problèmes liés au layout dans la version 7 du navigateur. Malheureusement, elle a procédé pour cela en repérant les bogues d'affichage courants et en créant des exceptions dans le code pour les gérer au lieu de résoudre les causes sous-jacentes. Il peut donc encore exister quelques bogues d'affichage obscurs qui n'ont pas été découverts. Internet Explorer 8, en revanche, utilise un tout nouveau moteur de rendu, volontairement débarrassé de la propriété `hasLayout` et donc exempt de ces problèmes.

Solutions de contournement

Dans le meilleur des mondes, les CSS correctement programmées devraient fonctionner dans tous les navigateurs qui les prennent en charge. Malheureusement, comme tous les éléments logiciels compliqués, les navigateurs ont leurs propres jeux de bogues et d'incohérences. Au départ, la prise en charge des CSS était assez peu répandue, si bien que les développeurs devaient s'efforcer d'être créatifs. En utilisant des bogues de traitement et des CSS non implémentées, ils ont pu trouver des solutions de contournement en appliquant sélectivement leurs règles aux différents navigateurs. Les hacks et les filtres sont ainsi devenus des armes incontournables dans leur arsenal.

Les navigateurs modernes proposent heureusement une meilleure prise en charge que leurs prédecesseurs, si bien qu'il n'est plus nécessaire de se soucier des hacks comme avant. Tant que les anciens navigateurs n'auront pas disparu pour de bon, il peut cependant encore être nécessaire de gérer du code à l'ancienne. Il est donc judicieux de se familiariser avec certains des hacks et des filtres les plus connus, ne serait-ce que pour les éviter dans le code. Avant cela, commençons cependant par une rapide présentation des commentaires conditionnels.

Les commentaires conditionnels d'Internet Explorer

Les commentaires conditionnels sont une extension propriétaire de Microsoft pour les commentaires HTML et ne sont donc pas standard eux-mêmes. Comme leur nom le suggère, ils permettent d'afficher des blocs de code en fonction d'une condition, comme une version de navigateur. S'ils ne sont pas standard, ils apparaissent dans tous les navigateurs anciens comme des commentaires standard et sont donc tout à fait inoffensifs. C'est la raison pour laquelle ils sont considérés comme le meilleur moyen de gérer les bogues propres à Internet Explorer. Ils sont apparus pour la première fois dans Internet Explorer 5 sous Windows et sont pris en charge par toutes les versions subséquentes du navigateur Windows.

Pour fournir une feuille de styles donnée à toutes les versions d'Internet Explorer 5 et au-delà, vous pouvez placer le code suivant dans l'en-tête de votre document HTML :

```
<!-- [if IE]
    <link rel="stylesheet" type="text/css" href="/css/ie.css" />
-->
```

Avec ce code, Internet Explorer 5 et ses versions ultérieures sous Windows reçoivent la feuille de styles ie.css, tandis que les autres navigateurs ne voient que du texte commenté. C'est intéressant, mais ce n'est pas particulièrement utile, car il est rare de trouver un bogue commun à toutes les versions d'Internet Explorer. Au lieu de cela, vous cherchez le plus souvent à cibler une version ou une gamme de versions spécifiques.

Avec les commentaires conditionnels, il est possible de cibler un navigateur particulier comme Internet Explorer 6, comme ceci :

```
<!-- [if IE 6]
    <link rel="stylesheet" type="text/css" href="/css/ie6.css" />
-->
```

Vous pouvez aussi cibler des séries de navigateurs comme IE 5 et IE 5.5 :

```
<!-- [if lt IE 6]
    <link rel="stylesheet" type="text/css" href="/css/ie5x.css" />
-->
```

Si les commentaires conditionnels permettent de proposer des feuilles de styles aux navigateurs Internet Explorer, ils peuvent aussi les leur masquer. La syntaxe suivante masque les styles plus avancés à toutes les versions d'Internet Explorer :

```
<!--[if !IE]-->
    <link rel="stylesheet" type="text/css" href="/css/advanced.css" />
<!--<![endif]-->
```

Le code suivant masque les styles à Internet Explorer 5.x :

```
<!--[if gte IE 6]><!-->
    <link rel="stylesheet" type="text/css" href="/css/modern.css" />
<!--<![endif]-->
```

Les commentaires conditionnels fonctionnent très bien et sont relativement simples à mémoriser. Leur principal désavantage tient à ce qu'ils doivent résider dans le code HTML et non dans les CSS. Si une nouvelle version d'Internet Explorer est lancée, vous pourrez être contraint de mettre à jour les commentaires conditionnels dans chaque page du site. Pour autant que vous n'oubliez pas de le faire, vous ne devriez cependant pas avoir de souci.

Avertissement concernant les hacks et les filtres

En tant que langage, les CSS ont été conçus pour rester compatibles avec l'évolution des versions. Si un navigateur ne comprend pas un sélecteur particulier, il ignore l'ensemble

de la règle. De la même manière, s'il ne comprend pas une propriété ou une valeur particulières, il ignore l'ensemble de la déclaration. C'est grâce à ce fonctionnement que l'ajout de nouveaux sélecteurs, de nouvelles propriétés et de nouvelles valeurs peut rester sans danger pour les navigateurs plus anciens.

Vous pouvez vous appuyer sur ce principe de fonctionnement pour fournir des règles et des déclarations aux navigateurs plus avancés tout en sachant que les anciens navigateurs se contenteront de les ignorer. Lorsqu'une nouvelle version d'un navigateur est lancée et prend en charge les règles CSS que vous utilisez comme filtre, tout doit fonctionner comme prévu. Si vous utilisez les CSS plus avancées pour résoudre un problème dans les navigateurs plus anciens, vous pouvez espérer que ce problème aura été réparé dans les versions plus récentes. Avec ce système, le recours aux CSS non prises en charge comme mécanisme de filtre se présente donc comme une option relativement sécurisée. On est forcé d'ajouter "relativement", car il existe toujours une possibilité que le navigateur prenne en charge le nouveau code CSS mais continue de produire le bogue qu'on croyait ainsi résolu.

Le recours aux filtres qui se fondent sur des bogues d'interprétation est un peu plus dangereux, parce que vous vous appuyez alors sur un bogue et non sur une fonctionnalité. Comme pour la précédente méthode, si le bogue d'interprétation est réparé mais que celui que vous essayez de corriger ne l'est pas, vous aurez un problème. Mais il y a pire, car certains bogues d'interprétation peuvent surgir dans des versions plus récentes de navigateurs. Ainsi, supposons qu'une nouvelle version de Firefox inclue un bogue d'interprétation particulier. Si celui-ci est utilisé comme filtre pour livrer à Internet Explorer d'autres valeurs de largeur à prendre en compte pour son modèle de boîte propriétaire, Firefox héritera soudain des mêmes largeurs et pourra commencer à mal afficher de nombreux sites. Il faut aussi se souvenir que ces types de hacks et de filtres invalident souvent le code. En règle générale, il est donc plus sûr d'utiliser des filtres qui s'appuient sur des propriétés CSS non prises en charge que sur celles qui utilisent des bogues de navigateurs. Mieux encore : tâchez de les éviter toutes.

Utiliser les hacks et les filtres avec raison

Malheureusement, les développeurs, notamment ceux qui découvraient les CSS, se sont excessivement appuyés sur les hacks et les filtres. Quand quelque chose ne fonctionnait pas dans un navigateur particulier, ils ont souvent eu le réflexe d'utiliser immédiatement un hack, comme un coup de baguette magique. Certains s'imaginent même pouvoir mesurer leur niveau d'expertise au nombre de hacks et de filtres obscurs qu'ils connaissent.

Si vous avez bien travaillé et que vous réalisiez que la seule option qui vous reste est d'employer un hack ou un filtre, vous devez le faire de manière raisonnable et contrôlée. Si vos fichiers CSS sont petits et simples et que vous deviez employer un ou deux hacks, il n'y a probablement pas de risque à les placer dans vos fichiers CSS principaux, en les accompagnant de commentaires appropriés. Cependant, les hacks sont assez souvent compliqués et peuvent rendre le code difficile à lire. Si vos fichiers CSS sont longs et compliqués ou si vous devez utiliser beaucoup de hacks, il peut être préférable de les séparer dans leurs propres feuilles de styles. Le fait de séparer les hacks non seulement rend le code plus facile à lire mais signifie également que si un hack commence à poser des problèmes dans un

prochain navigateur, vous saurez exactement où il se trouve. De la même manière, si vous décidez de ne plus vous préoccuper de la prise en charge d'un navigateur particulier, vous pourrez très facilement supprimer les hacks associés en supprimant le fichier correspondant.

Appliquer le filtre passe-bande pour Internet Explorer version Mac

Tantek Çelik a créé une série de filtres (<http://tantek.com/CSS/Examples/>) qui s'appuient sur des erreurs d'interprétation des navigateurs et permettent de fournir des feuilles de styles à des navigateurs sélectionnés en utilisant la règle `@import`. Les filtres ont d'abord été le moyen de choix de filtrer les différentes versions d'Internet Explorer, jusqu'à ce que les commentaires conditionnels se banalisent. Ils peuvent cependant encore être pratiques, par exemple si vous souhaitez explicitement cibler IE 5.2 sur Mac. Vous pouvez le faire en utilisant le filtre passe-bande IE 5 pour Mac de Tantek, qui exploite un bogue d'échappement dans les commentaires CSS :

```
/*\*\*\*/  
  @import "ie5mac.css";  
/**/
```

IE 5 pour Mac opère un échappement incorrect du second astérisque, ce qui le conduit à appliquer la règle `@import`. Il interprète ce code à peu près ainsi :

```
/* blah */  
  @import "ie5mac.css";  
/**/
```

Tous les autres navigateurs ignorent comme il se doit l'élément d'échappement, car il est entouré dans un commentaire et la règle `@import` est donc elle-même considérée comme un commentaire. Tous les autres navigateurs interprètent donc la règle de la manière suivante :

```
/* blah */*  
  blah  
*/
```

Comme avec les autres filtres passe-bande, il n'est pas nécessaire de comprendre le fonctionnement de ce filtre pour l'utiliser. L'intérêt de ces filtres est qu'ils ciblent spécifiquement des bogues dans les anciens navigateurs. Vous devriez donc pouvoir les utiliser en toute tranquillité, partant du principe qu'ils ne risquent pas de provoquer de problème dans les navigateurs plus récents.

Le hack de l'étoile HTML

Le filtre de "l'étoile HTML" est l'un des filtres CSS en ligne les plus connus et les plus utiles. Il est extrêmement aisé à mémoriser et cible le navigateur Internet Explorer 6 et ses versions antérieures. Comme vous le savez, l'élément HTML est supposé être le premier

élément (ou l'élément racine) des pages web. Pourtant, les anciennes versions d'Internet Explorer possèdent un élément racine anonyme qui enveloppe l'élément HTML lui-même. Avec le sélecteur universel, vous pouvez cibler un élément HTML enclos dans un autre élément. Comme cette cible n'existe que dans Internet Explorer 6 et ses versions antérieures, elle vous permet d'appliquer des règles propres à ces navigateurs :

```
* html {  
    width: 1px;  
}
```

Ce bogue a été corrigé dans Internet Explorer 7, si bien qu'il s'agit d'un moyen assez sécurisé de cibler les anciennes versions. Ce hack est utilisé dans le mécanisme du hack du modèle de boîte simplifié modifié (MSBMH ou *modified simplified box model hack*), un outil devenu populaire pour gérer le modèle de boîte propriétaire d'Internet Explorer dans les anciens navigateurs.

```
#content {  
    width: 80px;  
    padding: 10px;  
}  
  
* html #content {  
    width: 100px;  
    w\idth: 80px;  
}
```

Je ne conseille pas d'utiliser cette technique maintenant, mais il est bon de la connaître car vous risquez de la rencontrer si vous travaillez avec du code hérité.

Appliquer le hack du sélecteur d'enfants

Au lieu de cibler explicitement les anciennes versions d'Internet Explorer, supposons que vous souhaitez créer une règle que ces navigateurs ignoreront. Vous pouvez le faire en utilisant le hack du sélecteur d'enfants, encore que cette technique ne soit pas exactement un hack, puisqu'elle utilise simplement un sélecteur que les anciennes versions d'Internet Explorer ne comprennent pas alors que les navigateurs modernes la prennent en charge.

L'exemple suivant utilise le hack du sélecteur d'enfants pour masquer une image PNG d'arrière-plan transparente aux navigateurs Internet Explorer 5 et 6 sous Windows :

```
html>body {  
    background-image: url(bg.png);  
}
```

Cette règle ne sera pas aperçue par les versions anciennes d'Internet Explorer. En revanche, Internet Explorer 7 prend en charge le sélecteur d'enfants et la transparence PNG ; il pourra donc interpréter ce code correctement.

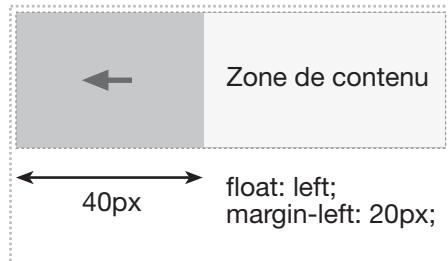
Bogues et correctifs courants

L'une des compétences les plus intéressantes dont puisse se doter un développeur CSS est la capacité à repérer les bogues courants des navigateurs. En connaissant les différents éléments qui concourent à provoquer ces bogues, vous pouvez les repérer et les corriger avant même qu'ils ne créent des problèmes.

Bogue de la double marge des éléments flottants

Le bogue de la double marge des éléments flottants d'Internet Explorer 6 et de ses versions antérieures est l'un des plus courants et des plus simples à repérer. Il amène Windows à doubler les marges de tous les éléments flottants (voir Figure 9.10).

Internet Explorer 6 et ses versions antérieures sous Windows doublent les marges des éléments flottants



Le bogue se corrige avec `display:inline`



Figure 9.10

Démonstration du bogue d'Internet Explorer doublant les marges des éléments flottants sous Windows. .

Ce bogue se corrige facilement quand on attribue la valeur `inline` à la propriété `display` de l'élément. Comme l'élément flotte, ce réglage n'affecte pas ses caractères d'affichage, mais il empêche miraculeusement Internet Explorer 6 et ses versions antérieures de doubler les marges. Ce bogue est si simple à repérer et à corriger que chaque fois que vous créez un élément flottant avec des marges horizontales, vous devriez attribuer la valeur `inline` à sa propriété `display`, par précaution.

Bogue du décalage de texte de 3 pixels

Le bogue du décalage de texte de 3 pixels d'Internet Explorer 5 et 6 sous Windows est aussi très courant. Il se manifeste lorsque du texte est adjacent à un élément flottant. Par exemple, supposons qu'un élément flotte à gauche et que vous ne souhaitez pas que le texte du paragraphe d'à côté l'enveloppe en le collant. Vous appliquez alors logiquement une marge gauche au paragraphe, de la même largeur que l'image :

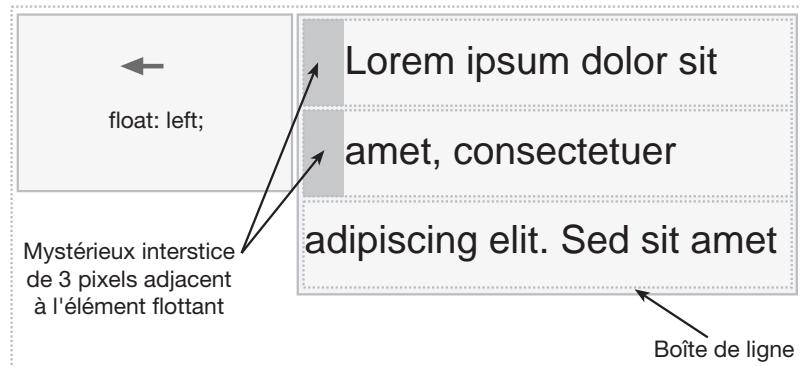
```
.myFloat {
  float: left;
  width: 200px;
}

p {
  margin-left: 200px;
}
```

Lorsque vous procédez ainsi, un mystérieux interstice de 3 pixels apparaît entre le texte et l'élément flottant. Dès le point où l'élément flottant s'arrête, l'interstice de 3 pixels disparaît (voir Figure 9.11).

Figure 9.11

Démonstration du bogue du décalage de texte de 3 pixels d'Internet Explorer 5 et 6.



Le correctif à ce bogue se déploie en deux phases. Tout d'abord, l'élément contenant le texte doit recevoir une hauteur arbitraire. Il est ainsi contraint de posséder un layout, ce qui supprime apparemment le décalage du texte. Comme Internet Explorer 6 et ses versions antérieures sous Windows traitent la hauteur comme une valeur `min-height`, le fait de définir une hauteur de petite taille n'a aucun effet sur les dimensions effectives de l'élément dans ce navigateur. En revanche, cela apparaît avec les autres navigateurs, si bien que vous devez masquer cette règle à tous les navigateurs sauf Internet Explorer 6 et ses versions antérieures sous Windows. Le meilleur moyen de le faire est de déplacer ces styles dans un fichier CSS séparé en utilisant des commentaires conditionnels.

```
p {
  height: 1%;
}
```

Malheureusement, cette technique pose un autre problème. Comme vous l'avez vu précédemment, les éléments avec un layout sont contraints de posséder une forme rectangulaire et apparaissent à côté des éléments flottants et non au-dessous. L'ajout de 200 pixels de remplissage crée en fait un interstice de 200 pixels entre l'élément flottant et le paragraphe dans Internet Explorer 5 et 6 sous Windows. Pour éviter cela, vous devez réinitialiser à zéro les marges :

```
p {
  height: 1%;
  margin-left: 0;
}
```

Le décalage de texte est réparé, mais un autre interstice de 3 pixels est maintenant apparu, cette fois sur l'image flottante. Pour le supprimer, vous devez définir une marge négative de 3 pixels sur l'élément flottant :

```
p {  
  height: 1%;  
  margin-left: 0;  
}  
  
.myFloat {  
  margin-right: -3px;  
}
```

Pour tous les éléments flottants qui ne sont pas des images, le problème est résolu à ce stade. Si l'élément est une image, il reste une dernière étape à franchir. Internet Explorer 5.x sous Windows ajoute un interstice de 3 pixels à gauche et à droite de l'image, alors qu'Internet Explorer 6 laisse les marges de l'image inchangées. Si vous devez assurer la prise en charge d'Internet Explorer 5.x, vous devez donc inclure une feuille de styles séparée pour ces versions de navigateurs :

```
p {  
  height: 1%;  
  margin-left: 0;  
}  
  
img.myFloat {  
  margin: 0 -3px;  
}  
and another for IE 6:  
p {  
  height: 1%;  
  margin-left: 0;  
}  
  
img.myFloat {  
  margin: 0;  
}
```

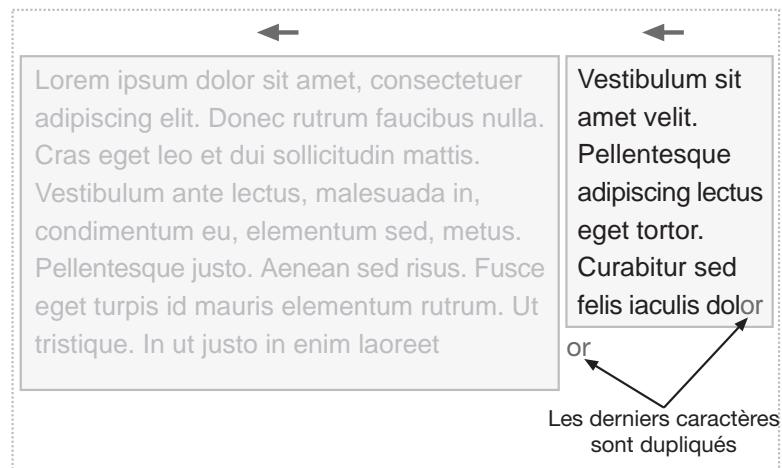
Bogue des caractères dupliqués d'Internet Explorer 6

Le bogue des caractères dupliqués d'Internet Explorer 6 est une autre curiosité à ajouter à notre musée. Dans certaines conditions, les quelques derniers caractères de la dernière série d'éléments flottants sont dupliqués sous l'élément flottant (voir Figure 9.12).

Ce bogue se manifeste lorsque plusieurs commentaires se trouvent entre la première et la dernière série d'éléments flottants. Les deux premiers commentaires n'ont pas d'effet, mais chaque commentaire subséquent provoque la duplication de deux caractères. Avec trois commentaires, on obtient donc deux caractères dupliqués. Avec quatre commentaires, on en obtient quatre et avec cinq, on en obtient six.

Figure 9.12

Le bogue des caractères dupliqués d'Internet Explorer 6.



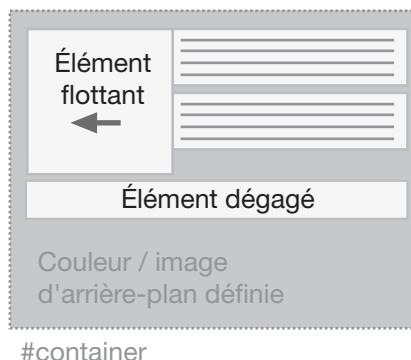
```
<div id="content">
  <!-- mainContent -->
  <div id="mainContent">
  ...
  </div><!-- end mainContent -->
  <!-- secondaryContent -->
  <div id="secondaryContent">
  ...
  </div>
```

Bizarrement, ce bogue semble lié au bogue de décalage de texte de 3 pixels que nous avons présenté précédemment. Pour le corriger, vous pouvez supprimer les 3 pixels du dernier élément flottant en définissant une marge négative ou agrandir le conteneur de 3 pixels en largeur. Ces deux méthodes risquent cependant de poser des problèmes dans Internet Explorer 7, qui ne provoque pas ce bogue. Voilà pourquoi le moyen le plus simple de l'éviter consiste à supprimer les commentaires de votre code HTML.

Bogue du coucou d'Internet Explorer 6

Dans le rayon des grandes bizarries, on trouve encore le bogue du coucou d'Internet Explorer 6, ainsi baptisé parce que, dans certains cas, le texte semble disparaître et ne réapparaît que lorsque la page est réactualisée. Il se produit lorsqu'un élément flottant est suivi par des éléments non flottants puis un élément qui spécifie un dégagement, tous contenus dans un élément parent possédant une couleur ou une image d'arrière-plan. Si l'élément qui spécifie un dégagement touche l'élément flottant, les éléments non flottants entre les deux disparaissent sous la couleur ou l'image d'arrière-plan de l'élément parent et ne réapparaissent qu'après actualisation de la page (voir Figure 9.13).

Le contenu suit un élément flottant et précède un élément dégagé.



Le contenu disparaît dans Internet Explorer 6 mais réapparaît une fois la page actualisée.

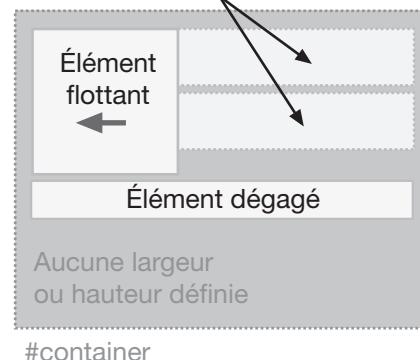


Figure 9.13

Démonstration du bogue d'Internet Explorer doublant les marges des éléments flottants sous Windows.

Il existe heureusement plusieurs moyens de combattre ce bogue. Le plus simple est probablement de supprimer la couleur ou l'image d'arrière-plan de l'élément parent. Mais ce n'est pas toujours très pratique. On peut aussi éviter à l'élément dégagé de toucher l'élément flottant. Le bogue ne semble pas se manifester si on applique des dimensions précises à l'élément conteneur. Il ne se manifeste plus non plus si le conteneur possède une hauteur de ligne définie. Enfin, le problème disparaît également quand la propriété `position` de l'élément flottant et du conteneur a la valeur `relative`.

Positionnement absolu dans un conteneur relatif

Le dernier bogue de navigateur important dont nous allons traiter concerne le positionnement absolu d'éléments dans un conteneur positionné de manière relative. Vous avez appris au cours des précédents chapitres combien il pouvait être utile d'imbriquer un élément positionné de manière absolue dans un conteneur relatif. Malheureusement, Internet Explorer 6 et ses versions antérieures produisent un certain nombre de bogues avec cette technique.

Ces bogues proviennent du fait qu'avec Internet Explorer pour Windows, les éléments positionnés de manière relative ne récupèrent pas la propriété interne `hasLayout`. Du coup, ils ne créent pas de nouveau contexte de positionnement et tous les éléments positionnés le sont relativement à la fenêtre du navigateur (voir Figure 9.14).

Pour corriger le comportement d'Internet Explorer 6 et de ses versions antérieures sous Windows, vous devez forcer l'attribution d'un layout au conteneur positionné de manière relative. L'un des moyens d'y parvenir est de lui attribuer une largeur et une hauteur explicites. Malheureusement, vous aurez souvent recours à cette technique précisément quand

vous ne connaissez pas la largeur et la hauteur du conteneur ou lorsque vous souhaitez que ces propriétés restent flexibles.

Au lieu de cela, vous pouvez utiliser des commentaires conditionnels pour filtrer Internet Explorer 5 et 6 et donner au conteneur un layout en appliquant une dimension arbitraire. Comme les éléments dans Internet Explorer 6 et ses versions antérieures s'étendent à tort pour englober leur contenu, la hauteur effective ne s'en trouvera pas affectée.

```
.container {
  height: 1%;
}
```

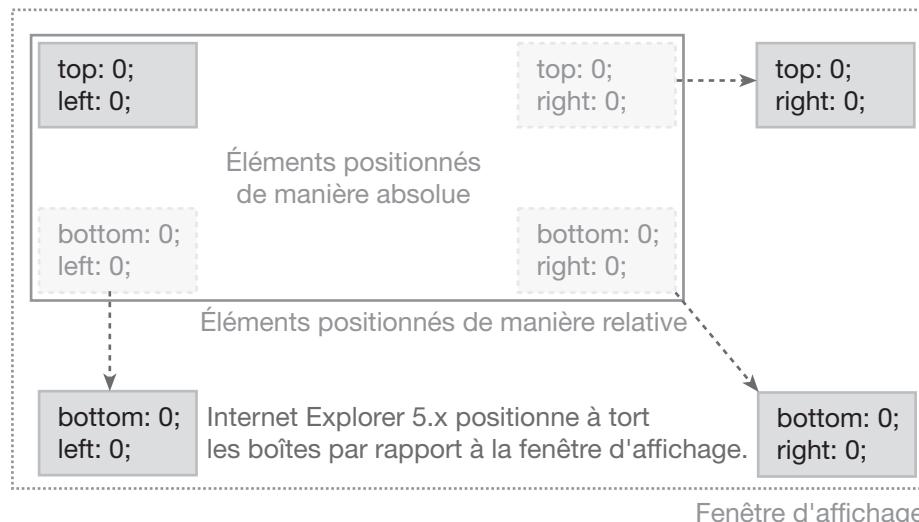


Figure 9.14

IE 5.x place mal les éléments positionnés de manière absolue lorsque leur conteneur est positionné de manière relative.

Cesser de se référer à Internet Explorer

Internet Explorer n'est pas le seul navigateur bogue. Vous vous demandez donc peut-être pourquoi j'ai concentré mon attention sur ses bogues. Ce n'est pas de l'allergie à Microsoft. Il s'agit d'un choix raisonné.

Tout d'abord, Internet Explorer conserve une part de marché importante, si bien que ses bogues sont assez rapidement repérés et documentés, mais son rythme de développement est cependant plus lent que celui des autres navigateurs. Alors que Firefox et Safari proposent de nouvelles versions tous les quelques mois, il faut parfois attendre plusieurs années pour voir apparaître une nouvelle version d'Internet Explorer. Ses bogues ont donc une durée de vie bien plus étendue.

La vitesse à laquelle les bogues sont repérés et réparés dans Firefox et Safari est assez impressionnante, mais il y a bien quelques problèmes. Au lieu d'avoir à gérer deux ou trois

versions d'un navigateur, vous pouvez avoir affaire à dix ou vingt modèles distincts. Vous ne saurez jamais si vos utilisateurs possèdent la dernière version, ce qui complique énormément les tests. À la différence, Internet Explorer n'a pas proposé de révision majeure pendant près de cinq ans. Les bogues ont donc eu bien plus de temps pour remonter à la surface et trouver un correctif.

Internet Explorer 8 est, par chance, un navigateur beaucoup plus conforme aux standards que les précédentes versions. Un grand nombre des bogues connus ont été corrigés et la prise en charge des CSS 2.1 avancées s'est accrue. Comme avec tous les navigateurs, on verra apparaître de nouveaux bogues et Internet Explorer 8 est loin d'être parfait, mais plus les utilisateurs pourront se convaincre qu'ils ont intérêt à passer aux navigateurs modernes comme Internet Explorer 8 et Firefox, plus on pourra rapidement ranger au placard les anciens modèles comme Internet Explorer 6.

Prise en charge graduelle des navigateurs

On ne peut clore un chapitre sur les bogues sans mentionner la question de la prise en charge des navigateurs. Chaque fois qu'une nouvelle version d'Internet Explorer est lancée, elle provoque de vifs débats concernant la date à laquelle on pourra sans risque cesser de prendre en charge les précédentes versions. Après tout, si Microsoft n'assure officiellement plus le support d'Internet Explorer 6, pourquoi s'en préoccuper ? Malheureusement, il n'existe pas de solution simple et évidente au problème de la prise en charge des navigateurs. Tout dépend des caractéristiques de votre site.

Si vous hébergez un site pour les développeurs web, vos utilisateurs surferont en majorité avec Firefox ou Safari sur Mac, auquel cas la part des utilisateurs d'Internet Explorer 6 peut être si réduite qu'il ne vaut pas la peine de s'en soucier. En revanche, même quelques pour cent sur un site qui accueille un million de visiteurs par mois, cela peut signifier plusieurs milliers d'internautes mécontents. Pour les sites commerciaux ou les sites d'abonnés, le nombre des utilisateurs d'Internet Explorer 6 peut être bien plus important. Dans certains cas, il peut même dépasser celui des utilisateurs de la version 7. Certains services informatiques dans les bureaux empêchent en effet les employés de changer de version de navigateur, tandis que les utilisateurs domestiques ne font parfois la mise à niveau que le jour où ils changent d'ordinateur. Au lieu d'envisager d'abandonner complètement la prise en charge d'un navigateur, il peut donc être préférable de définir des échelons et de décider ce que signifient les différentes prises en charge pour chaque site. C'est tout l'enjeu de la prise en charge graduelle des navigateurs.

Les grandes organisations comme Yahoo! et la BBC ont bien conscience que tous les navigateurs ne sont pas égaux et que s'assurer que leurs sites possèdent exactement la même apparence dans tous les navigateurs augmenterait radicalement les coûts de maintenance et entraînerait de nombreuses possibilités d'innovation. Pour éviter d'avoir à calibrer la conception des sites sur le plus petit dénominateur commun en termes de navigateur, ces organisations ont commencé à adopter des chartes de prise en charge graduelle (voir Figures 9.15 et 9.16).

	Win 2000	Win XP	Win Vista	Mac 10.4+	Mac 10.5+
Firefox 3.0+		Niveau A	Niveau A		Niveau A
Firefox 2.0+		Niveau A			Niveau A
IE 8.0		Niveau A	Niveau A		
IE 7.0		Niveau A	Niveau A		
IE 6.0	Niveau A	Niveau A	Niveau A		
Opera 9.6+		Niveau A			Niveau A
Safari 3.2+				Niveau A	Niveau A

Figure 9.15

La charte de prise en charge graduelle de Yahoo! pour les navigateurs de niveau A.

Navigateur	IE	Mozilla	Opera	Safari	Konqueror	IE	NS 4-
Plateforme	Windows	Tout	Tout	Mac	Linux	Mac	Tout
Niveau 1	6, 7	FF 1.5.x / 2.0.x	9	1.3+, 2.x, 3.x			
Niveau 2	5, 5.5	FF 1.0.x	8	1.0, 1.1, 1.2	3+		
Niveau 3	1, 2, 3, 4		7		2-	5-	1, 2, 3, 4
Test obligatoire	6, 7	FF 2.0.x	9	2.0			
Test souhaité	5.5	FF 1.5	8	1.3, 3.x			
Notes	n°1	n°1	n°1	n°1	n°2		
Moteur	Trident (4-7) Ohare (1-3)	Gecko	Presto	Webcore	KHTML	Tasman	Mariner

Figure 9.16

La table de prise en charge graduelle des navigateurs pour le site de la BBC.

Plutôt que de considérer la prise en charge des navigateurs comme une option binaire permettant entre deux états que sont la prise en charge et l'absence totale de prise en charge, ces chartes proposent une variété de niveaux de prise en charge, des caractéristiques d'affichage complètes pour les navigateurs modernes jusqu'au contenu brut pour les plus anciennes versions. Si chaque organisation aborde différemment ce problème, les étapes restent sensiblement les mêmes.

Pour commencer, vous devez identifier les navigateurs avec lesquels vous souhaitez obtenir un rendu uniforme dans l'ensemble de votre site et effectuer des tests sur tous ceux-là. Il s'agira généralement des plus populaires utilisés par vos visiteurs. S'y rangeront donc sans doute les dernières versions de Firefox, Safari et Opera, ainsi qu'Internet Explorer 7 et 8. Avec ces navigateurs, le but est que le site possède la même apparence, même si, pour des raisons pratiques, on peut accepter quelques innocents pixels de différence ici ou là.

Ensuite, vous devez identifier un ensemble de navigateurs vieillissants mais importants. Ce groupe peut inclure les versions plus anciennes de Firefox et de Safari ainsi qu'Internet Explorer 6. Vous effectuerez les tests sur un échantillon aléatoire de ces navigateurs et tentez de résoudre tous les problèmes que vous trouverez. Vous devrez cependant accepter que l'affichage ne soit pas parfait et qu'il puisse différer d'un navigateur à l'autre, pour autant que le contenu reste parfaitement accessible.

Pour finir, vous devez identifier un ensemble de navigateurs plus rares ou relativement périmés que vous ne souhaitez pas officiellement prendre en charge. Ce groupe peut inclure les navigateurs Internet Explorer 4, Netscape Navigator 4 ou Opera 7. Avec ces navigateurs, il faut toujours que le contenu et les fonctionnalités restent disponibles, mais inutile de se soucier de la présentation. Vous pouvez donc accepter des variations importantes du rendu. Il serait même préférable, pour ces navigateurs, de supprimer tous les styles.

La technique de la prise en charge graduelle est un moyen plus souple de gérer l'ensemble des navigateurs et des agents utilisateur. Les tableaux de la BBC sont un bon point de départ, mais puisque chaque site est unique, je recommande vivement de créer vos propres tableaux pour chacun de vos projets.

En résumé

Vous avez découvert quelques importantes techniques pour pister et éradiquer les bogues CSS. Vous avez pu vous familiariser avec la propriété interne `hasLayout` d'Internet Explorer sous Windows et découvrir sa part de responsabilité dans de nombreux bogues. Vous avez également découvert certains des bogues de navigateurs les plus courants et appris à les corriger. Pour finir, vous avez vu comment gérer une myriade de navigateurs différents à l'aide de chartes de prise en charge graduelle.

À présent, vous allez voir, dans deux études de cas créées par deux des meilleurs concepteurs et développeurs CSS de notre temps, comment toutes ces informations peuvent être combinées.

10

Étude de cas : Roma Italia

Annoté et corné, mon exemplaire de la première édition de ce livre trône toujours fièrement sur mon étagère. Je l'ai souvent consulté au cours des trois années qui ont suivi sa publication. Bien des choses ont changé dans ce domaine, depuis le temps, notamment avec la sortie d'Internet Explorer 7 (et plus tard, d'Internet Explorer 8), qui ouvre le champ à de nouvelles fonctionnalités CSS 2 et CSS 3 maintenant prises en charge par les navigateurs les plus importants. Nous aborderons quelques exemples de ces fonctionnalités dans la présente étude de cas.

Beaucoup de choses sont aussi restées inchangées. Le balisage est identique à lui-même. Les standards sont toujours des standards. Et il est plus que jamais important de disposer d'individus talentueux comme vous l'êtes pour produire des sites attrayants et passionnants en HTML, CSS et JavaScript.

Il est probable que vous, lecteur, soyez plus savant et expérimenté que vous ne l'étiez à la première publication de ce livre. J'espère donc que l'étude de cas que j'ai élaborée vous donnera plus de sensations encore que la précédente. Dans cet exemple, vous en apprendrez plus sur :

- la mise en page et la grille à 1 080 pixels ;
- les fonctionnalités CSS 2 et CSS 3 avancées ;
- la liaison des polices et la typographie web améliorée ;
- l'interactivité avec Ajax et jQuery.

Pour observer l'étude de cas en ligne : roma.cssmastery.com. Les fichiers sources sont également disponibles sur le site www.pearson.fr.

À propos de cette étude de cas

Roma Italia est un site web fictif créé spécialement pour cette étude de cas (la Figure 10.1 présente la page d'accueil). Les techniques CSS qui y sont employées sont en revanche tout sauf artificielles. Chaque technique a été soigneusement sélectionnée dans le but de proposer un choix robuste de méthodes CSS avancées, dont un grand nombre peuvent être appliquées dans des environnements de production. D'autres techniques expérimentales, qui ne sont pas encore uniformément prises en charge dans les navigateurs courants, sont proposées afin d'illustrer les possibilités du futur.

Le site de cette étude de cas, qui se présente comme un guide pour la ville de Rome, est constitué de deux pages : la page d'accueil et la page de vidéo. Certains liens dans la page d'accueil conduisent à de véritables ressources tandis que d'autres sont des liens morts uniquement utilisés pour la démonstration. Toutes les photographies, les vidéos et le contenu

du site Roma Italia proviennent d'un séjour que j'ai effectué à Rome avec ma femme. À tous points de vue, ce site fictif pourrait être réel si les liens renvoient réellement quelque part.

Je tiens tout particulièrement à remercier Aaron Barker (aaronbarker.net) qui m'a aidé avec quelques-uns des exemples jQuery et Ajax dans cette étude de cas, ainsi que ma femme, Suzanne, pour quelques-unes des photos qui ornent le site.

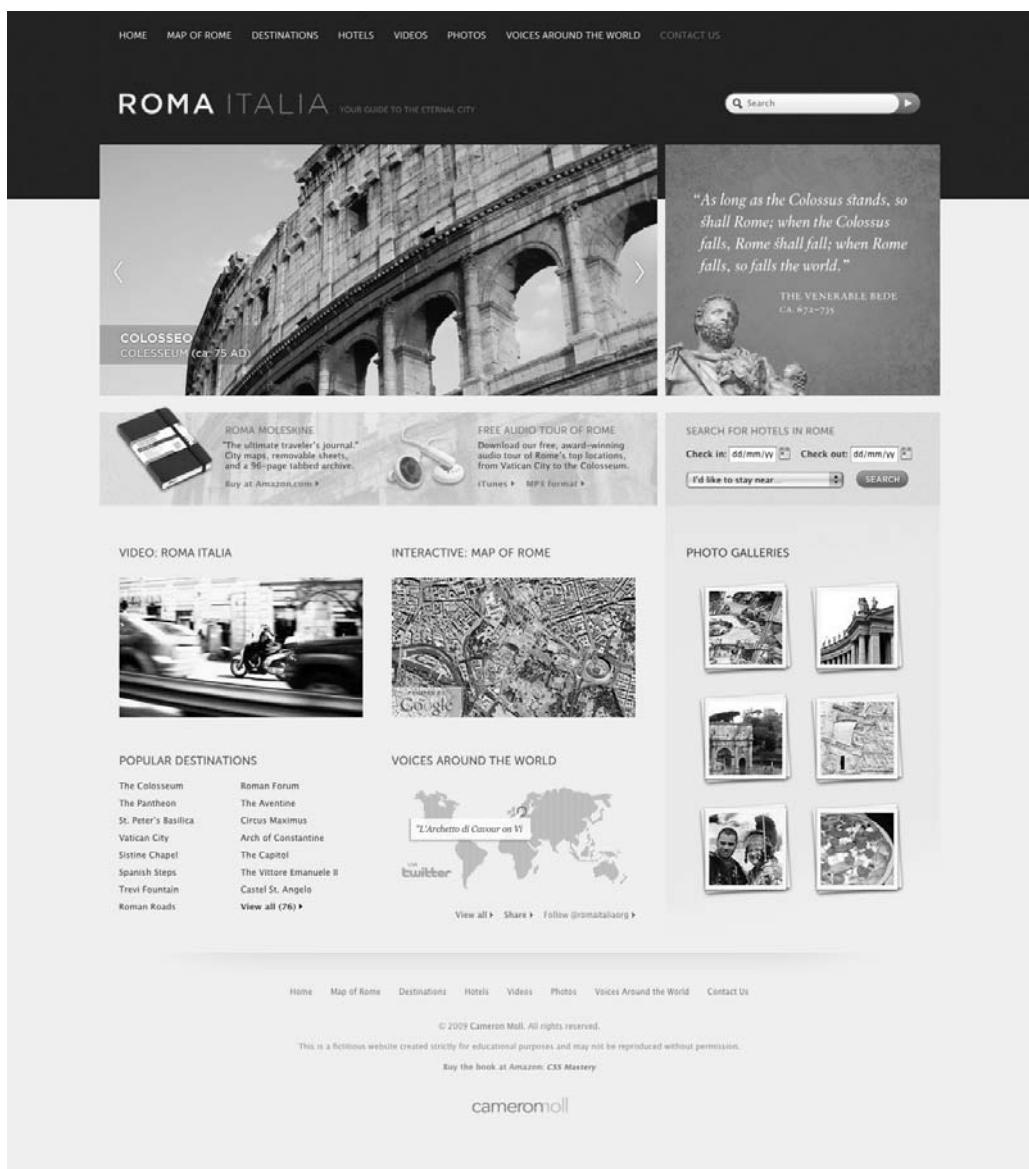


Figure 10.1

La page d'accueil du site Roma Italia.

Les fondations

Lors de la préparation du code HTML, ma principale préoccupation est de rendre ce code aussi signifiant et léger que possible. Par signifiant, j'entends que les noms des éléments HTML et des sélecteurs choisis représentent le contenu de telle manière qu'en supprimant tous les styles, on pourrait comprendre la hiérarchie et la structure du contenu. Ils sont loin les temps où les GIF d'espacement et les éléments br répétés truffaient nos lignes de code ! Ces bizarreries ont été remplacées par des éléments qui représentent de manière logique – ou sémantique – le contenu :

- une liste numérotée des éléments les plus vendus (ol) ;
- le titre principal de la page (h1) ;
- une citation d'un client satisfait (blockquote et cite).

Cette approche nécessite de chasser de son esprit les informations relatives à la présentation, conformément à la philosophie décrite très complètement par Andy Clarke dans son remarquable livre *Transcender CS, Sublimez le design web !* (Eyrolles). Je me souviens encore de mes premières expériences en CSS alors que je programmais une application web à grande échelle. J'étais très fier d'avoir créé une série de noms de classes de présentation qui permettaient de marquer le contenu avec une élégante clarté, comme ceci :

```
<p class="arial red 10">
```

Quelle déconvenue, lorsque je me suis rendu compte qu'il fallait revoir la présentation de l'application et que les dizaines de modèles utilisés devaient inclure à peu près tout sauf des polices Arial rouges de 10 pixels !

Par "léger", je veux dire que le balisage doit être aussi concis que possible (éléments, attributs et valeurs pour le HTML, sélecteurs, propriétés et valeurs pour les CSS). Par exemple,

```
background-color: #c27e28;  
background-image: url(..../img/feature-orange.jpg);  
background-repeat: no-repeat;
```

peut être remplacé par :

```
background: #c27e28 url(..../img/feature-orange.jpg) no-repeat;
```

Vous rencontrerez de nombreux exemples de balisage signifiant et léger dans cette étude de cas. J'en décrirai une partie, mais vous découvrirez la plupart vous-même.

Un œil rivé sur le HTML 5

Pour le balisage signifiant et léger, j'ai choisi de m'en tenir au HTML 4.01 Strict au niveau du DOCTYPE, en le préférant au XHTML 1.0 Strict et au HTML 5. Je vais rapidement expliquer pourquoi.

XHTML 1.0 Strict. Il s'agit du langage que de nombreux professionnels, dont je fais partie, ont utilisé au cours de ces dernières années. Dave Shea propose cependant une argumentation convaincante qui pousse à abandonner le XHTML en prévision du HTML 5 :

"Il y a six ans, bon nombre d'entre nous ont considéré que le XHTML serait l'avenir du Web et que nous entrions désormais dans l'ère du XML. Entre-temps, il est cependant apparu avec évidence, pour moi comme pour d'autres, que le XHTML 2 ne menait véritablement nulle part, en tout cas pas vers les horizons heureux auxquels nous rêvions. Je n'ai pas le courage de m'imposer les contorsions requises pour faire fonctionner encore mes sites avec un DOCTYPE HTML 5, ce qui me laisse avec la version la plus récemment implémentée du langage... En attendant de pouvoir me convaincre que l'heure du HTML 5 est venue, le 4.01 devrait très bien me convenir pour les quatre ou cinq ans à venir" ("Switched", <http://mezzoblue.com/archives/2009/04/20/switched/>).

HTML 5. En un mot, le HTML 5 est la prochaine version majeure du HTML. La bonne nouvelle, c'est que les éléments `div` et `span` sans signification seront remplacés par des éléments plus descriptifs comme `nav`, `header` et `video`.

Au lieu de ce balisage :

```
<div class="header">
  <h1>Page Title</h1>
</div>
```

ou de celui-ci :

```
<object><param/><embed src="http://vimeo.com/3956190"></embed></object>
```

on utilisera donc ceci :

```
<header>
  <h1>Page Title</h1>
</header>
```

et ceci :

```
<video src="http://vimeo.com/3956190">
```

La mauvaise nouvelle, c'est qu'à l'heure où ces lignes sont écrites, le HTML 5 n'est pas correctement pris en charge par les principaux navigateurs (et notamment par Internet Explorer). On estime qu'il faudra de quelques mois à quelques années pour qu'il le soit et devienne donc une option viable pour la création des sites web.

L'autre méthode possible consiste à conserver un œil en direction du HTML 5 en rédigeant le code HTML avec les DOCTYPE actuels, mais en adoptant la sémantique et les noms de classes propres au HTML 5. Jon Tan décrit parfaitement cela dans son article (en anglais) "Preparing for HTML 5 with Semantic Class Names" (<http://jontangerine.com/log/2008/03/preparing-for-html5-with-semantic-class-names>).

Par exemple, avec l'élément `nav`, le balisage HTML 5 donnerait ceci :

```
<nav>
  <ul>
    <li><a href="">Élément de menu 1</a></li>
```

```
...  
</ul>  
</nav>
```

Alors que notre balisage sémantique de style HTML 5 utilisant le HTML 4 ou le XHTML 1 donnerait :

```
<div class="nav">  
  <ul>  
    <li><a href=""> Élément de menu 1</a></li>  
    ...  
  </ul>  
</div>
```

L'inconvénient ici tient au fait que vous pouvez vous retrouver avec une grande quantité de div supplémentaires. Si votre but est de créer un code structuré et léger, la meilleure manière consiste pour le moment à procéder ainsi :

```
<ul class="nav">  
  <li><a href=""> Élément de menu 1</a></li>  
  ...  
</ul>
```

Quelle est donc mon opinion sur le HTML 5 ? Nous nous y adapterons très bien lorsque son heure viendra. Cette reconversion ne nécessitera pas une énorme gymnastique mentale. En attendant, je me contenterai donc de programmer comme nous le faisions jusque-là.

Pour plus d'informations sur le HTML 5, consultez les sites web suivants :

- http://cameronmoll.com/archives/2009/01/12_resources_for_html5/ : douze ressources pour démarrer avec le HTML 5.
- <http://smashingmagazine.com/2009/08/04/designing-a-html-5-layout-from-scratch/> : pour programmer une mise en page HTML 5 de toutes pièces.
- http://fr.wikipedia.org/wiki/HTML_5 : un article Wikipédia sur le HTML 5.
- <http://adactio.com/journal/1540> : l'avènement du HTML 5.
- <http://radar.oreilly.com/2009/05/google-bets-big-on-html-5.html> : où Google mise gros sur le HTML 5.

reset.css

Lorsque j'ai commencé à programmer des sites en CSS, il y a quelques années, il était courant de déclarer quelques styles "globaux" en haut de la feuille de styles maître : body, a img, h1, h2, h3, etc. Ce qui était alors destiné à redéfinir les styles par défaut des navigateurs a finalement évolué jusqu'à devenir la feuille de styles "reset" standard, généralement appelée reset.css.

Comme l'indique l'équipe Yahoo!, la feuille de styles reset "supprime et neutralise l'ensemble hétéroclite des mises en forme par défaut des éléments HTML, afin de créer un

terrain de jeu neutre et uniforme pour tous les navigateurs de niveau A" (<http://developer.yahoo.com/yui/reset/>). Personnellement, je préfère le fichier CSS reset d'Eric Meyer, qui est utilisé dans cette étude de cas. Vous pouvez télécharger cette feuille de styles à l'adresse suivante : <http://meyerweb.com/eric/tools/css/reset/>.

J'utilise une seule feuille de styles (master.css) pour importer toutes les feuilles de styles utilisées dans mes sites. Je déclare en premier la feuille de styles reset, afin que toutes les suivantes puissent redéfinir les styles réinitialisés comme il se doit :

```
@import url("reset.css");
@import url("screen.css");
@import ...
```

Tous les styles pour l'affichage sur écran sont listés dans screen.css. Dans le site d'étude de cas, trois feuilles de styles supplémentaires sont utilisées :

- autocomplete.css contient des styles pour la fonctionnalité de recherche en direct.
- datepicker.css contient des styles pour le calendrier sélecteur de date.
- ie.css, qui est référencée à l'aide de commentaires conditionnels (voir la section suivante), contient des styles propres à Internet Explorer.

Nous aurions facilement pu insérer les styles des fichiers autocomplete.css et datepicker.css dans screen.css, mais afin de mieux vous guider dans cette étude de cas, je préfère les séparer.

La mise en page et la grille à 1 080 pixels

En 2006, j'ai posté un message concernant un dilemme relatif à la largeur optimale des moniteurs dotés d'une résolution de $1\,024 \times 768$ pixels ou plus (voir <http://www.cameron-moll.com/archives/001220.html>). C'est vers cette époque qu'un certain nombre d'entre nous, alors que nous avions l'habitude de développer des sites optimisés pour un affichage en 800×600 pixels depuis un certain temps, ont commencé à explorer des options pour les résolutions à 1 024 pixels.

Dans cet article, j'ai proposé une largeur idéale de 960 pixels pour le passage au-delà des 800×600 . Elle tenait compte du chrome des navigateurs ainsi que du fait que bien des utilisateurs ne naviguent pas en mode plein écran. Plus important encore, elle partait du constat que 960 est un nombre assez magique : il est divisible par 2, 3, 4, 5, 6, 8, 10, 12, 15 et 16. Imaginez donc toutes les possibilités de quadrillages (nous reviendrons sur ce sujet dans un instant) !

Suite à la publication de cet article, la largeur de 960 pixels est quasiment devenue le standard *de facto* pour les mises en page à largeur fixe sur le Web. Des plug-ins Photoshop, des extensions de navigateur et de systèmes d'exploitation l'adoptent même par défaut. Il existe aussi un framework CSS complet établi sur une grille à 960, appelé 960.gs (<http://960.gs/>). Plus de trois ans plus tard, une nouvelle question se pose : est-il temps d'aller au-delà des 960 pixels ? Je ne suis pas certain de la réponse, mais cette étude de cas est l'occasion rêvée d'explorer l'une des voies possibles qui s'ouvrent devant nous.

Avant que ne fusent les critiques des aficionados des largeurs fixes, permettez-moi de vous dire que je suis un grand fan des conceptions fluides contraintes avec `min-width` et `max-width`, comme en témoignent mes "Extensible CSS series" (http://cameronmoll.com/archives/2008/02/the_highly_extensible_css_interface_the_series/) et l'étude de cas de la première édition de ce livre (<http://tuscany.cssmastery.com/>). Les mises en page fluides permettent de réaliser des choses fascinantes, comme l' excellente méthode de mise en page en fonction de la résolution de Cameron Adams (<http://themaninblue.com/writing/perspective/2006/01/19/>) et la technique des images fluides d'Ethan Marcotte (<http://unstop-pablerobotninja.com/entry/fluid-images/>). Je pense néanmoins qu'il y aura toujours des raisons d'utiliser des largeurs fixes et pour être honnête, à bien des égards, les mises en page à largeur fixe sont plus commodes que les fluides.

Si l'on part du principe que nous nous accordons sur le fait qu'il est temps d'engager le débat concernant le dépassement de la limite des 960 pixels, quelle pourrait donc être la largeur idéale ? Voici quelques options :

- 1 020 est divisible par 2, 3, 4, 5, 6, 10, 12, 15 mais pas par 8 ou par 16. Ce n'est pas beaucoup plus large que 960.
- 1 040 est divisible par 2, 4, 5, 8, 10, 16 mais pas par 3, 12 ou 15. Cette largeur est cependant raisonnable et se situe à mi-chemin entre la limite inférieure de 960 et les configurations les plus hautes des utilisateurs qui naviguent en plein écran (beaucoup ne le font pas, comme je l'ai déjà indiqué).
- 1 080 est divisible par 2, 3, 4, 5, 6, 8, 10, 12, 15 mais étrangement, pas par 16. Cette largeur nous pousse vers l'extrême haute du spectre, et la mesure des longueurs de lignes peut devenir un problème si on ne la gère pas correctement.

Il vaut la peine de noter que les divisions entières ne constituent pas l'unique possibilité pour les divisions de la grille, ni forcément le cas idéal. On peut ainsi envisager des divisions proportionnelles, par exemple en utilisant le nombre d'or (http://fr.wikipedia.org/wiki/Nombre_d'or). Mais, comme le signale Jason Santa Maria, les divisions proportionnelles peuvent ne pas être commodes sur le Web car elles demandent de pouvoir observer simultanément les divisions horizontales et verticales, alors que la dernière n'est souvent pas visible entièrement quand on consulte les pages (voir <http://jasonsantamaria.com/articles/whats-golden/>).

Si nous dépassons les 960 pixels, je ne suis donc pas certain que nous aurons cette fois un grand gagnant comme ce fut le cas la dernière fois. Aucune des largeurs indiquées précédemment ne semble aussi flexible que celle de 960 pixels, à tout le moins du point de vue mathématique. Pour cette étude de cas, j'ai cependant choisi une largeur de 1 080 pixels. Elle offre de nombreuses options pour la grille et elle est suffisamment plus large que 960 pixels pour valoir la peine d'être testée.

Utilisation des grilles pour la conception web

Les grilles sont utilisées en conception graphique depuis des décennies, mais ce n'est qu'au cours des dernières années qu'elles ont véritablement reçu les faveurs des concepteurs web – et, enfin, pour de bonnes raisons. Wikipédia propose une description succincte de la grille et de ses mérites : "Une grille topographique est une structure à deux dimensions constituée d'une série d'axes verticaux et horizontaux entrecroisés utilisée pour structurer du contenu. La grille est utilisée comme une armature sur laquelle le concepteur peut organiser le texte et les images de manière rationnelle et intuitive" ([http://en.wikipedia.org/wiki/Grid_\(page_layout\)](http://en.wikipedia.org/wiki/Grid_(page_layout))).

Les grilles sont composées d'éléments tels que des colonnes, des lignes, des marges, des gouttières (l'espace entre les colonnes ou entre les lignes), de lignes de flux (les divisions horizontales) et d'autres composants. Sur un support comme le papier, la largeur et la hauteur de chacun de ces composants sont liées à la taille finale du document ; les dimensions peuvent aisément être calculées par le graphiste. Dans le cas de la conception web, les dimensions de largeur sont plus simples à calculer que les dimensions de hauteur, les pages étant par nature amenées à défiler.

L'armature du site Roma Italia se focalise donc sur les divisions verticales. La Figure 10.2 présente la grille du site.



Figure 10.2

La grille à 12 divisions utilisée pour concevoir le site.

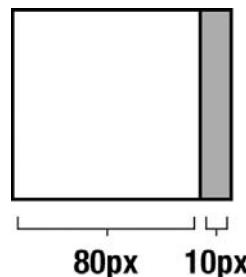
Vous pouvez activer et désactiver cette grille en commentant et décommentant les balises suivantes :

```
<div id="grid"></div>
```

Vous pouvez le faire encore plus facilement avec une extension de navigateur comme Firebug pour Firefox (<http://getfirebug.com/>) qui permet de modifier temporairement le balisage du document dans le navigateur. Lorsque Firebug est ouvert, double-cliquez sur la balise body pour révéler le commentaire et éditer le code HTML directement à cet endroit dans Firebug.

Figure 10.3

Chaque colonne fait 80 pixels de large avec une gouttière de 10 pixels à droite.



Comme vous pouvez le voir, la grille est divisée en 12 colonnes, chacune faisant 80 pixels de large avec une gouttière de 10 pixels à sa droite (voir Figure 10.3), ce qui produit une mise en page de 1 080 pixels de large. Un décalage de 10 pixels (une gouttière supplémentaire) est ajouté à la marge de gauche pour équilibrer la grille. Ce décalage ainsi que la gouttière le long de la dernière colonne sont cependant invisibles pour l'internaute. On pourrait avancer que la grille fait en réalité 1 070 pixels une fois ces composants invisibles supprimés ou, qu'à l'inverse, elle fait 1 090 pixels lorsqu'ils sont rendus visibles. Quoi qu'il en soit, notre grille s'articule autour d'une mesure globale de 1 080 pixels.

Pour l'essentiel, le texte et les images s'alignent sur les colonnes et les gouttières. C'est évidemment tout l'intérêt d'utiliser une grille. Vous remarquerez que je n'ai pas parfaitement aligné tous les éléments. Il est important de noter ici qu'une grille ne doit pas systématiquement dicter le positionnement exact des éléments. Elle le facilite, mais elle doit laisser au concepteur la liberté de faire les meilleurs choix. Dans *Making and Breaking the Grid* (Rockport Publishers, 2005), Timothy Samara décrit ce principe mieux que moi :

"Il est important de comprendre que la grille, bien qu'elle offre un guide précis, ne doit jamais subordonner les éléments qu'elle contient. Son rôle est de fournir une unité générale, sans inhiber la vitalité de la composition. Dans la plupart des cas, la variété de solutions de mise en page que propose la grille est inépuisable, mais même ainsi, il est judicieux d'enfreindre son schématisme à l'occasion. Le graphiste ne doit pas être contraint par sa grille, mais s'appuyer sur elle pour en tester les limites. Une grille bien établie crée d'infinites possibilités d'exploration."

Ce n'est pas dans une page unique que se manifeste le mieux l'intérêt des grilles, mais dans la composition d'ensemble. Dans une brochure imprimée, la grille sert ainsi à unifier l'emplacement des éléments au fil des pages. Si Roma Italia était un vrai site, toutes ses pages (et pas seulement les deux que vous voyez) s'appuieraient sur la même grille afin d'offrir une continuité visuelle à l'utilisateur et des options infinies de mise en page au concepteur.

Je n'ai ici fait qu'aborder rapidement le sujet des grilles. Vous trouverez de nombreuses ressources sur le site The Grid System (<http://www.thegridsystem.org/>) et dans l'article "Designing with Grid-Based Approach" (en anglais) du site du magazine *Smashing* (<http://www.smashingmagazine.com/2007/04/14/designing-with-grid-based-approach/>).

Fonctionnalités CSS 2 et CSS 3 avancées

Ce n'est qu'à la sortie d'Internet Explorer 7 en octobre 2006 que les instructions présentées à cette section sont devenues à la fois rationnelles et techniquement réalisables. Internet Explorer 7 offrait enfin accès aux nombreuses fonctionnalités excitantes des spécifications CSS 2 et CSS 3 que prenaient déjà en charge Firefox, Safari et quelques autres navigateurs. Parmi les plus importantes, on peut citer `min-width` et `max-width`, le sélecteur d'attribut, le sélecteur de frère adjacent, le sélecteur d'enfants et la transparence alpha dans les images PNG.

Ces fonctionnalités bien prises en charge combinées avec d'autres fonctionnalités un peu moins bien prises en charge permettaient de réaliser des choses véritablement fascinantes. La barre de navigation Apple recréée par John Allsopp en CSS uniquement et sans image en était un exemple (voir http://westciv.com/style_master/blog/apples-navigation-bar-using-only-css). Cette section propose quelques autres exemples, tout aussi intéressants.

Les fonctionnalités CSS 2 et CSS 3 avancées suivantes sont utilisées dans le site Roma Italia :

- sélecteur adjacent ;
- sélecteur d'attribut ;
- box-shadow ;
- opacity ;
- RGBa ;
- content ;
- multicolonnage ;
- text-overflow ;
- arrière-plans multiples ;
- @font-face ;
- `min-/max-width`, `min-/max-height` ;
- transparence alpha dans les images PNG.

Dans cette étude de cas, je traiterai des fonctionnalités suivantes : les sélecteurs d'attributs, box-shadow, RGBa, text-overflow, les colonnes multiples, les arrière-plans multiples et @font-face. Pour les fonctionnalités qui ne sont pas abordées, je me suis efforcé d'insérer des commentaires dans le code afin de vous aider à les découvrir par vous-même. Vous pourrez aussi disposer de l'antisèche CSS suivante, pour la consulter à l'écran ou l'imprimer : <http://cameronmoll.com/articles/widget/cheatsheet.pdf>.

Les sites doivent-ils être identiques dans tous les navigateurs ?

C'est la question que pose, en entier et dans son nom de domaine, le site <http://dewebsites-needtolookexactlythesameineverybrowser.com>... Tapez son nom dans la barre d'adresse et voyez la réponse. Ce site très simple, développé par Dan Cederholm, a circulé sur le Web

en 2008 comme une propagande virtuelle discréditant le mythe selon lequel tous les sites web devaient posséder exactement la même apparence dans tous les navigateurs. C'était une invitation lancée à la communauté du développement web pour qu'elle adopte une méthode plus progressive du balisage au lieu de se contraindre à l'uniformité absolue. En un mot, le site de Dan dénonçait ce réflexe de mépris affiché par les développeurs à l'encontre des variations visuelles.

Dans le site Roma Italia, la différence visuelle la plus frappante concerne la fonctionnalité des arrière-plans multiples. Elle permet d'utiliser plusieurs images d'arrière-plan pour un seul élément, alors qu'actuellement seule une image est autorisée par élément. Les aficionados des bords arrondis s'en frottent les mains.

À l'heure actuelle, Safari est le seul navigateur important à prendre en charge les arrière-plans multiples (c'est intéressant de noter qu'il le faisait déjà dès sa version 1.3, soit en 2005 !). Cela signifie que dans les navigateurs comme Firefox et Internet Explorer, le site possédera une apparence légèrement différente. C'est non seulement intentionnel pour les besoins de cette étude de cas, mais cela démontre aussi qu'il est parfaitement légitime de proposer un affichage distinct dans les différents navigateurs sans que cela ait un effet négatif sur l'expérience générale de l'utilisateur.

Les arrière-plans multiples, que les professionnels du Web célébreront comme une aubaine dès qu'ils seront bien pris en charge, sont faciles à créer en CSS. Vous devez simplement séparer chaque image et ses valeurs par une virgule :

```
background: url(image1.png) no-repeat top left,  
url(image2.png) no-repeat top right,  
url(image3.png) no-repeat bottom left;
```

Les propriétés et les valeurs peuvent aussi être définies séparément :

```
background-color: #000;  
background-image: url(image1.png), url(image2.png), url(image3.png);  
background-repeat: no-repeat;  
background-position: top left, top right, bottom left;
```

Les arrière-plans multiples sont utilisés dans notre site d'exemple à plusieurs endroits (voir Figure 10.4). Notez les différences entre Safari, Firefox et Internet Explorer.

Deux images d'arrière-plan sont utilisées dans l'élément body pour donner au site sa texture d'arrière-plan : la grande image brun foncé et la bande brun clair avec un dégradé léger, respectivement nommées bg-dark.jpg et bg-light.jpg. Le code CSS ressemble à ceci :

```
body {  
background: url(..../img/bg-dark.jpg) repeat-x top center,  
url(..../img/bg-light.jpg) repeat-x 239px left;  
background-color: #f1efe8;  
}
```

Figure 10.4

Les différences dans les images d'arrière-plan entre Safari (à gauche) et Firefox et Internet Explorer (à droite).

Les arrière-plans multiples affichés par Safari



Comme Firefox et Internet Explorer ne prennent pas encore en charge les arrière-plans multiples, si nous conservons le code CSS en l'état, aucune image ne s'affiche. On obtient alors un arrière-plan complètement vide, ce qui n'est pas souhaitable. Pour pouvoir afficher au moins l'image sombre, nous insérons donc la propriété dupliquée suivante au-dessus de la première :

```
background: #f1efe8 url(..../img/bg-dark.jpg) repeat-x;
```

Firefox lit cette propriété et ignore l'autre. Nous répétons la même propriété dans ie.css car Internet Explorer n'aime pas trop ce petit hack que nous venons de fomenter. Le CSS final dans screen.css est le suivant :

```
body {
  background: #f1efe8 url(..../img/bg-dark.jpg) repeat-x;
  background: url(..../img/bg-dark.jpg) repeat-x top center,
              url(..../img/bg-light.jpg) repeat-x 239px left;
  background-color: #f1efe8;
}
```

Soyons clairs : ce n'est pas le moyen le plus efficace d'aboutir au résultat présenté. Pour commencer, nous aurions pu prendre une unique image combinant les deux graphismes, ce qui aurait tout simplement évité d'avoir à utiliser des images d'arrière-plan multiples. Ensuite, nous ajoutons du code supplémentaire pour forcer Firefox et Internet Explorer à afficher au moins une image. Mais le but de ces inefficacités est uniquement d'éliminer les incohérences visuelles insignifiantes d'un navigateur à l'autre et de présenter les arrière-plans multiples. Pour le bien de la collectivité des professionnels du Web, espérons que nous sommes plus près du but qu'il n'y paraît.

Sélecteur d'attribut

Le sélecteur d'attribut évite d'avoir à ajouter une classe ou un ID à un élément en permettant de faire référence à n'importe quel attribut, ou valeur d'attribut, contenu dans l'élément. Il peut être utilisé sur presque tous les éléments qui possèdent des attributs. Par exemple, `img[alt]` cible un attribut tandis que `img[src="small.gif"]` cible une valeur d'attribut. En outre, des valeurs d'attribut similaires peuvent être ciblées à l'aide de chaînes syntaxiques comme `img[src^="sm"]`, qui vise toutes les valeurs qui commencent par le préfixe

"sm" (par exemple, "small"). Pour d'autres exemples, consultez l'article (en anglais) "CSS3: Attribute selectors" (<http://www.css3.info/preview/attribute-selectors/>).

Les sélecteurs d'attributs sont pratiques dans certains cas, mais ils le sont particulièrement avec les formulaires. Si des éléments sont mis en forme avec un connecteur générique comme `input { }`, tous les éléments `input` dans le formulaire le seront également. Cela signifie que si vous souhaitez cibler des champs de texte uniquement ou le bouton d'envoi uniquement, vous allez devoir ajouter des classes et des ID superflus. Le sélecteur d'attribut est donc un moyen propre de cibler des éléments particuliers.

Figure 10.5

Ce champ de recherche utilise deux éléments `input`, chacun ciblé par un sélecteur d'attribut.



Le champ de recherche en haut du site sert d'exemple de sélecteur d'attribut (voir Figure 10.5). Voici le code HTML :

```
<form action="#" method="get" accept-charset="utf-8">
  <fieldset>
    <legend></legend>
    <label for="search-input">Search</label>
    <input type="text" id="search_input" name="search" value="" title="Search">
    <input type="image" name="" src="img/search-go.gif">
  </fieldset>
</form>
```

Ici, nous voulons mettre en forme le champ de texte avec plusieurs propriétés et faire flotter `search-go.gif` à gauche. Les deux éléments `input` que nous allons cibler avec un sélecteur d'attribut sont affichés en gras. Vous remarquerez qu'ils ne portent aucun attribut de classe ou d'ID. Nous pouvons en effet les cibler en utilisant l'attribut `type`, comme ceci :

```
#header form input[type="text"] {
  display: block;
  ...
  background: url(..../img/search-bg.gif) no-repeat;
}

#header form input[type="image"] {
  float: left;
}
```

Tous les éléments `input` qui contiennent l'attribut `type="image"` sont rendus flottants à gauche et tous les éléments `input` contenant l'attribut `type="text"` sont mis en forme comme nous l'avons indiqué. Cette même syntaxe est en outre utilisée dans `jquery.plugins.js` pour ajouter une fonctionnalité jQuery et Ajax :

```
$('#header form input[type="text"]').searchField();
```

Nous reviendrons sur jQuery et Ajax ultérieurement.

box-shadow, RGba et text-overflow

Au centre du site Roma Italia, vers le bas de la page, figure un petit widget intitulé Voices Around the World. Si ce widget était réel, il afficherait dans la carte, les mises à jour sur Twitter (tweets) de tous les utilisateurs qui incluent le hashtag #romaitalia dans leur tweet, en fonction de leur emplacement géographique et de manière aléatoire. Les visiteurs du site pourraient cliquer sur ces citations aléatoires pour accéder à la page du tweet complet, sur le nom d'utilisateur de l'auteur et les tweets d'autres fans de Roma Italia – un moyen d'en apprendre plus sur Rome en temps réel à l'aide de mises à jour Twitter. Les tweets affichés dans l'étude de cas sont fictifs, mais vous pouvez suivre @roma_italia, un véritable compte Twitter que j'ai configuré pour cette étude de cas.

Figure 10.6

Les marqueurs cartographiques apparaissent et disparaissent et affichent du texte supplémentaire quand l'utilisateur les survole.



Les marqueurs qui apparaissent et disparaissent toutes les quelques secondes ont un code assez complexe (voir Figure 10.6) et nous utiliserons leur balisage pour illustrer le fonctionnement des propriétés CSS 3 box-shadow, RGba et text-overflow.

Chaque marqueur est composé de trois parties : le texte du tweet, un arrière-plan blanc avec une ombre portée et une image de cercle pour le marqueur cartographique. Le marqueur est enveloppé dans un élément de liste (li), lui-même abrité sur une liste à puces (ul) contenant l'image d'arrière-plan de la carte du monde :

```
<ul>
  <li class="l1" id="map2" style="top: 61px; left: 53px;"><a href="#">
    <em>Absolutely divine. Don't skip the Il Vittoriano. Its size alone is
    impressive. There's a stunning view from the top.</em></a></li>
  <li>...</li>
</ul>
```

Nous appliquons un style à la liste ul avec les propriétés suivantes :

```
#voices ul {
  position: relative;
  width: 310px;
  height: 178px;
  background: url(..../img/bg-map.gif) no-repeat;
}
```

Vous remarquerez que nous avons défini un positionnement relatif. Il permet de positionner de manière absolue chaque élément de liste par rapport à l'élément ul. Sans cela, l'élément de liste serait positionné de manière relative à un autre élément parent – selon toute vraisemblance, l'élément body (voir le Chapitre 3 pour un rappel sur le positionnement absolu).

Chaque marqueur cartographique est mis en forme en fonction :

```
#voices ul li.11 {  
  position: absolute;  
  padding-top: 16px;  
  background: url(..../img/mapmarker-dot.png) no-repeat 2px top;  
}
```

Le cercle du marqueur cartographique est incorporé sous forme d'image d'arrière-plan, et la classe 11 indique l'emplacement numéro un (le point à droite), alors que 12 indique l'emplacement numéro deux (le point à gauche). Les propriétés d'emplacement qui positionnent chaque marqueur ne sont pas indiquées. Nous positionnons en effet chaque marqueur de manière dynamique lorsqu'il apparaît, en utilisant un style incorporé. Pour ce marqueur particulier, il s'agit de `style="top: 61px; left: 53px;"`. Soit à 61 pixels du haut de l'élément ul et à 53 pixels de sa gauche.

L'arrière-plan blanc sur lequel réside le texte du tweet est légèrement transparent et possède une ombre portée sur les bords gauche, droit et inférieur. Ces deux styles sont produits en utilisant respectivement RGBa et box-shadow :

```
#voices ul li.11 a {  
  display: block;  
  padding-left: 11px;  
  font: 11px/14px Georgia, serif;  
  color: #32312a;  
  -webkit-box-shadow: 0 2px 3px rgba(0, 0, 0, 0.35);  
  -moz-box-shadow: 0 2px 3px rgba(0, 0, 0, 0.35);  
  background-color: rgba(255, 255, 255, 0.78);  
}
```

Les propriétés box-shadow et RGBa sont traitées dans l'étude de cas de Simon au Chapitre 11, aussi référez-vous à ce chapitre pour des explications sur ces deux fonctionnalités. Notez toutefois que l'opacité RGBa diffère d'une autre fonctionnalité CSS 3 appelée opacity. Elle peut être appliquée à une propriété particulière, comme l'arrière-plan, et ne concerne qu'elle. opacity, en revanche, affecte tout dans l'élément qu'elle modifie, comme ceci :

```
#voices ul li.11 a {  
  opacity: 0.35  
  ...  
}
```

Les valeurs pour opacity sont analogues à celles pour RGBa et s'étendent de 0 (entièrement transparent) à 1 (entièrement opaque). J'insiste cependant sur le fait que ce réglage modifie tout l'élément. Si nous l'avions utilisé ici, c'est non seulement l'arrière-plan blanc qui serait opaque à 35 %, mais le texte du tweet également.

Lorsque l'utilisateur survole un marqueur cartographique, son affichage change (voir Figure 10.7).

Figure 10.7

Le texte complet s'affiche en cas de survol.



C'est le moment où intervient `text-overflow`. J'aurais aimé toucher un dollar chaque fois que cette fonctionnalité m'a été utile dans ma carrière – j'écrirais alors ces lignes de la terrasse d'un bungalow aux Bahamas. La bonne nouvelle, c'est qu'aujourd'hui cette propriété est assez bien prise en charge par les principaux navigateurs. Internet Explorer le fait même mieux que Firefox, de même que Safari. `text-overflow` tronque le bloc de texte quand il est trop grand pour tenir dans son élément conteneur. La valeur `ellipsis` ajoute trois petits points (...) au texte tronqué.

Ici, `text-overflow` est utilisé pour chaque marqueur cartographique, afin de limiter le texte à une ligne :

```
#voices ul li.l1 a em {
  white-space: nowrap;
  width: 135px;
  overflow: hidden;
  text-overflow: ellipsis;
  -o-text-overflow: ellipsis;
  -moz-text-overflow: ellipsis;
  -webkit-text-overflow: ellipsis;
}
```

Comme cette fonctionnalité n'est pas officiellement prise en charge par les différents navigateurs (même si tous les principaux navigateurs le font de fait), nous avons ajouté les préfixes `-o-` (Opera), `-moz-` (Mozilla) et `-webkit-` (WebKit). Pour l'effet de survol, nous avons ajouté la pseudo-classe `:hover` à l'élément, changé la hauteur à 72 pixels et attribué la valeur `visible` à la propriété `overflow`.

```
#voices ul li.l1 a em:hover {
  white-space: normal;
  overflow: visible;
  text-overflow: inherit;
  -o-text-overflow: inherit;
  cursor: hand;
  cursor: pointer;
  background: #fff none;
  height: 72px;
  padding-left: 11px;
  padding-bottom: 5px;
  margin-left: -9px;
}
```

Voilà qui complète l'effet : chapeau bas à CSS3.info pour leurs exemples de `text-overflow`, qui ont inspiré la création de cet effet. Vous pouvez découvrir d'autres fonctionnalités CSS 3 sur leur site web à l'adresse <http://www.css3.info/>.

Liaison des polices et typographie web améliorée

Il serait aisé de remplir les pages de ce livre de techniques de typographie pour le Web. Dans le cadre de cette étude de cas, nous nous contenterons des suivantes :

- l'unité px pour `font-size` ;
- la ponctuation hors justification ;
- l'affichage du texte sur plusieurs colonnes ;
- la liaison et l'incorporation des polices.

Définir `font-size` comme en 1999

Pendant des années, px a été le standard implicite pour définir la taille du texte avec `font-size`. Il donnait aux graphistes qui transféraient leurs maquettes de Photoshop (ou d'un autre logiciel) vers le HTML une unité absolue et uniforme pour le texte. À mesure que les développeurs ont pris conscience des problèmes d'accessibilité, la taille de texte relative (em ou %) est cependant progressivement devenue l'unité de choix. Elle permettait aux utilisateurs souffrant d'un handicap visuel, et à vrai dire, à n'importe qui, de modifier la taille de texte du navigateur de manière permanente grâce aux réglages du navigateur ou à la volée, en utilisant les commandes clavier Ctrl++ et Ctrl+- (Windows) ou Cmd++ et Cmd+-.

Conformément à ces pratiques, et jusqu'à récemment encore, tous les principaux navigateurs redimensionnaient le texte en conservant le formatage des autres éléments ainsi que la mise en page. Il s'agissait d'une mise à l'échelle du texte ou d'un zoom texte. Cette adaptation requérait qu'on crée un balisage autorisant le redimensionnement relatif de n'importe quel élément contenant du texte. Par exemple, si une div contenait du texte par-dessus une image d'arrière-plan, il fallait soit répéter l'image quand la div devenait plus grande avec le redimensionnement du texte, soit créer l'image plus grande afin de prévoir cet agrandissement. J'ai traité de ce sujet en détail dans une série d'articles (en anglais) intitulée "The Highly Extensible CSS Interface" (voir http://cameronmoll.com/archives/2008/02/the_highly_extensible_css_interface_the_series/).

Les dernières versions des principaux navigateurs (Safari, Firefox, Google Chrome, Opera et, non, vous ne rêvez pas, Internet Explorer) proposent maintenant par défaut un zoom de page plutôt qu'une mise à l'échelle du texte avec les commandes Ctrl++/- et Cmd++/-. Le zoom de page est un agrandissement qui affecte la totalité de la page (mise en page, formatage et taille de texte) à l'unisson. Les éléments conservent leur taille et leur forme, ce qui réduit considérablement la nécessité d'un redimensionnement du texte. C'est en effet le navigateur qui se charge du redimensionnement relatif.

Que signifie donc tout cela ? Que l'unité px peut à nouveau être considérée comme une valeur viable pour `font-size`. En outre, au niveau du texte, la différence entre les unités absolues et les unités relatives peut être négligeable pour les utilisateurs. Pour vous et moi,

cette différence est en revanche tout à fait considérable. On peut ainsi s'épargner tout un travail pour calculer les unités relatives dans un document CSS ; car tout est plus commode avec les unités absolues : 14 px, c'est 14 px n'importe où dans le document, quels que soient les éléments parents, dont la valeur de `font-size` peut être différente.

Cette étude de cas est censée être réaliste, certes, mais elle est avant tout expérimentale. Je m'offre la liberté de me demander parfois "et si... ?" et "pourquoi ne pas tenter... ?" Vos projets peuvent être plus restrictifs, aussi efforcez-vous de prendre la bonne décision en fonction de votre public cible, comme toutes les décisions des professionnels du Web. C'est la constante absolue pour les programmeurs de sites, l'invariant qui résiste à tous les bouleversements passés, présents et futurs de cette industrie. Autrement dit, si le choix des unités relatives convient à votre projet, il est exclu que quiconque puisse affirmer le contraire – pas même moi.

Pour plus d'informations sur le débat au sujet de l'unité px pour `font-size`, consultez ces articles (en anglais) :

- <http://www.wilsonminer.com/posts/2007/mar/16/problem-pixels/> : le problème avec les pixels.
- http://www.456bereastreet.com/archive/200703/ie_7_does_not_resize_text_sized_in_pixels/ : Internet Explorer 7 ne redimensionne pas le texte spécifié en pixels.
- <http://mezzoblue.com/archives/2008/10/07/zoom/> : Mezzoblue – Zoom.
- <http://orderedlist.com/articles/hello-old-friend> : Hello Old Friend.

Ponctuation hors justification

La ponctuation hors justification, quand on la découvre, est le signe évident qu'un typographe expérimenté œuvre à l'arrière-plan. Cette fonctionnalité est disponible dans la plupart des applications de conception d'Adobe, mais elle n'est pas disponible sous forme de propriété CSS. Du moins, pas encore. Une propriété appelée `hanging-punctuation` est bien proposée dans la spécification CSS 3 (voir <http://www.w3.org/TR/css3-text/#hanging-punctuation>), mais, à ma connaissance, aucun navigateur ne la prend encore en charge.

La ponctuation hors justification aligne les marques de ponctuation en dehors du bloc de texte afin de ne pas interrompre l'enchaînement visuel du texte. La Figure 10.8 en présente un exemple avec des guillemets.

Cette technique est utilisée à trois endroits dans la page d'accueil : la première ligne de texte sous le titre Roma Moleskine (voir Figure 10.9), le texte du tweet dans les marqueurs de la carte Voices Around the World et pour la citation orange par Bède le Vénérable. La dernière est une image, aussi commençons par elle. La même technique est utilisée pour la seconde.

Figure 10.8

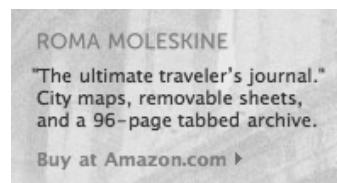
Cet exemple montre comment la ponctuation hors justification (en bas) diffère de la ponctuation alignée avec le bord du texte (en haut). Cette dernière est utilisée par défaut dans la plupart des logiciels de conception graphique, ainsi que pour le texte dans les navigateurs.

“This text does not use hanging punctuation. Notice how the quotation mark aligns with the left edge of the text block.”

“This text uses hanging punctuation. Notice how the entire text block aligns with the left edge, while the quotation mark sits outside it.”

Figure 10.9

Exemple de ponctuation hors justification dans le site Roma Italia.



Le code HTML est très simple :

```
<div id="featurette1">
  ...
  <p>
    &ldquo;The ultimate traveler&rsquo;s journal.&rdquo; City maps,
    removable sheets, and a 96-page tabbed archive.</p>
  ...
</div>
```

Les entités HTML “, ’ et ” correspondent à des guillemets courbes. Ils ne sont pas nécessaires pour la ponctuation hors justification, mais il s'agit d'une autre marque subtile de typographie de qualité. Ces entités chargent un peu le code source pour l'œil qui n'y est pas exercé, mais la ponctuation ajustée reproduite par le navigateur (et que l'œil entraîné sait repérer) le compense largement.

C'est dans le code CSS que la magie se produit :

```
#featurette1 p {
  text-indent: -.3em;
}
```

Et voilà. Dans vos projets, ajustez cette valeur en fonction de la taille et de la famille de police.

Affichage du texte sur plusieurs colonnes

Avec une mise en page aussi large que celle-ci, il devient important de conserver une "mesure" (longueur de ligne) correcte pour la lisibilité du texte. La largeur du bloc de texte se mesure par son nombre de caractères (dont les espaces) par ligne. Il existe un nombre infini d'études et d'opinions concernant le nombre optimal de caractères par ligne, qu'on fait varier de 45 à 95, notamment en fonction du support. Cette section ne traite pas véritablement de la longueur de ligne optimale, mais plutôt de la manière de conserver une longueur raisonnable.

Comme cette mise en page atteint 1 080 pixels de large, elle est l'occasion toute trouvée de tester la fonctionnalité de texte multicolonne que propose la spécification CSS 3, actuellement prise en charge par les navigateurs WebKit et Mozilla. Les autres navigateurs affichent le texte dans une seule colonne dont la largeur égale celle de toutes les colonnes combinées.

Robert Bringhurst, dans son livre exceptionnel truffé de conseils typographiques *The Elements of Typographic Style* (Hartley and Marks, 2004), suggère d'utiliser une longueur de ligne comprise entre 40 et 50 caractères pour le texte sur plusieurs colonnes. Par commodité, j'ai suivi cette recommandation pour ma mise en page.

La Figure 10.10 présente un fragment de video.html.

Cameron Moll
Add as contact

Highlights from a recent vacation to Rome, Italy. Shot with a Canon HG10, edited with Final Cut Express. Music is First Breath After Come by Explosions in the Sky. Typeface is Gotham by Hoefler & Frere-Jones.

I'm no video virtuoso. I only dabble with video as time allows, which it usually doesn't. But this is one short film I managed to get lucky with. I had a passion for film production ever since I was young. I saw myself first as a stuntman, then later as a film score composer. My time has passed on the first, but I hope I still have a shot at the second sometime in life. In more recent years, I've found myself behind the camera and in the editor's chair. I have much to learn about what it takes to shoot and edit and a great film. If there's one thing I've done right since purchasing my Canon HG10 last year, it's to shoot far more film than I think I'll need. That's how I got lucky with "Roma Italia". I shot more angles than I figured I'd need, and I left the camera running longer than I assumed was adequate. It paid off in the end, leaving me with five minutes of solid shots from a total of roughly two hours of footage. I had no script for this piece, and there were only a few shots I planned beforehand to use in whatever the final film would become. The rest was editing mojo. I analyzed the material I had, pieced it together in way I felt told a compelling story, and added effects that complimented the storytelling. "First Breath After Come", a moving and driving piece by instrumental rock band Explosions in the Sky, was a perfect fit. Not only is it the best song title ever, but its story matches that of "Roma Italia": One of awakening, surprise, and climatic rush and release.

In the end, I hope you enjoy the story. It's a chance to see the magnificent city of Rome through my eyes. Perhaps it'll encourage you to see it through yours.

Tags
rome (171)
roma (189)
italy (461)
italia (393)
Add a tag
Uploaded 2 weeks ago.

Figure 10.10

Mise en page multicolonne dans la page Video.

Vous remarquerez que le texte est disposé en deux colonnes. Le code HTML est tout ce qu'il y a de plus classique :

```
<div id="main-video">
  <h3>
    Highlights from a recent vacation to Rome, Italy. Shot with a...
  </h3>
  <p>I'm no video virtuoso. I only dabble with video as time...
  ...
</div>
```

Le code CSS, à l'inverse, est loin d'être standard :

```
#main-video {  
    float: left;  
    margin: 40px 10px 70px;  
    width: 520px;  
    -moz-column-count: 2;  
    -moz-column-gap: 20px;  
    -webkit-column-count: 2;  
    -webkit-column-gap: 20px;  
}
```

Ici encore, nous nous trouvons contraints d'utiliser les préfixes `-moz-` (Mozilla) et `-webkit-` (WebKit). Vous remarquerez que deux propriétés entrent en jeu : `column-count` et `column-gap`. Elles sont assez simples à comprendre et à utiliser ; vous n'avez qu'à définir une valeur pour le nombre de colonnes souhaité et sélectionner une valeur pour l'interstice qui les sépare. Le bloc de texte entier est ensuite automatiquement réparti dans le nombre de colonnes spécifié. Une troisième propriété, `column-rule`, permet d'ajouter une bordure entre les colonnes (par exemple, `column-rule: 1px solid #000;`).

La question se pose, certes, du bien-fondé du texte multicolonne sur le Web et des problèmes liés à la nécessité de faire défiler la page tantôt vers le haut, tantôt vers le bas, mais je ne doute pas qu'entre les mains d'un typographe expérimenté l'affichage sur plusieurs colonnes permettra d'étendre les options typographiques dont nous disposons sur le Web.

@font-face

Ce sujet ne saurait être mieux présenté qu'en reprenant les termes de Jeffrey Veen, fondateur de Typekit (<http://typekit.com/>) :

"La recommandation du W3C pour les polices Web CSS [`@font-face`] va bientôt fêter son septième anniversaire. Pourquoi, après tant d'années, la typographie web n'a-t-elle pas plus progressé ? Pourquoi les concepteurs n'ont-ils pas adopté les polices liées et téléchargeables dans leurs maquettes ?" (<http://blog.typekit.com/2009/06/02/fonts-JavaScript-and-how-designers-design/>).

Excellent questions, Jeffrey. Il y a fort à parier que d'ici à ce que ce livre atteigne les étagères des librairies et se fraie finalement un chemin jusqu'à vos mains, le produit Typekit de Jeffrey aura en grande partie répondu à cette question, justement. Typekit tente de résoudre les problèmes d'implémentation et de sécurité (traités plus loin dans cette section) avec `@font-face` en hébergeant de manière centralisée des polices qui ont déjà été approuvées par les éditeurs de polices pour la liaison (voir Figure 10.11).

Figure 10.11

Typekit propose une liaison de polices web sans le fardeau de l'implémentation `@font-face` et ses problèmes de sécurité afférents.



En un mot, `@font-face` offre la possibilité d'utiliser virtuellement n'importe quelle police dans les maquettes reproduites sous forme de texte HTML sans se soucier de savoir si cette police est ou non installée sur l'ordinateur de l'utilisateur. Ce procédé est généralement appelé *liaison* ou *incorporation* des polices. Au lieu de placer ceci en haut du document :

```
body {
  font-family: Georgia, serif;
  ...
}
```

Il est possible d'écrire ceci :

```
@font-face {
  font-family: "Garamond Premier Pro";
  src: url(fonts/GaramondPremrPro.otf);
}
```

Ensuite, il suffit de faire référence à la famille de police, comme d'habitude :

```
h1 {
  font-family: "Garamond Premier Pro", serif;
}
```

Je frétille rien qu'en tapant ça (je sais, je suis un vrai *geek*). Mais quand même : imaginez, utiliser n'importe quelle police dans la maquette de votre site en laissant votre texte s'afficher en HTML, sans sIFR, sans Cufón, sans images ! Ah, vous voyez, ça vous chatouille, vous aussi, maintenant...

Bon, évidemment, si c'était si facile que cela, on se serait précipité pour utiliser @font-face il y a sept ans. Mais il y a quelques difficultés. La première, vous l'aurez deviné, est la prise en charge des navigateurs. Safari 3 et Firefox 3.1 ainsi que leurs versions ultérieures prennent en charge @font-face. Internet Explorer, qui le reconnaît depuis sa version 4, ne reconnaît cependant que le format .eot (*Embedded OpenType*), un format de police propriétaire de Microsoft.

Toutefois, Internet Explorer ne prend en charge que le format .eot (*Embedded OpenType*), qui est un format de police propriétaire de Microsoft. Les fichiers .eot ne peuvent être créés qu'à partir de fichiers .ttf (*TrueType*) et les autres formats de police comme .otf (*OpenType*) doivent être convertis en .ttf pour être ensuite convertis en .eot. Il n'est donc pas étonnant que @font-face ait eu tant de mal à décoller.

Ensuite, les éditeurs de polices et les vendeurs ont été très circonspects concernant la liaison des polices sur le Web. Leur préoccupation était double : comme les fichiers de police sont stockés sur le site et donc publiquement accessibles, ils sont exposés à des téléchargements et à un usage illégaux et parce que bon nombre de leurs contrats de licence d'utilisation n'ont pas été mis à jour pour autoriser la liaison sur le Web.

Il y a toutefois une bonne nouvelle, double, elle aussi : de nouvelles technologies émergent, comme Typekit, qui résolvent les deux problèmes mentionnés et @font-face encourage l'utilisation d'autres polices que le jeu standard auquel nous sommes habitués (Arial, Georgia, etc.), ce qui doit inévitablement augmenter la demande commerciale en matière de polices. Les vendeurs et les éditeurs de polices ont donc un réel intérêt à ce que la liaison et l'incorporation des polices finissent par décoller. En fait, pendant que j'écrivais ce chapitre, plusieurs éditeurs ont annoncé que de nouvelles polices étaient disponibles pour la liaison sur le Web et certains ont même annoncé d'importantes modifications dans leurs contrats de licence d'utilisation.

Museo Sans (voir Figure 10.12) est une police conçue par Jos Buivenga et commercialisée en 2008. La version Museo Sans 500 est gratuite et il s'agit du corps que j'ai utilisé ici. Mieux, le contrat de licence d'utilisation (EULA) autorise la liaison sur le Web (notez que Gotham, la police utilisée dans le logo et dans les légendes en semi-transparence sur les images du diaporama, aurait été mon premier choix pour la liaison des polices, mais son contrat de licence, au moment où je concevais le site, ne l'autorisait pas).

Figure 10.12

Exemples de police Museo. (Copyright MyFonts.com.)

The Oneironauts
Inducing lucid dreams
Run 32 miles
SACRED GEOMETRY
Picking parts

Dans Roma Italia, j'utilise @font-face pour montrer ce qui est possible à présent et dans un avenir proche. Dans screen.css, vous verrez le code suivant vers le haut du document :

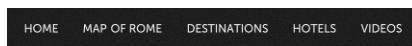
```
@font-face {  
    font-family: "Museo Sans X";  
    src: url(..../fonts/MuseoSans_500.otf);  
}
```

Museo Sans est utilisée dans plusieurs titres et dans le système de navigation du haut (voir Figure 10.13):

```
#home h3, #home h4, #home #header h2, #home #header ul a {  
    font-family: "Museo Sans X", "Lucida Grande", "Lucida Sans Unicode",  
    Arial, sans-serif;  
}
```

Figure 10.13

Museo Sans est la police utilisée dans le système de navigation du haut.



Vous remarquerez que j'ai indiqué des choix de polices de repli dans l'éventualité où le navigateur de l'utilisateur ne prendrait pas en charge @font-face. Vous remarquerez aussi que le nom de la police est Museo Sans X. Lorsque vous définissez @font-face dans les CSS, vous pouvez nommer la famille de police comme vous le souhaitez. J'aurais pu utiliser Musei Vaticani si cela m'avait chanté, dès le moment où je faisais référence au bon fichier de police (MuseoSans_500.otf). Comme Museo Sans est gratuite, il se peut qu'elle soit déjà installée sur votre ordinateur. J'ai spécifiquement ajouté X afin de m'assurer que vous voyiez fonctionner @font-face en pratique, avec ma copie de Museo Sans et non avec une copie qui serait déjà installée sur votre ordinateur.

Je n'ai pas converti le fichier Museo Sans.otf en un fichier .eot. Internet Explorer ne le reconnaîtra donc pas. Si vous voyez Museo Sans dans Internet Explorer, c'est que Cufón est à l'œuvre et non @font-face (voir la section suivante pour plus d'informations à ce sujet).

Quelques autres ressources à signaler sur ce thème :

- <http://myfonts.com/fonts/exljbris/museo-sans/> : pour télécharger une copie de Museo Sans.
- <http://jontangerine.com/log/2008/10/font-face-in-ie-making-web-fonts-work> : pour une présentation complète de @font-face et d'EOT, consultez l'article de Jon Tan (en anglais) intitulé "@font-face in IE: Making Web Fonts Work".

Cufón, étape intermédiaire avant `@font-face`

J'ai posté un didacticiel complet concernant Cufón sur mon site web personnel que vous pouvez consulter (en anglais) à l'adresse suivante : http://cameronmoll.com/archives/2009/03/cufon_font_embedding/. Je ne traiterai donc que brièvement de Cufón ici. En un mot, Cufón permet de reproduire du texte HTML avec la police de son choix sans avoir à utiliser d'image ni `@font-face` (voir Figure 10.14).

Commençons par le commencement. sIFR, vous le savez, est un moyen de remplacer "de courts passages de texte brut par du texte affiché avec la police de votre choix, que vos utilisateurs l'aient installée ou non sur leur ordinateur", grâce à une combinaison de Flash et de JavaScript (voir <http://www.mikeindustries.com/blog/sifr/>). Shaun Inman, Mark Wubben, Mike Davidson et quelques autres ont consacré de nombreuses heures à développer et à affiner IFR et sIFR. Nous leur sommes tous très reconnaissants d'avoir fait avancer à pas de géant la typographie sur le Web. Ce qui manquait à `@font-face` en termes de prise en charge des navigateurs et d'adhésion de la part des éditeurs de polices au fil des ans, sIFR l'a comblé dans le même temps.

La partie Flash de ces technologies les rend cependant difficiles à configurer et à utiliser pour nombre d'entre nous. Cufón, à la différence, peut être configuré et exécuté sur votre site en cinq minutes environ. C'est la raison pour laquelle je le considère comme une bonne étape intermédiaire entre sIFR et `@font-face` si vous n'avez pas la possibilité de réaliser des liaisons de polices.

Figure 10.14

Le générateur d'écriture de Cufón.



Voici comment vous pouvez vous appuyer sur Cufón :

1. Téléchargez le fichier de script Cufón à l'adresse <http://wiki.github.com/sorccu/cufon>.
2. Placez sur votre serveur la police de votre choix à l'aide du générateur Cufón, qui fournit un second fichier de script.
3. Dans l'en-tête de votre document, ajoutez des références au script Cufón et le script de police fourni par le générateur, comme ceci :

```
<script src="js/cufon-yui.js" type="text/JavaScript" charset="utf-8"></script>
<script src="js/Museo_400.font.js" type="text/JavaScript" charset="utf-8"></script>
```

Ajoutez aussi ceci juste avant la balise body de fermeture pour éviter un problème de clignotement dans Internet Explorer :

```
<script type="text/JavaScript">Cufon.now();</script>
```

Dans l'en-tête, indiquez aussi quels éléments HTML ou sélecteurs doivent être remplacés par votre police, comme :

```
<script type="text/JavaScript">
  Cufon.replace('h1');
</script>
```

ou

```
<script type="text/JavaScript">
  Cufon.replace('h1')('h2')('blockquote');
</script>
```

4. Sinon, si vous utilisez un framework JavaScript comme jQuery sur le site où Cufón sera employé (comme Roma Italia), Cufón tire parti du moteur de sélecteur de ce framework de telle sorte que vous pouvez appeler des sélecteurs spécifiques comme ceci :

```
<script type="text/JavaScript" charset="utf-8">
  Cufon.replace('#header h2,#header ul a');
</script>
```

5. Dans les fichiers CSS, modifiez le texte remplacé par Cufón de la même manière que vous le feriez avec n'importe quel autre texte – `color: #333;`, `font-size: 12px;`, `text-transform: uppercase;`, etc.

Et c'est tout. Cufón est actuellement pris en charge par Internet Explorer 6, 7 et 8, Firefox 1.5 et ultérieur, Safari 3 et ultérieur, Opera 9.5 et ultérieur et Google Chrome. Dans le site d'étude de cas, j'ai inclus Cufón en plus de `@font-face`, afin que vous puissiez observer les deux options. Notez que Cufón est soumis aux mêmes problèmes que `@font-face` concernant les licences d'utilisation : les polices que vous choisissez doivent autoriser l'incorporation sur le Web.

Vous remarquerez que j'ai enveloppé `Cufon.replace` dans un commentaire conditionnel, car Internet Explorer ne lit pas le fichier de police .otf que nous utilisons pour `@font-face`. Cufón devient donc un remplacement pour `@font-face` dans Internet Explorer pour les besoins de cette étude de cas. Si vous souhaitez voir Cufón fonctionner dans un autre navigateur, supprimez tout simplement le commentaire conditionnel. Cufón vient alors en remplacement de `@font-face`.

Interactivité avec Ajax et jQuery

Lorsque j'ai abordé le sujet d'Ajax au cours d'un atelier de travail il y a quelques années, je n'ai pas vu beaucoup de mains se lever quand j'ai demandé qui avait déjà développé des sites et des applications avec Ajax. Si je devais poser la même question aujourd'hui, il est probable que vous seriez nombreux à lever le bras. Il en irait sans doute de même si l'on posait cette question à propos de jQuery.

Ajax et jQuery ont en effet trouvé leur place dans les sites de haut niveau comme le duo de choix pour l'interactivité riche sur le Web. Les deux peuvent évidemment s'utiliser l'un sans l'autre, mais il est courant de les employer en tandem. Des technologies émergentes, comme Flex d'Adobe ou Silverlight de Microsoft prétendent contester la suprématie de ce duo, mais j'ai toutes les raisons de croire qu'Ajax et les frameworks resteront les principaux acteurs du Web pour quelques années encore.

Cette section n'a pas pour but de les présenter en détail. Il existe de nombreux livres, didacticiels et articles de blog sur le sujet. J'aimerais plutôt proposer une brève introduction (ou un rappel) à Ajax et jQuery et montrer comment ils sont utilisés dans le site Roma Italia. Si vous êtes déjà familiarisé avec ces technologies, passez directement à la section "Utilisation d'Ajax et de jQuery pour la fonctionnalité de recherche".

Ajax

Ajax – l'acronyme approximatif d'*Asynchronous JavaScript and XML* (JavaScript et XML asynchrones) – inclut généralement au moins trois composants :

- une communication asynchrone avec le serveur, réalisée à l'aide de XMLHttpRequest ;
- une manipulation du DOM (*Document Object Model* ou modèle objet de document) pour l'affichage dynamique et l'interaction ;
- du JavaScript pour ficeler le tout.

L'aspect asynchrone de la requête est le composant-clé d'Ajax (et de toute technologie Internet riche en la matière), car c'est lui qui donne, dans l'environnement web, cette impression de travailler sur une application native extrêmement réactive. Au lieu du modèle classique de requête/réponse qui récupère la page entière en opérant un aller-retour complet vers le serveur, l'asynchronie se contente de n'extraire que les données d'une portion sélectionnée de la page (par exemple, en vérifiant la disponibilité d'un nom d'utilisateur lors de l'ouverture d'un compte).

Dans le site Roma Italia, nous avons simulé la communication asynchrone avec le serveur pour les besoins de la démonstration, en utilisant du code JavaScript et en extrayant les données de quelques pages PHP statiques :

- imageLoad.php pour les images du diaporama ;
- search.php pour la fonctionnalité de saisie semi-automatique du champ de recherche.

La simulation de l'asynchronie permet de télécharger les exemples de code et d'ouvrir l'interface sur n'importe quel ordinateur qui exécute PHP, sans avoir besoin d'une véritable communication serveur.

Nous examinerons le code de la fonctionnalité de saisie semi-automatique juste après la section sur jQuery.

jQuery

Le très utile livre *jQuery* de Karl Swedberg et Jonathan Chaffer (Pearson France, 2009) décrit jQuery comme "une couche d'abstraction de portée générale pour la programmation web courante". Dans mes termes, cela donnerait plutôt : "le JavaScript pour les bleus dans mon genre..." .

jQuery permet de :

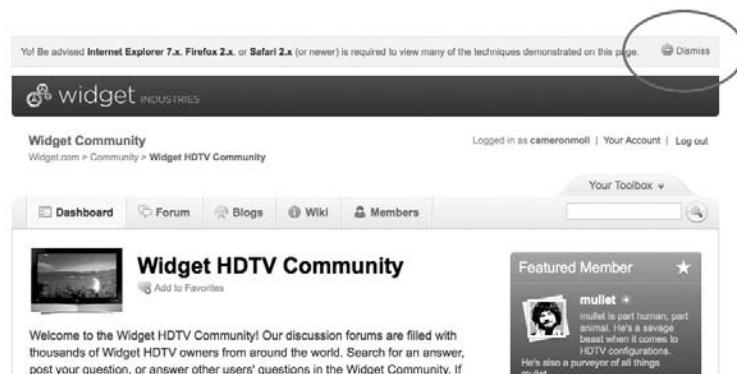
- parcourir le DOM ;
- modifier l'apparence d'une page ;
- modifier le contenu d'une page de manière dynamique.

Il permet en outre de réaliser tout cela sans écrire des kilomètres de JavaScript. Mieux, il exploite la syntaxe CSS pour utiliser les sélecteurs dans les documents comme points d'attache pour ses opérations interactives.

Examinons un exemple d'un autre site que j'ai programmé afin de comprendre les composants de jQuery. Il s'agit d'un site fictif créé à des fins éducatives qu'on peut consulter à l'adresse <http://cameronmoll.com/articles/widget/>.

Figure 10.15

Le site de démonstration Widget créé pour une série d'articles rédigés sur l'extensibilité de l'interface CSS.



Nous allons utiliser le bouton *Dismiss* situé dans la barre de notification jaune en haut de la page (voir Figure 10.15). Lorsqu'on clique dessus, la barre jaune disparaît lentement en glissant vers le haut.

Voici le code ajouté à la balise d'ancre du bouton :

```
$( '#alert' ).slideUp('slow');
```

Voici une description de chaque composant :

- **`$()`**. Cette construction (ou fonction) jQuery de base est utilisée pour sélectionner des parties du document. Dans cet exemple, nous sélectionnons un élément dont l'ID est `alert`.
- **`.slideUp`**. L'une des nombreuses méthodes jQuery, qui sont, en réalité, des raccourcis de longs blocs de code JavaScript. La méthode `slideUp` fait glisser l'élément que nous ciblons (`#alert`) vers le haut.
- **`('slow')`**. Cette chaîne prédefinie détermine le fonctionnement de la méthode. Ici, elle indique que l'élément doit glisser lentement.

Ce code peut être ajouté directement dans le code ou dans un fichier .js séparé (ou incorporé de manière dynamique). La seconde méthode est préférable. Mais attendez la suite : je n'ai pas eu à programmer le moindre code de mon côté. La construction, la méthode et la chaîne étaient toutes prédefinies dans jQuery ! Il me suffisait de savoir que l'élément devait glisser lentement vers le haut. J'ai ensuite retrouvé les références appropriées dans la bibliothèque jQuery qui correspondaient à l'animation et au mouvement que je recherchais. Point final.

Utilisation d'Ajax et de jQuery pour la fonctionnalité de recherche

Maintenant que nous nous sommes occupés du ménage au niveau des instructions, considérons un exemple d'Ajax et de jQuery combinés dans le site Roma Italia. La fonctionnalité que nous allons explorer concerne le champ de recherche. Nous avons précédemment vu comment le sélecteur d'attribut pouvait être utilisé pour cibler des éléments spécifiques enclos dans le formulaire pour le champ de recherche. À présent, nous allons examiner les autres composants en action.

Le code de la fonctionnalité de recherche est, en fait, l'un des plus complexes du site. Lorsque l'utilisateur tape une requête de mot-clé, l'icône de la loupe est remplacée par une icône de chargement et les résultats correspondants sont affichés. On appelle parfois cette fonctionnalité *suggestions automatiques* (ou autocomplétion). Ajax et jQuery œuvrent d'arrache-pied à l'arrière-plan pour articuler ce mécanisme. Malgré sa complexité, l'assemblage des composants est plus simple qu'il n'y paraît.

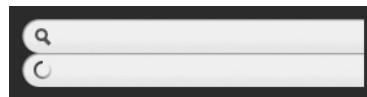
Tout d'abord, voici le code HTML :

```
<form action="#" method="get" accept-charset="utf-8">
  <fieldset>
    <legend></legend>
    <label for="search-input">Search</label>
    <input type="text" id="search-input" name="search" value="" title="Search">
    <input type="image" name="" src="img/search-go.gif">
  </fieldset>
</form>
```

Pour remplacer l'icône par l'icône de chargement, la manière la plus légère (que je n'ai pourtant pas encore vue sur d'autres sites) consiste à combiner l'image d'arrière-plan pour le champ de saisie et l'icône de chargement dans une unique image GIF animée (voir Figure 10.16). Le code CSS est ensuite utilisé pour positionner l'image en fonction de l'état de l'interaction, en la décalant vers le haut ou vers le bas quand l'utilisateur tape dans le champ, afin de permutez entre l'icône de la loupe et l'icône de chargement.

Figure 10.16

search-bg.gif, un unique GIF animé qui inclut deux états.



Voici le fichier CSS pour le champ de saisie :

```
#header form input[type="text"] {  
    ...  
    padding: 6px 0 0 28px;  
    height: 20px;  
    background: url(..../img/search-bg.gif) no-repeat;  
}
```

Par défaut, l'image d'arrière-plan est positionnée en haut à gauche et nous restreignons la hauteur à 20 pixels. Ajoutez-y 6 pixels de remplissage supérieur et cela donne 26 pixels de l'image qui s'affichent – soit exactement la moitié de la hauteur de l'image. Seule la portion correspondant à la loupe apparaît.

Lorsque l'utilisateur commence à taper, plusieurs choses se passent. Tout d'abord, chaque fois qu'il tape un caractère, quatre fichiers sont concernés : autocomplete.css, jquery.plugins.js, jquery.autocomplete.js et search.php. Lorsque la saisie commence, `class="ac_loading"` est ajouté dynamiquement au champ de saisie puis supprimé lorsque la recherche automatique affiche les résultats. Ce sélecteur se trouve dans autocomplete.css, où il reçoit le style suivant :

```
.ac_loading {  
    background: url(..../img/search-bg.gif) no-repeat 0 -26px !important;  
}
```

Vous remarquerez que l'image d'arrière-plan est maintenant positionnée à -26 pixels du haut, ce qui la décale vers le haut et révèle sa seconde moitié (l'icône de chargement). Nous indiquons ainsi à l'utilisateur que des données sont récupérées de manière asynchrone auprès du serveur (search.php).

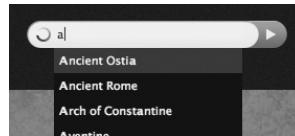
Ensuite, pendant que l'icône de chargement tournoie, des données sont échangées avec search.php afin que soient localisés les résultats correspondant aux caractères que l'utilisateur a tapés. Si vous ouvrez search.php, vous trouverez certains des termes avec lesquels j'ai rempli le fichier : Ancient Ostia, Ancient Rome, Arch of Constantine, etc.

Les résultats correspondants sont renvoyés à la page et un petit menu de sélection est affiché sous le champ de saisie avec la liste des résultats (voir Figure 10.17). Ce menu, qui n'est

qu'une simple liste à puces (ul), est généré par une combinaison de jquery.autocomplete.js et jquery.plugins.js et mis en forme par autocomplete.css. L'utilisateur peut ensuite sélectionner une correspondance avec la souris ou le clavier ou continuer à taper et appuyer sur la touche Entrée. L'interaction s'arrête là.

Figure 10.17

La fonctionnalité de recherche complète en action.



jquery.autocomplete.js inclut plusieurs centaines de lignes de code, mais voilà toute la beauté de jQuery : je n'en ai pas écrit une seule. Il s'agit d'un plug-in jQuery conçu par la communauté. Or, il en existe bien d'autres du même genre. En fait, l'essentiel des exemples de code jQuery de cette étude de cas provient de plug-ins. Le plug-in de saisie semi-automatique, jQuery Autocomplete, peut être trouvé à l'adresse <http://bassistance.de/jquery-plugins/jquery-plugin-autocomplete/>.

Pour obtenir d'autres ressources et des didacticiels, consultez les sites web suivants :

- <http://bulletproofajax.com/> : Bulletproof Ajax par Jeremy Keith.
- <http://ajaxian.com/> : Ajaxian.com.
- <http://dhtmlsite.com/ajax.php> : DHTML site, didacticiels Ajax (en anglais) et scripts.
- <http://jquery.com/> : site officiel jQuery.
- http://www.digital-web.com/articles/jquery_crash_course/ : cours intensif en jQuery du magazine *Digital Web*.
- <http://simonwillison.net/2007/Aug/15/jquery/> : jQuery pour les programmeurs Java-Script par Simon Willison.
- <http://www.webdesignerwall.com/tutorials/jquery-tutorials-for-designers/> : didacticiel jQuery pour les concepteurs de Web Designer Wall.
- <http://www.noupe.com/jquery/50-amazing-jquery-examples-part1.html> : plus de cinquante didacticiels jQuery.
- <http://www.sastgroup.com/jquery/240-plugins-jquery> : deux cent quarante plug-ins pour jQuery.

En résumé

Vous avez maintenant abordé de nombreuses techniques utilisées pour programmer le site Roma Italia. Il y en a bien d'autres ; ouvrez le capot, plongez dans le code et vous dénicherez sans doute encore quelques perles.

Notez que les tailles de fichier pour ce site sont assez importantes, entre autres en ce qui concerne les scripts et les images. Si le site était un véritable site déployé, nous utiliserions

Ajax pour charger les images individuellement et nous réduirions les scripts autant que possible. Par exemple, jquery-1.3.2.js fait environ 120 KB, mais minimisé et zippé avec gzip, il ne fait plus que 19 Ko (cette version compressée est disponible en téléchargement sur le site jquery.com). Ces techniques d'optimisation réduiraient considérablement la taille globale de la page.

Tout l'intérêt des techniques illustrées dans cette étude de cas réside cependant certainement dans le balisage HTML brut sous-jacent qui est aussi robuste que la maquette est esthétique. Si tous les styles devaient être désactivés, les utilisateurs n'auraient aucune difficulté à lire le contenu du site et à le parcourir. Si ce n'est pas en soi un ravissement pour l'œil, le balisage signifiant et léger est une véritable confiserie pour les lecteurs d'écran, les robots des moteurs de recherche et autres outils de ce genre. C'est en quelque sorte le beurre et l'argent du beurre : une esthétique soignée et un code source élégant.

11

Étude de cas : Climb the Mountains

Par Simon Collison

Dans la précédente édition de *Maîtrise des CSS*, j'ai présenté mon étude de cas "More Than Doodles" en parlant de la "palette très riche" à l'aide de laquelle les concepteurs web pouvaient peindre. À cette époque, nous atteignions une phase critique où les standards du Web avaient opéré une avancée significative dans le secteur. C'était excitant et nous nous amusions beaucoup avec les CSS 2.1, en créant d'incroyables mises en page malgré les problèmes que nous posaient certains parmi les anciens navigateurs.

Plus de trois ans après, nous voilà déjà sérieusement engagés à implémenter des techniques de la spécification CSS 3. Nous pouvons remplacer des images d'arrière-plan décoratives par des combinaisons de règles CSS 3 comme `border-radius` et `box-shadow` et nous sommes capables de mieux contrôler la transparence des calques sans avoir recours à des images d'arrière-plan semi-transparentes grâce à la propriété `RGBa`. Plus important encore, malgré ces améliorations progressives, il est possible de proposer un affichage très soigné aux navigateurs qui ne prennent pas en charge les CSS 3 et leurs outils et techniques.

Dans cette étude de cas, vous en apprendrez plus sur :

- les conventions et l'organisation du HTML et des CSS ;
- la flexibilité de la grille ;
- la mise en valeur de la page actuelle en fonction de la classe de l'élément `body` ;
- le ciblage des éléments avec des pseudo-classes et des sélecteurs de frères adjacents ;
- la combinaison de classes pour plus d'efficacité et de flexibilité ;
- les propriétés `RGBa`, `border-radius` et `box-shadow` ;
- le positionnement des éléments de liste et l'affichage de leur contenu.

À propos de cette étude de cas

Cette étude de cas est bâtie sur une structure XHTML en béton – aussi épurée, organisée et puissante que possible. En particulier, le code XHTML ne contient aucune balise superflue qui aurait été ajoutée dans le simple but de pouvoir donner prise aux mécanismes de mise en forme CSS. Tout ce qui se trouve dans le code y est nécessaire et rien d'autre n'y figure. Le but de ce chapitre est donc de partir de ce qu'on a et d'utiliser des sélecteurs CSS véritablement astucieux pour cibler des éléments XHTML spécifiques sans devoir ajouter de div supplémentaires, de div de dégagement ou quoi que ce soit d'autre qui n'aurait pas sa propre raison d'être. Le site Climb the Mountains (grimper les montagnes), auquel nous ferons référence à l'aide de l'acronyme CTM, est une application web fictive dédiée aux randonneurs, passionnés d'escalade et autres fondus de la montagne qui ne pensent qu'à quitter le nid douillet de leur paisible maison pour passer des heures, des jours ou même des semaines

à errer dans une nature sauvage, hostile et escarpée. CTM est un site web social principalement orienté sur la communauté et les possibilités de mise en relation de ses membres (voir Figure 11.1). L'une de ses fonctionnalités-clés permet de charger sur le serveur les itinéraires GPS des membres ou de les exporter sur leurs périphériques GPS ; ces données ajoutent des statistiques détaillées concernant chaque itinéraire archivé. Par ailleurs, chaque itinéraire inclut des photographies, des cartes, des téléchargements et d'autres informations associées. L'architecture de l'information (AI) est empaquetée avec les données.

Figure 11.1

La page d'accueil du site Climb the Mountains.

Climb the mountains

Reach new heights by creating, sharing and recording your hiking routes with others.

FIND A ROUTE:
Keywords

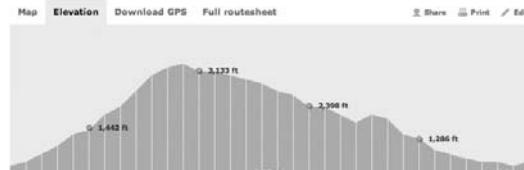
Home Routes About Shop

"Climb the mountains and get their good tidings. Nature's peace will flow into you as sunshine flows into trees. The winds will blow their own freshness into you, and the storms their energy, while cares will drop away from you like the leaves of Autumn."

JOHN MUIR, 1903



Your latest route: Scafell Pike via Corridor Route, 12th May 2009



Your other routes (view all)

- Kinderscout circuit
13.6 miles | Elevation 2,400ft
- Great Gable
8.9 miles | Elevation 3,900ft
- Snowdon Horseshoe
11.6 miles | Elevation 4,238ft
- Kinderscout circuit
13.6 miles | Elevation 2,400ft

[+ Create new route](#)

Recommended for you (view all)



Members' routes (view all)

- Kinderscout ridewalk circuit
13.6 miles | Elevation 2,400ft
from Glen Stanfield
- Castleton Ridges Walk
12.2 miles | elevation 1,343ft
from Phil Swan
- Branston and Eaton villages
8.7 miles | elevation 2,123ft
from Gregory Wood
- Ilkley Moor and Otley
34.7 miles | elevation 2,473ft
from Jamie Pittick

From the forums

- [Help! I am lost](#) Demis Willens 13/05/09
- [Best boots for under £100](#) Sapil Lineuph 29/04/09
- [Scrambling in Wales](#) Rory O'Deagóin 25/04/09
- [Do you take your dog walking?](#) Mike J Peletta 09/03/09
- [I'm too old for this!](#) Bell Campionma 29/02/09

Flickr



Nos super CSS vont permettre de donner vie à tout cela et de s'assurer que toutes ces données et images restent parfaitement immaculées avec une mise en page très flexible.

La maquette se décompose en blocs d'informations, afin que nous puissions facilement nous focaliser sur la zone dont nous traitons, plutôt que de bricoler avec des calques de mise en forme excessivement décoratifs. En analysant certaines de ces excellentes techniques, vous verrez sans doute comment elles peuvent être adaptées et appliquées à vos propres maquettes, avec d'autres fonctionnalités de ce livre.

Mes remerciements vont à mon collègue Greg Wood pour son aide considérable avec le concept du site. Toutes les photographies proviennent de mon compte Flickr. La plupart ont été prises dans la région de Lake District en Angleterre, au début de cette année. Les polices utilisées incluent quelques versions de la famille Palatino et des modèles plus courants dont Helvetica, Georgia et notre bonne vieille Verdana, avec des renvois par défaut à Arial ou Times New Roman.

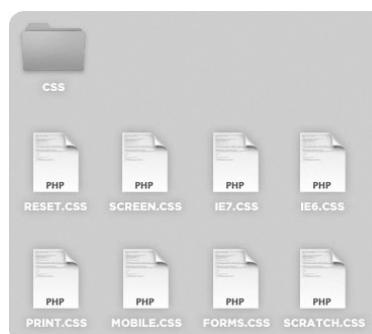
L'étude de cas est accessible en ligne à l'adresse <http://www.climbthemountains.com/cssm/>.

Organisation et conventions de la feuille de styles

Sans que la question se pose et sans envisager le moindre compromis, je considère que chaque site web dont je m'occupe doit être bâti sur de solides fondations. C'est un principe. Au cours des deux dernières années, j'ai travaillé avec mes collègues au développement d'une série de règles de base et de conventions qui doivent être un point de départ pour le HTML, les CSS, le JavaScript et ExpressionEngine. C'est une petite mallette de choc, remplie de fichiers CSS connectés, de conventions de noms, de modules, de plug-ins et de bibliothèques de scripts qui garantissent que chaque projet sur lequel travaillent les membres de mon équipe suivra ces conventions et sera facilement compréhensible par qui-conque viendra se joindre à l'action (voir Figure 11.2). En constante évolution, ce paquetage est l'un des plus précieux outils de notre arsenal.

Figure 11.2

Un ensemble classique de feuilles de styles de notre paquetage de fichiers de base.



Pour les CSS, nous disposons de feuilles de styles qui se combinent pour offrir une grande flexibilité dans le traitement des irrégularités des navigateurs et pour permettre aux membres

de l'équipe d'apporter leur contribution avec leurs propres fichiers de départ. Avec ces fichiers, les développeurs peuvent ajouter leurs propres règles CSS qui s'affichent dans le navigateur en raison de leur place dans la cascade. Si le directeur de projet est d'accord avec ces CSS, elles sont récupérées dans le fichier de départ et ajoutées à la feuille de styles principale appropriée, à la place de toutes les déclarations similaires existantes. Nous proposons aussi des feuilles de styles de base pour l'impression et les périphériques mobiles, ainsi qu'une feuille de styles séparée pour la mise en forme des formulaires. C'est notre façon de travailler.

La laborieuse feuille screen.css

La feuille de styles screen.css contient toutes les déclarations requises pour l'étude de cas CTM si vous travaillez dans un navigateur Mac ou dans Internet Explorer 8 ou Firefox sur PC, à côté de la feuille de styles reset.css. Si vous travaillez avec Internet Explorer 6 ou 7, vous aurez aussi besoin des feuilles de styles spécifiques d'Internet Explorer décrites plus loin.

Le fichier screen.css est lié de la manière suivante :

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

Un grand nombre des méthodes utilisées dans notre feuille de styles screen.css vous seront familières, mais examinons rapidement deux des outils que je considère essentiels.

Description du contenu

On a tôt fait d'ignorer cet outil ou de le négliger en considérant qu'il n'est pas nécessaire. Les CSS, pense-t-on, fonctionneront très bien sans ces notes descriptives. Repensez-y à deux fois. Que se passe-t-il si vous créez un site web au sein d'une équipe de plusieurs programmeurs ? Que se passe-t-il si vos feuilles de styles atteignent souvent des tailles considérables ? Comment s'assurer que d'autres personnes pourront facilement travailler avec vos maquettes et comment s'assurer que tout est bien organisé ?

C'est à ce titre que les notes de feuilles de styles – notamment, la présentation du contenu – peuvent être extrêmement précieuses. Rappelez-vous qu'il est possible d'ajouter des notes n'importe où dans les feuilles de styles en les intégrant avec la syntaxe suivante :

```
/* Voici une note simple */
```

Vous pouvez donc travailler ainsi pour fournir un tableau à jour du contenu de la feuille de styles. Cette référence permet aux autres concepteurs et développeurs de l'équipe de vérifier rapidement s'ils consultent la bonne feuille de styles et si les règles qu'ils cherchent s'y trouvent.

```
/*
CLIMB THE MOUNTAINS par SIMON COLLISON
VERSION 1.0
```

CONTENU -----

1. BODY
2. STYLES PAR DÉFAUT
3. TITRES
4. LIENS
5. IMAGES
6. MISE EN PAGE
7. LOGO/EN-TÊTE
8. NAVIGATION
9. INFO SITE/PIED DE PAGE
10. ÉLÉMENTS GLOBAUX
 - 10.1 IMAGE À LÉGENDE
 - 10.2 ÉLÉVATION
 - 10.3 DISTANCE/ÉLÉVATION PARAGRAPHE
11. PAGE D'ACCUEIL
 - 11.1 CONTENU PRINCIPAL
 - 11.2 CONTENU SECONDAIRE
 - 11.3 CONTENU TERTIAIRE

*/

La présentation exacte de ces notes dépend de la personne ou de l'équipe qui s'en charge. Dans l'exemple précédent, j'ai utilisé une structure composée de retours chariot et de tabulations pour créer une table des matières très lisible. L'important est de mettre à jour le contenu et d'y revenir chaque fois que vous ajoutez des règles à la feuille de styles, que vous en supprimez ou que vous en déplacez.

Réinitialisation

Le but, avec une feuille de styles reset.css, est de constituer une table nette pour tous les navigateurs et périphériques. Par exemple, certains navigateurs ont une feuille de styles par défaut qui définit des valeurs de marges, de remplissage, de tailles de police pour les titres, de hauteur de ligne, etc., différemment des autres navigateurs.

Nous insérons la feuille reset.css dans la cascade avec la ligne suivante dans screen.css :

```
@import url(reset.css); /* CSS DE REINITIALISATION */
```

On peut ainsi continuer en sachant que l'on a affaire à des éléments XHTML qui n'ont pas de marge, pas de remplissage, pas de hauteur de ligne, pas de taille de police, etc. Il est ainsi possible de travailler en confiance dans screen.css, en appliquant les valeurs souhaitées sans se soucier du risque d'hériter de valeurs provenant de la feuille de styles du navigateur.

Le maître en CSS Eric Meyer propose ce qu'il appelle un "point de départ, et non une boîte noire autonome à laquelle on ne touche pas" à l'adresse <http://meyerweb.com/eric/tools/css/reset/>. Cette feuille a été mon point de départ pour le site CTM, avec quelques ajustements et ajouts mineurs.

Feuilles de styles Internet Explorer utilisant des commentaires conditionnels

Cette méthode qui permet de cibler les versions de navigateur propres à Microsoft a été introduite au départ avec Internet Explorer 5 et ses versions à point. En utilisant le balisage XHTML enveloppé dans une instruction conditionnelle à l'intérieur d'un commentaire XHTML, vous pouvez utiliser cette combinaison syntaxique spéciale n'importe où dans un document XHTML et vous en servir pour transmettre des informations bien précises à des navigateurs ciblés.

```
<!--[if IE 6]> Tout ce qui vient ici n'est vu que par IE6 <![endif]-->
```

Dans l'exemple suivant, nous utilisons donc cette combinaison de syntaxe pour appeler trois feuilles de styles supplémentaires si l'utilisateur observe le site avec Internet Explorer 6, 7 ou 8.

```
<!--[if IE 6]><link href="assets/css/screen-ie6.css" type="text/css"  
  rel="stylesheet" media="screen" /><![endif]-->  
  
<!--[if IE 7]><link href="assets/css/screen-ie7.css" type="text/css"  
  rel="stylesheet" media="screen" /><![endif]-->  
  
<!--[if IE 8]><link href="assets/css/screen-ie8.css" type="text/css"  
  rel="stylesheet" media="screen" /><![endif]-->
```

L'intérêt de cette approche tient à ce que l'on peut éviter d'ajouter des hacks caractéristiques d'Internet Explorer aux règles CSS existantes dans screen.css (qui seront appliquées par tous les autres navigateurs) et créer à la place des règles propres au navigateur dans les feuilles de styles Internet Explorer. C'est ce que nous faisons pour la maquette du site CTM, car nous appelons certains ajouts spécifiques d'Internet Explorer dans les CSS, comme nous allons le voir un peu plus loin.

Flexibilité de la grille

Une grille agit à la manière d'une fondation solide pour n'importe quelle page dans n'importe quel site web. Le fait d'y recourir doit vous libérer et non vous restreindre. N'ayez jamais peur de sortir des lignes et de risquer quelques essais. La grille est un rappel, un guide – en quelque sorte, une présence qui rassure.

En général, de largeur prédéterminée et dotée d'un nombre de colonnes et de gouttières optionnelles, la grille est un appui, un guide tranquillisant. Elle a une fonction intermédiaire entre Photoshop et les CSS. Elle livre les informations concernant vos choix de mise en page initiaux pour ce qui concerne les éléments flottants, le positionnement, les marges, le remplissage, les bordures, etc.

Comme bien d'autres concepteurs, quand je réalise mes prototypes dans Photoshop, sous Fireworks ou dans le navigateur lui-même, je travaille avec un calque de grille que j'active et désactive à volonté (voir Figure 11.3).

Figure 11.3

Le site Climb the Mountains avec sa grille de colonne sous-jacente.



Fonctionnement de la mise en page du site CTM

Le site CTM utilise une mise en page construite à partir d'une grille robuste mais flexible de 1 000 pixels de large. Dans ce canevas figurent douze colonnes, chacune pourvue d'une gouttière bien marquée. Chaque colonne fait 65 pixels de large et chaque gouttière en fait 25 (voir Figure 11.4).

Figure 11.4

Disposition des colonnes.



Chaque colonne possède également sa propre structure : elle est scindée en trois sous-colonnes ou largeurs (de gauche à droite) de 25 pixels, 15 pixels et 25 pixels (voir Figure 11.5).

Figure 11.5

Largeurs des sous-colonnes internes.



Ces sous-colonnes permettent en quelque sorte de travailler avec une grille à l'intérieur de la grille. Elles offrent des points de référence supplémentaires, des unités de mesure ou des repères qui peuvent être utilisés quand les deux colonnes ne sont pas assez précises pour le positionnement de certains éléments. Observez l'exemple de la Figure 11.4 et voyez comment les éléments sont alignés sur les principales colonnes et parfois sur les sous-colonnes.

Contrôle de la navigation avec les classes de l'élément **body**

La valeur attribuée à l'élément **body** peut être utilisée pour changer la mise en page, le comportement du contrôle et d'autres choses importantes en CSS. Dans la première édition de ce livre, j'ai utilisé une technique avec des ID pour l'élément **body** de chaque page afin de contrôler la mise en page, en combinant cela avec une classe pour l'emplacement, comme `<body id="threeColLayout" class="home">`. Cette fois, je n'utilise que la classe.

```
<body class="home">
```

Peu importe que vous utilisiez un ID ou une classe à cette occasion ; la technique est tout aussi efficace dans les deux cas.

Mise en valeur de la page courante

Il existe plusieurs moyens de mettre en valeur la page sur laquelle vous vous trouvez. De nombreux concepteurs utilisent d'astucieux mécanismes PHP pour déclencher les CSS, par exemple en passant en surbrillance le lien Accueil de la barre de navigation principale quand on se trouve sur la page d'accueil. C'est très intéressant, mais il est tout aussi facile de procéder en appliquant intelligemment des CSS, en couplant la classe **body** et une classe de navigation. Voyons comment cela fonctionne.

```
<body class="home">
```

La page est identifiée comme la page d'accueil et nous allons maintenant nous assurer que chaque élément de navigation possède une classe appropriée :

```
<ul id="navigation_pri">
  <li class="nav_home"><a href="#">Home</a></li>
  <li class="nav_routes"><a href="#">Routes</a></li>
  <li class="nav_about"><a href="#">About</a></li>
  <li class="nav_shop"><a href="#">Shop</a></li>
</ul>
```

Dans ce fragment de code, vous remarquerez que le lien Home possède la classe `nav_home`. Nous avons également ajouté une classe à l'élément **body** de la page Routes, afin de pouvoir tester ce comportement par la suite :

```
<body class="routes">
```

Ensuite, nous utilisons les CSS pour appliquer la mise en forme à la liste de navigation. Vous remarquerez que l'élément est positionné de manière absolue, à un endroit précis défini par rapport au haut et à la gauche du conteneur principal.

```
ul#navigation_pri {
    list-style:none;
    margin:0;
    position:absolute;
    top:0;
    left:415px;
    font-size:19px;
    font-weight:bold;
    font-family:Helvetica,Arial,sans-serif;
}
ul#navigation_pri li {
    float:left;
    margin:0;
    padding:30px 10px 0 10px;
    height:3000px;
}
ul#navigation_pri li a {
    color:#000;
}
ul#navigation_pri li a:hover,
ul#navigation_pri li a:focus {
    color:#333;
    text-decoration:underline;
}
```

Notre système de navigation de base est maintenant mis en forme (voir Figure 11.6), mais sans indication concernant la page actuellement affichée (nous reviendrons sur l'élément `blockquote` qui se trouve sur le calque de navigation plus loin dans cette section).

Figure 11.6

Navigation principale de base.

Home Routes About Shop

Logged in as collylogic
ACCOUNT | LOG OUT

"Climb the mountains and get their good tidings. Nature's peace will flow into you as sunshine flows into trees. The winds will blow their own freshness into you, and the storms their energy, while cares will drop away from you like the leaves of Autumn."

JOHN MUIR, 1903



L'étape suivante consiste à utiliser un sélecteur pour définir la relation entre la classe de l'élément `body` et la classe `home` de navigation. Vous remarquerez que nous avons groupé deux règles identiques pour la page d'accueil et la page `Routes` :

```
.home ul#navigation_pri li.nav_home,
.routes ul#navigation_pri li.nav_routes {
    background-color:#f5f5f5;
}
```

La première partie du sélecteur `.home` ou `.routes` vérifie que nous sommes sur cette page. La mise en forme n'est appliquée que si l'élément `ul#navigation_pri` est un enfant de `.home` ou de `.routes`. Si une correspondance est trouvée, l'action est réalisée. Ce style crée l'arrière-plan gris clair qui remplit toute la zone de l'onglet dans la barre de navigation.

Ensuite, on peut définir des styles pour le comportement du lien, à nouveau en regroupant les règles identiques pour `.home` et `.routes` :

```
.home ul#navigation_pri li.nav_home a,
.routes ul#navigation_pri li.nav_routes a {
    color:#278dab;
    background:#f5f5f5 0 center no-repeat;
    padding:0 0 0 20px;
}
.home ul#navigation_pri li.nav_home a:hover,
.home ul#navigation_pri li.nav_home a:focus,
.routes ul#navigation_pri li.nav_routes a:hover,
.routes ul#navigation_pri li.nav_routes a:focus {
    text-decoration:none;
    color:#000;
}
```

Nous obtenons ainsi les liens texte bleus dont nous avons besoin. Pour finir, on peut ajouter une décoration au lien de la page d'accueil uniquement. Lorsque la page d'accueil est affichée, nous affichons une petite icône de maison à gauche du lien Home :

```
.home ul#navigation_pri li.nav_home a {
    background-image:url(../images/site/nav_back.gif);
}
```

La Figure 11.7 montre le résultat quand on affiche la page d'accueil.

Figure 11.7

Le lien Home sélectionné.



La Figure 11.8 présente la page Routes.

Figure 11.8

Le lien Routes sélectionné.



Mise en forme de la citation

Revenons maintenant à cette citation de John Muir qui se trouve sur le calque de navigation principal. J'aime cette citation ; elle m'incite à sortir par tous les temps. N'oublions jamais qu'il faut aller dehors et voir le monde tel qu'il est ! Sur le site CTM, il serait possible de proposer des citations bien choisies et de les faire tourner dans cette partie de la page, pour stimuler les visiteurs. Voici le code :

```
<blockquote id="johnmuir">
  <p>&ldquo;Climb the mountains and get their good tidings. Nature's peace will
  ↪flow into you as sunshine flows into trees. The winds will blow their own
  ↪freshness into you, and the storms their energy, while cares will drop away
  ↪from you like the leaves of Autumn.&rdquo;</p>
  <p><cite>John Muir, 1903</cite></p>
</blockquote>
```

Hormis l'emploi des entités de caractères spécifiques pour les guillemets, il n'y a là rien de bien inhabituel. Passons donc à la mise en forme. Auparavant, nous avons positionné l'élément `ul#navigation_pri` de manière absolue avec `position: absolute`, à 0 pixel du haut et 415 pixels de la gauche.

```
ul#navigation_pri {
  list-style:none;
  margin:0;
  position:absolute;
  top:0;
  left:415px;
  font-size:19px;
  font-weight:bold;
  font-family:Helvetica,Arial,sans-serif;
}
```

On peut maintenant définir la mise en forme pour le bloc de citation :

```
div#branding blockquote {
  width:505px;
  float:right;
  padding:0 70px 20px 0;
  background:url(..../images/site/branding_johnmuir.jpg) no-repeat right top;
}
```

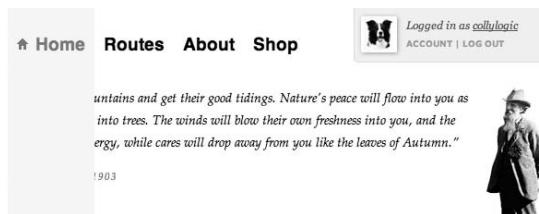
Le bloc figure cependant sous la zone d'onglet de navigation (voir Figure 11.9).

Comme le système de navigation est positionné de manière absolue, on peut ajouter `position: relative` à l'élément `blockquote` et le faire remonter au-dessus de la zone de l'onglet :

```
div#branding blockquote {
  position:relative;
  width:505px;
  float:right;
  padding:0 70px 20px 0;
  background:url(..../images/site/branding_johnmuir.jpg) no-repeat right top;
}
```

Figure 11.9

Le bloc de citation est partiellement masqué par le lien Home.



À présent, l'élément `blockquote` se trouve au-dessus de la barre de navigation principale, comme le montraient les précédents exemples (voir Figure 11.7).

Ciblage stratégique des éléments

Nous avons vu précédemment combien il était efficace d'utiliser des sélecteurs descendants pour contrôler la mise en forme de la navigation principale en fonction de la classe de l'élément `body` de chaque page. Ici, nous allons voir encore comment des choix réfléchis au niveau du code HTML peuvent offrir une grande flexibilité de contrôle avec des sélecteurs descendants et comment ce système forme la base d'un contrôle plus avancé grâce au ciblage des éléments.

Sélecteurs descendants profonds

Pour commencer, disposons les fondations. Du côté droit du site figure le panneau jaune clair Members' Routes (itinéraires des membres), qui présente les randonnées transmises par d'autres utilisateurs du site. Il n'y a rien de particulièrement remarquable dans le code, mais examinons-le pour mieux comprendre les prouesses que nous allons réaliser à l'aide de sélecteurs plus loin.

Vous remarquerez que plusieurs éléments, comme `h3`, `p` et `img`, sont regroupés dans chaque élément de liste à puces. J'ai souvent été surpris de voir que des développeurs ne savaient pas qu'il était possible d'ajouter toutes sortes d'éléments intéressants dans l'élément `li`. Le plus souvent, vous ne découvrirez que des éléments `a` et parfois `img`, mais en réalité, on peut faire bien plus.

```
<div id="others_routes">
  <h2>Members' routes <a href="#" class="more">(view all)</a></h2>
  <ul>
    <li>
      <h3><a href="#">Kinderscout circuit</a></h3>
      <p class="dist_elev">13.6 miles | Elevation 2,400ft</p>
      <p class="username"><a href="#">from Glen Swinfield </a></p>
    </li>
    <li>
      <h3><a href="#">Castleton Ridge Walk</a></h3>
      <p class="dist_elev">12.2 miles | elevation 1,343ft</p>
      <p class="username"><a href="#">from Phil Swan </a></p>
```

```

</li>
<li>
  <h3><a href="#">Branston Circular</a></h3>
  <p class="dist_elev">5.7 miles | elevation 1,213ft</p>
  <p class="username"><a href="#">from Gregory Wood </a></p>
</li>
<li>
  <h3><a href="#">Ilkley Moor and Otley</a></h3>
  <p class="dist_elev">24.7 miles | elevation 2,473ft</p>
  <p class="username"><a href="#">from Jamie Pittock </a></p>
</li>
</ul>
</div>

```

En construisant le contenu de cette manière, on peut collationner les blocs d'information sous forme de listes en profitant de tous les mécanismes de hiérarchisation et de mise en forme qui font l'intérêt des listes.

Il est ensuite très facile d'utiliser des sélecteurs simples pour cibler la liste à puces dans la div conteneur `others_routes` et les différents éléments dans les éléments `li`. Notez que nous utilisons des règles `border-radius`, `-webkit-border-radius` et `-moz-border-radius` pour appliquer des bords arrondis à l'élément `ul`. Nous y reviendrons un peu plus loin.

```

div#others_routes ul {
  list-style:none;
  border:1px solid #dedeaf;
  background:#ffffcc;
  border-radius:5px;
  -webkit-border-radius:5px;
  -moz-border-radius:5px;
  margin:0;
  padding:10px;
}
div#others_routes ul li {
  margin:0;
  padding:10px 55px 10px 0;
  position:relative;
  border-bottom:1px solid #dedeaf;
  border-top:1px solid #fff;
}

div#others_routes ul li h3 {
  margin-bottom:5px;
}
div#others_routes ul li img {
  position:absolute;
  top:10px;
  right:10px;
}

```

```

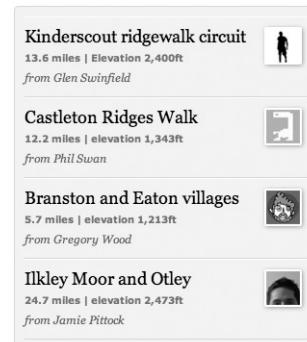
div#others_routes ul li p.username {
  margin:3px 0 0 0;
  font-style:italic;
  font-size:12px;
}
div#others_routes ul li p.username a {
  color:#666;
}
div#others_routes ul li p.username a:hover,
div#others_routes ul li p.username a:focus {
  text-decoration:underline;
}

```

Dans ce code, nous ciblons les éléments HTML plus profonds avec des sélecteurs descendants simples. Par exemple, on peut cibler de manière stratégique la mise en forme de survol du lien de nom d'utilisateur avec `div#others_routes ul li p.username a:hover`, en descendant de plus en plus profondément avec le sélecteur, jusqu'à atteindre l'élément cible – un élément que possède chaque élément antérieur du sélecteur. Le résultat est présenté à la Figure 11.10.

Figure 11.10

Le conteneur initial
Members' Routes.



Parfait. La liste de randonnées et d'itinéraires transmis par les membres prend agréablement forme et nombreux sont ceux qui s'en tiendraient là. Tout a l'air bien soigné. Ah, mais attendez ! Nous sommes perfectionnistes et nous disposons de puissants styles CSS. Pourquoi s'en tenir au bon quand on peut atteindre l'excellent ?

Dans les deux exemples suivants, nous allons peaufiner le haut et le bas du conteneur des itinéraires grâce à d'intéressantes astuces CSS.

La pseudo-classe `:first-child`

S'il vous est déjà arrivé de vous demander comment cibler la première lettre ou la première ligne d'un bloc de texte, sachez qu'il suffit d'utiliser une pseudo-classe comme `:first-letter` ou `:first-line`. Ces astuces passionnantes permettent de mettre en forme des éléments en fonction d'une logique simple.

La pseudo-classe `:first-child` ne cible qu'un élément qui est lui-même le premier enfant d'un élément conteneur.

Dans le conteneur des itinéraires des membres, les informations de détail de chaque élément sont ajoutées sur une liste à puces. Chacun des éléments li possède le même remplissage et une fine bordure supérieure et inférieure.

```
div#others_routes ul li {
  margin:0;
  padding:10px 55px 10px 0;
  position:relative;
  border-bottom:1px solid #dedeaf;
  border-top:1px solid #fff;
}
```

Le contenu de la liste est ainsi espacé de manière régulière, mais j'aimerais réduire la quantité de remplissage uniquement pour l'élément du haut de la liste (dans cet exemple, il s'agit du circuit de randonnée Kinderscout). Je ne veux d'ailleurs pas du tout de remplissage en haut, et pas de bordure non plus.

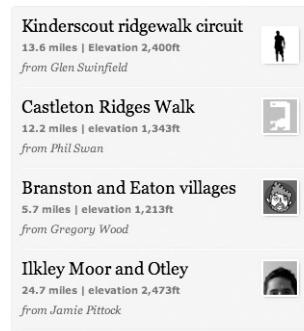
Pseudo-classe, à la rescouisse ! Ici, je crée une nouvelle règle et j'utilise le même sélecteur pour cibler les éléments de la liste à puces à l'intérieur de la div conteneur `#others_routes`, mais en ajoutant `:first-child` juste après l'élément `li`, afin de transmettre en somme ce message au navigateur : "Entrez dans le conteneur, trouvez la liste à puces et appliquez la mise en forme suivante en ne redéfinissant que le tout premier élément trouvé."

```
div#others_routes ul li:first-child {
  padding-top:0;
  border-top:none;
}
```

Comme le montre la Figure 11.11, l'élément du circuit de Kinderscout n'a plus de bordure ni de remplissage supérieurs et se cale maintenant bien serré sur la partie supérieure du conteneur parent.

Figure 11.11

Le remplissage et la bordure du haut ont été supprimés.



Cette technique est un jeu d'enfant, mais elle constitue une méthode très efficace pour cibler un élément particulier, à mille et une fins possibles. Maintenant que nous avons traité le haut de la liste, voyons ce qu'on peut faire avec le bas.

Sélecteurs de frères adjacents

Après avoir vu `:first-child`, le moment est tout trouvé pour vous de découvrir `:last-child`, qui permet de cibler la dernière occurrence d'un élément enfant dans un conteneur parent spécifique. Le principe est le même que pour `:first-child`, aussi n'hésitez pas à faire des essais. Malheureusement, seules les dernières versions des navigateurs comme Safari, Firefox, Google Chrome et Opera prennent en charge cette méthode. Il convient donc de tenir compte d'Internet Explorer 6, 7 et 8 en employant une méthode alternative, grâce aux sélecteurs de frères adjacents.

Dans cet exemple, nous devons maintenant réaliser l'inverse de ce que nous venons de faire avec la pseudo-classe `:first-child`. Comme indiqué précédemment, chaque élément de liste à puces possède un remplissage supérieur et inférieur ainsi qu'une bordure supérieure et inférieure. Nous avons pu annuler ces styles pour le premier élément `li`, nous devons maintenant le faire pour le dernier élément.

Comment procéder ? Comment la feuille de styles va-t-elle savoir quel est le dernier élément d'un certain groupe et comment le cibler précisément ?

Les sélecteurs de frères adjacents sont constitués de plusieurs sélecteurs séparés par le combinateur `+`. Ce symbole représente un élément qui est le prochain frère du premier élément. Notez que les éléments doivent avoir le même parent et que le premier doit immédiatement précéder le second.

Comme pour `:first-child`, nous ciblons encore la div parente `others_routes` et nous descendons méthodiquement vers les profondeurs à l'aide des sélecteurs, jusqu'à atteindre l'élément que nous souhaitons mettre en forme. La liste à puces aura toujours quatre éléments `li`. C'est ici la clé de notre méthode :

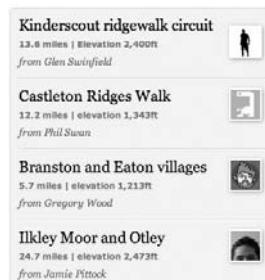
```
div#others_routes ul li + li + li + li {
    padding-bottom:0;
    border-bottom:none;
}
```

Nous avons créé un sélecteur dont la signification pourrait se traduire ainsi : "Ah, lorsque vous êtes dans la div `others_routes`, trouvez la liste à puces, comptez jusqu'au quatrième élément `li` et appliquez les styles à cet élément uniquement." Simple.

Le résultat (voir Figure 11.12) présente le quatrième élément `li` sans remplissage inférieur ou bordure inférieure. Le résultat est plus soigné. Il témoigne d'une grande attention portée aux détails, grâce simplement aux sélecteurs CSS à disposition.

Figure 11.12

La bordure et le remplissage inférieurs ont été retirés avec succès.



Transparence, ombres et bords arrondis

L'étude de cas que j'ai proposée dans la première édition de ce livre s'appuyait fortement sur des boîtes à bords arrondis. Tout le monde cherche, à un moment ou un autre, à créer des bords arrondis par souci esthétique et pour changer des ennuyeux coins carrés.

Parmi les nombreuses méthodes permettant de créer des bords arrondis, j'en avais choisi une qui utilisait une quantité non négligeable de code JavaScript, plusieurs sprites d'image d'arrière-plan et quelques pincées de code HTML superflu. C'était lourd, maladroit, peu commode et il n'y avait pas vraiment d'alternative.

Aujourd'hui, tout a changé et je n'ai qu'une envie, c'est de crier "CSS 3 !" aussi fort que je le peux. On a fait du chemin : les attentes se sont amplifiées et les outils ont évolué. Évidemment, les navigateurs n'ont pas tous suivi (quoi de neuf sous le soleil ?), mais nous avons gagné en bravoure et sommes prêts à travailler avec de nouvelles idées et à faire avancer les choses.

À cette section, je prendrai une simple image et une légende de la page d'accueil du site CTM pour réaliser toutes sortes de merveilles en CSS 3, sans utiliser ni JavaScript, ni autres images ou balise superflu. Vive la révolution !

Notre objectif

Nous allons commencer par une image JPEG de 310×185 pixels nommée campsite.jpg (voir Figure 11.13). Ensuite, nous appliquerons une légende avec du texte blanc sur un calque semi-transparent gris en bas de l'image, puis nous appliquerons une bordure photo de style Polaroid autour de l'image, en veillant à y ajouter des bords arrondis et une ombre portée réaliste. Et tout cela, grâce aux CSS.

Figure 11.13

L'image campsite.jpg initiale.



Le code HTML est très simple. L'image et la légende doivent être contenues dans une div nommée `captioned_image` pour les besoins de cet exemple. Le paragraphe se voit attribuer la classe `caption`, afin qu'on puisse le cibler. Pour l'instant, voilà tout.

```
<div class="captioned_image">
  
  <p class="caption">From the campsite bridge towards the village, Great Gable
  and Lingmell.</p>
</div>
```

Une fois le code HTML en place, il est temps d'expérimenter trois des techniques CSS 3 les plus passionnantes qui existent.

Bandeau translucide de légende et transparence RGBa

Il existe plusieurs moyens de spécifier des couleurs. Le plus souvent, cela se fait à l'aide d'un triplet RVB au format hexadécimal. D'autres développeurs utilisent les noms anglais des couleurs, dans certains cas. Il est également possible d'utiliser des pourcentages RVB ou des chiffres décimaux. Tous les exemples suivants sont ainsi valides pour spécifier la couleur rouge :

```
color: #f00
color: #ff0000
color: red
color: rgb(255,0,0)
color: rgb(100%, 0%, 0%)
```

RGB est l'acronyme de *Red, Green, Blue* (rouge, vert, bleu). C'est un mode colorimétrique bien connu des graphistes. RGBa introduit une quatrième couche, dite alpha, qui gère la transparence. La beauté des CSS 3 tient à ce qu'il est possible de continuer à spécifier les couleurs en RVB, mais également de définir une transparence alpha de ces couleurs à l'aide d'une quatrième valeur décimale. Toutes les valeurs comprises entre 0.0 (transparence totale) et 1.0 (opacité totale) sont autorisées.

Dans l'exemple suivant, nous déclarons à nouveau la couleur rouge en RVB, mais en fixant une transparence à 50 % avec la valeur 0.5 pour la transparence alpha.

```
color: rgb(255,0,0,0.5)
```

La valeur RGBa n'est attribuée qu'à l'élément déclaré : les éléments enfants n'héritent donc pas de la transparence. Elle se distingue ainsi de la propriété `opacity`, qui est toujours héritée.

Pour le site CTM, les déclarations suivantes exécutent les premières formules magiques pour la photo et la légende. Nous positionnons la div conteneur de manière relative, puis la légende de manière absolue, afin qu'elle puisse être placée exactement où nous le voulons au-dessus de l'image.

```
div.captioned_image {
    position: relative;
}
div.captioned_image p.caption {
    position: absolute;
    bottom: 0;
    left: 0;
    margin: 0;
    color: #fff;
    font-size: 13px;
    line-height: 16px;
    font-style: italic;
    padding: 5px;
}
```

Ensuite, nous déclarons la valeur RGBa avec la formule `rgba(0,0,0,0.5)`, les trois premières valeurs combinées produisant du noir et la valeur de transparence alpha de 0.5

produisant une semi-transparence, qui peut être ajustée jusqu'à ce que l'effet global soit pleinement satisfaisant.

```
div.captioned_image p.caption {
  position: absolute;
  bottom: 0;
  left: 0;
  margin: 0;
  background: rgba(0, 0, 0, 0.5);
  color: #fff;
  font-size: 13px;
  line-height: 16px;
  font-style: italic;
  padding: 5px;
}
```

On obtient ainsi le bandeau semi-transparent de légende souhaité (voir Figure 11.14).

Figure 11.14

La légende
transparente est mise
en place.



Comme c'est le cas avec de nouvelles possibilité CSS 3, certains navigateurs restent à la traîne, et notamment Internet Explorer (dont la version 8 actuelle), qui ne reproduit pas la transparence alpha. Par exemple, Internet Explorer 7 affiche par défaut un calque gris uni à peu près acceptable, comme on en obtient en proposant une image PNG transparente sans forcer la prise en charge de la transparence alpha (voir Figure 11.15).

Figure 11.15

La légende telle que
l'affiche Internet
Explorer 7.



Internet Explorer 8, qui ne prend toujours pas en charge le mode RGBa, affiche simplement le texte de la légende au-dessus de l'image, sans arrière-plan. Pour contourner ce problème, on peut ajouter une règle à la feuille de styles screeni-ie8.css pour s'assurer qu'un arrière-plan gris est placé derrière le texte.

```
div.captioned_image p.caption { background: #666; }
```

La leçon à retenir est de ne pas se décourager à cause d'Internet Explorer et de ses défaillances. Je vous encourage tous à expérimenter l'incroyable flexibilité de RGBa dès maintenant, sur une variété d'éléments dans vos maquettes. Vous n'en démordrez plus et toutes vos mises en page y gagneront !

Combinaison de classes

Je suis toujours ébahi quand je parle à des concepteurs qui ne savent pas que les classes peuvent être combinées pour un traitement plus flexible des éléments.

Vous pouvez ainsi utiliser `class="profile"` plusieurs fois dans une page donnée, en attribuant des informations de couleur et de disposition courantes. Mais imaginons que vous souhaitiez ne modifier que la couleur d'arrière-plan en fonction d'un unique critère, comme le fait que l'utilisateur soit un membre ou un simple invité. Au lieu de créer deux styles de profil pour proposer ces références de couleur distinctes, vous pouvez conserver simplement les couleurs dans des règles séparées et les combiner avec la classe `profile`.

```
.profile {  
    width:300px;  
    margin:0 10px;  
    padding:10px;  
    font-size:11px;  
}  
.guest {  
    background-color:#ff9900;  
}  
.member {  
    background-color:#ff0000;  
}
```

Ce style peut ensuite être appliqué avec différentes combinaisons de classes selon le statut de l'utilisateur. Vous pouvez combiner autant de classes que vous le souhaitez, en les séparant simplement par un espace, comme ceci :

```
<div class="profile member">  
    <p>Member options...</p>  
</div>
```

Dans le site CTM, nous avons combiné des classes pour ajouter un cadre autour de certaines images illustrées. Vous remarquerez qu'à côté de la classe `captioned_image` nous avons ajouté la classe `polaroid` :

```
<div class="captioned_image polaroid">  
      
    <p class="caption">From the campsite bridge towards the village, Great Gable and Lingmell.</p>  
</div>
```

Il est maintenant possible de définir la mise en forme pour ce cadre de Polaroïd. Celle-ci va faire appel quelques autres astuces de la spécification CSS 3.

border-radius

Dans la précédente édition de ce livre, Andy et moi-même avions présenté une technique utile mais assez laborieuse pour ajouter des cadres et des ombres aux images. Elle nécessitait deux div et des images d'arrière-plan qu'il fallait mettre en forme et positionner très soigneusement. Bon, d'accord, c'était il y a bien longtemps, en 2005.

Aujourd'hui, la propriété *border-radius* permet d'ajouter des bords arrondis aux éléments grâce à de simples déclarations CSS. Malheureusement, Internet Explorer (tiens, ce nom me dit quelque chose...) ne prend pas du tout en charge cette propriété et affiche des coins carrés (ce qui reste acceptable, selon moi). Il en va pour le moment de même avec le navigateur Opera.

À ce jour, aucun navigateur populaire ne prévoit de prendre en charge la propriété standard *border-radius*. S'il est important d'inclure la déclaration pour voir à long terme et aller de l'avant, il faut cependant aussi ajouter deux déclarations supplémentaires à court terme : l'une pour les navigateurs Mozilla, comme notre adoré Firefox, et l'autre pour les navigateurs WebKit, comme notre cher Safari (qui prend d'ailleurs aussi en charge les bords elliptiques). Pour plus d'informations, des exemples et quelques astuces utiles, consultez les descriptions et les exemples à l'adresse <http://www.the-art-of-web.com/css/border-radius/>.

Les trois déclarations sont clairement définies ici :

```
.polaroid {  
    border:5px solid #f00;  
    border-radius:5px;  
    -webkit-border-radius:5px;  
    -moz-border-radius:5px;  
}
```

Considérant que le blanc sur blanc n'était pas idéal pour une démonstration, vous remarquerez que j'ai spécifié une bordure rouge temporaire afin que vous puissiez voir ce qui se passe à la Figure 11.16. Vous verrez que chaque coin est arrondi, suivant un rayon de courbure d'environ 5 pixels. Il s'agit en fait du rayon d'un quart d'ellipse définissant le coin. Comme avec les marges, le remplissage et les bordures, il existe quatre propriétés *border-radius* individuelles (une pour chaque coin de l'élément) et une propriété raccourcie.

Figure 11.16

La bordure rouge temporaire fait clairement apparaître les bords arrondis.



Comme vous pouvez le voir, cette méthode est bien plus simple que celle de la première édition du livre ! Maintenant que les bords sont arrondis, on peut ajouter une ombre portée simple pour donner à l'image un peu plus d'attrait.

box-shadow

Les CSS 3 proposent une méthode sensiblement plus simple de créer des ombres portées pour Safari 3 et Firefox 3.5 ainsi que leurs versions ultérieures. La propriété prend trois attributs de longueur (le décalage horizontal, le décalage vertical et le rayon de flou) et, pour finir, une couleur.

Si une valeur positive est appliquée au décalage horizontal, l'ombre apparaît du côté droit de l'élément. Un décalage négatif place l'ombre du côté gauche.

Si une valeur négative est appliquée au décalage vertical, l'ombre apparaît en haut de l'élément. Avec une valeur positive, elle apparaît en bas.

Le rayon de flou est un réglage très utile. Si sa valeur est fixée à 0, l'ombre est nette. Plus sa valeur est élevée, plus l'ombre devient floue.

Dans la classe `polaroid`, les quatre valeurs peuvent être utilisées pour créer une ombre portée en bas à droite de notre image illustrée, avec un flou de 5 pixels en gris moyen :

```
.polaroid {
    border:5px solid #fff;
    border-radius:5px;
    -webkit-border-radius:5px;
    -moz-border-radius:5px;
    -webkit-box-shadow:1px 1px 5px #999;
    -moz-box-shadow:1px 1px 5px #999;
}
```

Intelligemment, `box-shadow` respecte la valeur donnée précédemment à `border-radius`. On obtient ainsi un cadre d'image arrondi et son ombre portée complémentaire en parfaite harmonie (voir Figure 11.17).

Figure 11.17

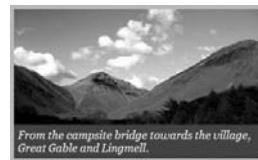
L'image possède maintenant une légende, un cadre et des bords arrondis.



Malheureusement, ce n'est pas encore le cas pour Internet Explorer (voir Figure 11.18).

Figure 11.18

Internet Explorer n'affiche pas les propriétés CSS 3, mais le résultat est acceptable.



Comme nous l'avons indiqué précédemment, Internet Explorer ne prend pas en charge la transparence RGBa pour la légende et vous pouvez voir aussi que notre cadre est affiché dans un rectangle gris à angles droits. Il possède la même largeur (5 pixels) mais n'est ni

blanc ni arrondi. Vous remarquerez aussi que nous sommes privés de notre belle ombre portée. Rappelez-vous que nous nous sommes assurés que le texte de la légende repose sur un arrière-plan gris en ajoutant une règle à notre feuille de styles screen-i8.css. J'espère que les choses auront évolué pour Internet Explorer avant que je ne devienne grand-père.

Positionner des listes et révéler leur contenu

Dans cette partie de notre étude de cas, nous allons nous concentrer sur la zone Your latest route (votre dernier itinéraire) du site, du côté gauche de la page. Dans une version totalement aboutie du site, il s'agirait d'une sélection de statistiques, de cartes et de graphiques liés à une randonnée particulière, chaque panneau pouvant être observé en cliquant sur un onglet différent.

Pour commencer, nous allons ajouter la navigation pour la section des statistiques de la page d'accueil du site CTM. Le code HTML requiert deux listes, l'une pour les onglets de statistique à gauche et l'autre pour les options de partage, d'impression et d'e-mail à droite :

```
<ul id="route_nav">
  <li><a href="#">Map</a></li>
  <li class="cur"><a href="#">Elevation</a></li>
  <li><a href="#">Download GPS</a></li>
  <li><a href="#">Full routesheet</a></li>
</ul>

<ul id="route_action">
  <li class="share"><a href="#">Share</a></li>
  <li class="print"><a href="#">Print</a></li>
  <li class="edit"><a href="#">Edit</a></li>
</ul>
```

Notez que nous avons ajouté `class="cur"` à l'onglet Elevation, car nous souhaitons qu'il apparaisse comme sélectionné dans cet exercice, et nous ciblons directement ce lien avec cette classe supplémentaire.

Les deux listes à puces héritent de quelques styles existants provenant d'autres endroits du document ; on obtient ainsi les liens bleus et les règles `font-family` et `font-size` présentées à la Figure 11.19.

Figure 11.19

Les listes de navigation de base de la zone Your latest route.

- Map
- Elevation
- Download GPS
- Full routesheet
- Share
- Print
- Edit

Sinon, j'ai fait flotter la liste Share, Print et Email à droite. Rien de spécial à ce sujet. Donc, à partir de maintenant, concentrons-nous sur la liste à gauche. Nous allons la faire flotter à

gauche et ajouter quelques déclarations de style de base, dont une couleur plus vive pour l'onglet sélectionné :

```
ul#route_nav {
    list-style:none;
    font-family:Verdana,sans-serif;
    font-size:11px;
    font-weight:bold;
    float:left;
    margin:0;
}
ul#route_nav li {
    float:left;
    margin:0;
    padding:7px 10px;
}
ul#route_nav li a {
    color:#666;
}
ul#route_nav li a:hover,
ul#route_nav li a:focus {
    color:#333;
}
ul#route_nav li.cur a {
    color:#000;
}
```

On obtient ainsi les deux listes présentées à la Figure 11.20, l'une flottante à gauche et l'autre flottante à droite, avec une mise en forme initiale.

Figure 11.20

Map Elevation Download GPS Full routesheet  Share  Print  Edit

Les deux listes flottent respectivement à gauche et à droite.

Nous allons maintenant dégager ces éléments flottants en utilisant `clear:both` sur le conteneur des statistiques qui suivra cette barre de navigation. Ensuite, nous ajouterons une couleur d'arrière-plan à la classe sélectionnée :

```
ul#route_nav li.cur {
    background:#dff1f1;
}
```

Ce code définit la zone exacte de l'onglet sélectionné (voir Figure 11.21).

Figure 11.21

Map **Elevation** Download GPS Full routesheet  Share  Print  Edit

Les deux listes flottent respectivement à gauche et à droite lorsque l'onglet sélectionné est mis en surbrillance.

Notre système de navigation pour la section Your latest route est maintenant fonctionnel, mais on peut encore tirer parti d'une intéressante astuce CSS 3.

Arrondir les coins

Nous avons examiné précédemment la propriété CSS 3 `border-radius`, en créant un cadre à bords arrondis simple pour une image. Ici, nous allons utiliser des propriétés similaires pour donner des bords arrondis à une forme simple (l'élément sélectionné de la liste dans notre système de navigation).

Nous utilisons des variétés propres aux navigateurs de `border-radius-top-left` et `border-radius-top-right` pour n'appliquer les bords qu'au haut de la forme :

```
ul#route_nav li.cur {
    background:#dff1f1;
    -moz-border-radius-topleft:3px;
    -webkit-border-top-left-radius:3px;
    -moz-border-radius-topright:3px;
    -webkit-border-top-right-radius:3px;
}
```

Cet ajustement simple au code CSS produit, dans Firefox et Safari, l'onglet subtil présenté à la Figure 11.22.

Figure 11.22

Map **Elevation** Download GPS Full routsheet   

L'onglet sélectionné possède maintenant des coins supérieurs arrondis, avec des CSS uniquement.

Notre système de navigation terminé, passons maintenant aux données d'altitude.

Le graphique d'altitude

Sous la barre de navigation de la zone Your latest route, nous allons maintenant ajouter le panneau de statistiques qui apparaît par défaut dans l'étude de cas, soit le graphique d'altitude (Elevation).

```
<div id="route_elevation">
</div>
```

Le point le plus important à noter est que nous utilisons la div `route_elevation` pour dégager les listes de navigation flottantes. `clear:both` est donc immédiatement défini pour cette div.

```
.home div#route_elevation {
    clear:both;
    background:#dff1f1 url(..../images/site/elevation_home.gif) 0 bottom no-repeat;
    position:relative;
    height:195px;
}
```

Nous attribuons aussi un positionnement relatif à la div, afin de pouvoir par la suite tracer les éléments li (nous les ajouterons sous peu) sur le graphe. Nous appliquons également l'image d'arrière-plan `elevation_home.gif` (voir Figure 11.23) avec une règle raccourcie pour les propriétés de couleur, d'image, de position et de répétition.

Figure 11.23

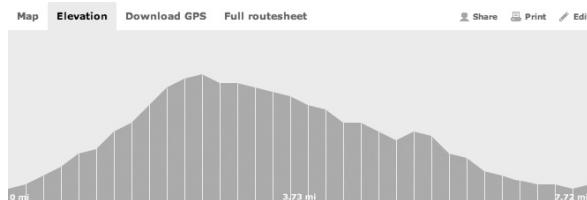
L'image d'arrière-plan d'altitude.



On obtient ainsi l'image présentée à la Figure 11.24, qui combine les données de navigation et le graphique d'altitude.

Figure 11.24

La navigation et le conteneur d'altitude combinés.



Ensuite, nous pouvons créer une nouvelle liste à puces qui contiendra les références d'altitude. Chaque référence contiendra une hauteur et une image associée provenant de Flickr. Notez que chaque liste possède une classe unique, comme `marker_01` et `marker_02`.

```
<div id="route_elevation">
  <ul>
    <li class="marker_01">
      <a href="#">
        <strong>1,442 ft</strong>
        
      </a>
    </li>
    <li class="marker_02">
      <a href="#">
        <strong>3,133 ft</strong>
        
      </a>
    </li>
    <li class="marker_03">
      <a href="#">
        <strong>2,398 ft</strong>
        
      </a>
    </li>
    <li class="marker_04">
      <a href="#">
        <strong>1,286 ft</strong>
        
      </a>
    </li>
  </ul>
</div>
```

```


</a>
</li>
</ul>
</div>

```

Nous utiliserons ces classes uniques pour définir la position d'affichage de chaque élément de liste sur le graphique, en définissant les coordonnées à partir du coin supérieur gauche du conteneur de graphique d'altitude.

```

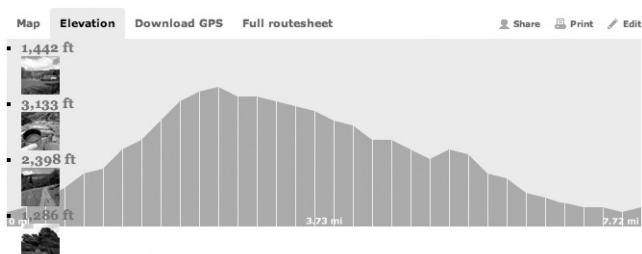
.home div#route_elevation li.marker_01 {
  top:123px;
  left:97px;
}
.home div#route_elevation li.marker_02 {
  top:50px;
  left:237px;
}
.home div#route_elevation li.marker_03 {
  top:95px;
  left:377px;
}
.home div#route_elevation li.marker_04 {
  top:137px;
  left:517px;
}

```

Nous ne sommes qu'en cours de chemin, aussi ne disposons-nous pour l'instant que d'un paquet d'éléments de liste, avec pour chacun une hauteur et une image, empilés à gauche du conteneur (voir Figure 11.25).

Figure 11.25

Les éléments de liste sont empilés à gauche.



Avant de forcer les éléments de liste à se situer aux emplacements désirés, commençons par masquer les images et choisissons un graphisme de marqueur plus approprié. Le graphisme que nous allons utiliser correspond au point classique présenté à la Figure 11.26. Le fichier est nommé elevation_marker.png.

Figure 11.26

Le graphisme de marqueur.



Passons maintenant à la mise en forme. Nous allons définir des styles comme `font-family`, `font-size`, supprimer ou définir des marges et des remplissages, etc. Fait important à noter, nous déclarons l'image d'arrière-plan `elevation_marker.png` pour l'élément `div#route_elevation ul li`, en appliquant une marge gauche de 15 pixels afin de créer un espace entre le point et la mesure de l'altitude.

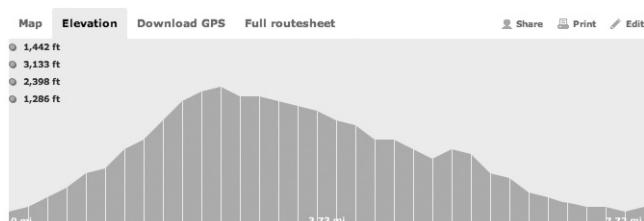
```
div#route_elevation ul {
    list-style:none;
    margin:0;
    font-family:Verdana,sans-serif;
    font-size:9px;
    font-weight:bold;
}
div#route_elevation ul li {
    margin:0;
}
div#route_elevation ul li a {
    color:#333;
    display:block;
    background:url(..../images/site/elevation_marker.png) no-repeat 0 5px;
    padding:0 0 0 15px;
}
div#route_elevation ul li a:hover,
div#route_elevation ul li a:focus {
    color:#000;
}
div#route_elevation ul li a img {
    display:none;
}
```

Nous avons ciblé les vignettes d'image de Flickr avec `div#route_elevation ul li a img`, en utilisant `display:none` afin d'empêcher les images de s'afficher.

Le graphique d'altitude présente maintenant la liste avec une mise en forme plus convenable, mais toujours en les empilant sur le côté gauche du conteneur (voir Figure 11.27).

Figure 11.27

Les marqueurs de liste sont mis en forme, mais toujours empilés à gauche.



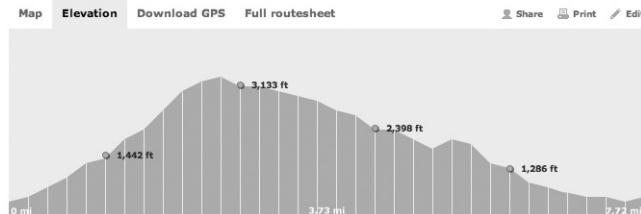
Afin de nous assurer que les éléments de liste trouvent leur emplacement en fonction des coordonnées de position, nous devons les positionner de manière absolue en utilisant `position:absolute`, comme ceci :

```
div#route_elevation ul li {
    margin:0;
    position:absolute;
}
```

Ce repositionnement fonctionne parce que le conteneur parent est pour sa part positionné de manière relative par rapport à son propre parent. Si ce n'était pas le cas, les éléments de liste positionnés de manière absolue feraient dépendre leurs coordonnées du coin supérieur gauche de la fenêtre du navigateur, ce qui ne conviendrait pas du tout. Comme le montre la Figure 11.28, les points d'altitude sont maintenant dessinés précisément sur le graphique.

Figure 11.28

Grâce au positionnement absolu, les marqueurs se placent aux coordonnées appropriées.



Nous devons maintenant gérer les images que nous avons masquées précédemment et les faire s'afficher lorsque l'utilisateur survole le point d'altitude.

Les cibles sont les états de pseudo-liens :`hover` et :`focus`. En les contrôlant, on peut aisément appliquer des déclarations qui afficheront les images en cas de survol.

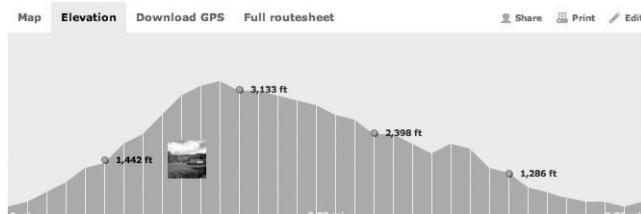
Vous remarquerez qu'après avoir défini la hauteur et la largeur nous positionnons de nouveau les éléments de manière absolue et nous utilisons des valeurs supérieure et droite négatives pour positionner les images exactement où nous le souhaitons par rapport au marqueur de la liste.

```
div#route_elevation ul li a:hover img,
div#route_elevation ul li a:focus img {
  display: block;
  width: 40px;
  height: 40px;
  padding: 4px 9px 10px 12px;
  position: absolute;
  top: -16px;
  right: -65px;
}
```

Quand l'utilisateur survole un marqueur, l'image s'affiche maintenant comme prévu, juste à droite du marqueur. Nous avons aussi ajouté un léger remplissage autour de la vignette (voir Figure 11.29).

Figure 11.29

Lorsque l'utilisateur survole un marqueur, sa vignette s'affiche.



Il est maintenant temps d'utiliser ce remplissage et d'insérer un joli cadre et une image de flèche sous les vignettes. J'ai utilisé pour cela le fichier `elevation_marker_image_bg.png` (voir Figure 11.30). Vous remarquerez que l'ombre est ajoutée à l'image dans Photoshop et exportée sous forme de PNG transparent.

Figure 11.30

Un graphisme soigné
à insérer sous les
vignettes.



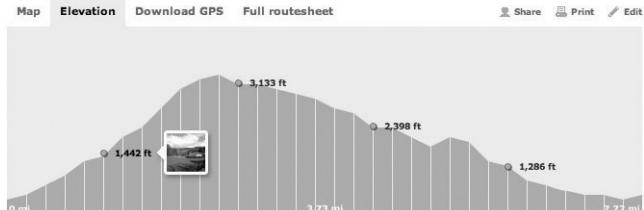
Il ne reste maintenant plus qu'à appliquer cette image d'arrière-plan aux CSS, comme ceci :

```
div#route_elevation ul li a:hover img,
div#route_elevation ul li a:focus img {
    display:block;
    width:40px;
    height:40px;
    padding:4px 9px 10px 12px;
    position:absolute;
    top:-16px;
    right:-65px;
    background:url(..../images/site/elevation_marker_image_bg.png) no-repeat 0 0;
}
```

Si vous procédez de la même manière, il se peut que vous deviez ajuster le remplissage, le positionnement et divers autres réglages, mais les éléments de construction sont bien en place. Tout le travail de cette section porte maintenant ses fruits : la page inclut une zone Your latest route, avec un graphique d'altitude interactif qui affiche des images de Flickr (voir Figure 11.31).

Figure 11.31

La section Your latest route, finalisée, avec de beaux survols d'images.



Tout cela devrait évidemment se trouver adossé à un puissant système de gestion de contenu pour créer un site vraiment exceptionnel, mais enfin, ce n'est ici qu'un livre sur les CSS.

En résumé

Et voilà. Ce moment de détente aura été plus studieux qu'un séjour de deux semaines au bord de la mer, mais j'ose espérer qu'il aura aussi été productif. J'ai particulièrement apprécié ce travail de mise en application des concepts du site CTM pour cette seconde édition du livre, notamment parce qu'il était possible d'explorer librement de nouvelles fonctionnalités CSS 3.

J'aurais évidemment aimé pouvoir présenter bien d'autres aspects du site CTM, mais un livre ne suffirait pas pour toutes ces explorations. Si vous appréciez telle fonctionnalité ou tel aspect du site qui n'est pas traité dans ce livre, vous ne devriez cependant pas avoir de difficulté à vous plonger dans le code source, à l'étudier et à comprendre comment les choses fonctionnent. Je me suis efforcé de structurer le code avec autant de clarté que possible, en insérant le maximum de commentaires utiles.

Le site reste en ligne à l'adresse <http://www.climbthemountains.com/cssm> et le code source est disponible sur le site www.pearson.fr. Téléchargez-le, examinez-le, observez-le sous toutes ses coutures, réassembllez-le et utilisez-le pour inspirer vos propres projets et vos propres chefs-d'œuvre CSS.



Index

Symboles

960 188
@font-face 241

A

active 92
Ajax 247
AlphaImageLoader 82
Arrière-plan 61
 défilement parallaxe 84
 dégradé 157
 effets de survol 101
 images multiples 71, 231
 PNG 210
 raccourci background 64
 remplissage 43

Attributs
 for 148
 rel 28
 sélecteur 232
 sélecteurs 27, 97
 summary 141
 title 27, 124
 XHTML 18

B

background 26, 64, 158
background-color 24, 56
background-image 61, 62
background-position 62
background-repeat 62
Biseautage 111
blockquote 6, 8, 23, 263
Blueprint 188
body 33, 37, 62, 260

Bogues 193
aide 202
caractères dupliqués 213
commentaires conditionnels 206
courants 211
décalage de texte de 3 pixels 211
du coucou 214
durée de vie 216
hacks et filtres 207
Internet Explorer 206, 216
layout 203
marge double 211
marges 196
spécificité 195
techniques 198
tests 201
Boîtes
 bords arrondis 64, 269
 box-sizing 45
 largeur 48
 niveau bloc 48
bookmarklet 19
border 147
border-bottom 112
border-collapse 143
border-image 74
border-radius 72, 157, 273, 277
Bords arrondis 269
Bordures 43, 44
 biseau 111
 de tableaux 143
 doubles 127
 Firefox 151
 images 74
 pointillé 28
Boutons 99
 biseau 111
 d'envoi 157

enfoncement 112
liens 117
radio 150
box-reflect 105
box-shadow 78, 234, 235, 274
box-sizing 45
button 157

C

caption 141
Cartes-images 123
 style Flickr 126
Cascade 30
Cases à cocher 150
 étiquettes 156
largeur 151
liens visités 94
multicolonnes 155
Cederholm, Dan 70
Çelik, Tanek 209
Centrage 165
Champs
 date 154
 obligatoires 152
Chrome
 first-child 268
 zoom 237
Citations 263
Clarke, Andy 28
Classes 9
 body 33
 combiner 272
 ou ID 11
 pseudo-classes 24
clear 56
Clearleft XI, 37
 équipe 123
Climb the Mountains 253

Code. *Voir aussi* HTML

- historique 6
- structurer 5
- validation 194
- Coins masqués 70
- col 142
- colgroup 142
- Colonnes 240
 - CSS 3 187
 - fausses 180
 - gouttière virtuelle 168
 - hauteurs égales 183
- colspan 8
- column-count 187
- column-gap 187
- column-width 187

Commentaires 38

- conditionnels 206
- mots-clés 39
- supprimer 39

Contour 44

Couleurs

- dégradés 62, 105
- héritage 34
- hexadécimales 21, 38
- ombres 78
- site noir et blanc 28
- variables 38

CSS 5

- cartes-images 123
- cascade 30
- commentaires 38
- CSS 3 17, 25, 45, 71, 72, 74, 78, 95, 97, 104, 187, 230, 235, 238, 270, 274
- fonctionnalités avancées 230
- frameworks et systèmes 188
- héritage 34
- importer 35
- info-bulles 106
- modèle de boîte 43
- ombres portées 75
- priorité des règles 31
- problèmes courants 194
- réinitialisation 257

sélecteurs 23

spécificité 31

sprites 103

unités 112

validation 194

versions 17

CSS-Discuss 202

Cufón 245

Curseur 134

D

- Date 154
- Davidson, Mike 87
- Défilement parallaxe 83
- Définitions 136
- Dégagement 54
- Dégradés 62, 105, 157
- display 107, 154
- div 13
- DOCTYPE 19, 199, 224
 - commutation 21
- DOM 247
- Données tableaux 139
- Downe, Natalie 176
- DTD 19

E

- Écrans 165
- Effondrement des marges 46, 196
- Éléments
 - blocs et incorporés 48
 - body 62
 - flottants 52
- Étiquettes 147
 - cases à cocher 156
- Études de cas
 - Climb the mountains 253
 - Roma Italia 221
- ExpressionEngine 255
- Extensions
 - Firebug 195, 228
 - Firefox 198
 - Web Developer 194, 199, 200
- Flux 98
- focus 92
- font-size 237
- Formulaires 146
 - boutons
 - d'envoi 157
 - radio 150
 - cases à cocher 150, 155
 - champs de saisie 149

F

- Fahrner, Todd 86
- Fausses colonnes 180
- Feuilles de styles
 - gestion 35
 - optimiser 39
 - reset 225
- fieldset 147, 150
- Filtres 207

- AlphaImageLoader 82
- étoile HTML 209
- passe-bande 209
- FIR 86
- Firebug 195, 228
- Firefox 108
 - arrière-plan 232
 - bordures 151
 - Cufón 246
 - extensions 198
 - Firebug 195, 228
 - first-child 268
 - legend 147
 - ombres portées 274
 - remplissage 110
 - sélecteur de cible 96
 - text-overflow 236
 - versions 216
- Web Developer 194, 199, 200
- zoom 237

- first-child 266, 268

- Flash 87
- Flickr 126
- Flottement 52, 116
 - clear 55
 - déplacement 54
 - marge double 211

- Flux 98
- focus 92
- font-size 237
- Formulaires 146
 - boutons
 - d'envoi 157
 - radio 150
 - cases à cocher 150, 155
 - champs de saisie 149

champs obligatoires 152
 étiquettes 147
 Frameworks CSS 188
 Fusion des marges. *Voir* Effondrement des marges

G

GIF
 animée 94, 250
 espacement 6
 transparence 7, 126
 Google Chrome. *Voir* Chrome
 Gouttière 43, 229, 259
 mises en page flottantes 167
 virtuelle 168
 Griffiths, Patrick 120
 Grilles 162, 228
 1080 pixels 226
 CTM 258
 Guides 162
 de styles 40

H

Hacks 207
 étoile HTML 209
 sélecteur d'enfants 210
 hanging-punctuation 238
 hCalendar 14
 height 44
 Héritage 34
 Homesite 7
 hover 92, 93, 100
 HTML
 body 33
 button 157
 conventions de nom 11
 div 13
 entités 239
 historique 6
 HTML 4 9
 HTML 5 10, 18, 223

microformats 14
 span 14
 structurer 37
 style 35
 validation 19, 194
 versions 17

ID 9
 body 113, 260
 ou classes 11
 sélecteurs 23
 spécificité 31
 superflus 233
 IFR 87
 Images
 arrière-plan 61, 71
 cartes-images 123
 élastiques 178
 espacement 6
 fluides 227
 GIF animée 94
 liquides 178
 ombres portées 75
 positionnement 182
 pour les bordures 74
 remplacement 85
 shim.gif 7
 sprites 103
 survol 101
 img 178

Importation 35
 Info-bulles 106
 Inman, Shaun 87
 input 150
 Internet Explorer 216
 arrière-plan 232
 bogue du coucou 214
 caractères dupliqués 213
 commentaires conditionnels
 206

Cufón 246
 décalage de texte de 3 pixels 211
 feuilles de styles séparées 258
 filtre de l'étoile HTML 210
 filtre passe-bande 209

first-child 268
 layout 203
 legend 147
 marge double 211
 modèle de boîte 45
 positionnement 52
 fixe 52
 site noir et blanc 28
 text-overflow 236
 transparence 82
 versions 216

iPhone 73

J

JavaScript 100
 bookmarklet 20
 insertion d'éléments de présentation 69
 menus déroulants 120
 positionnement fixe 52
 remplacement d'images 87
 sélecteur d'attribut 99
 jQuery 247, 248

L

label 147
 Largeur
 boîtes
 fixes 64
 incorporées 48
 cellules 7
 champs de saisie 149
 colonnes 38, 168

écran 165
fixe 101
liste 111
maximale 150
max-width 176
ombre 78
police 176
variable 178
zone de contenu 44
layout 203
legend 147
li 109
Liens 91
attribut rel 28
boutons 99, 117
cartes-images 123
cible 95
curseur 134
différencier 96
externes 96
Liens visités 94
relation 28
soulignements 93
survol 91
téléchargements 98
link 91
Liquid Fold 174
Listes 109
de définitions 136
flottantes 277
largeur 111
navigation 261
positionner 275
puces 119
résultats 250
list-style 124

M

Marcotte, Ethan 227
Marges 44
centrer la page 165
double pour les éléments flottants 211

effondrement 46, 196
paragraphes vides 47
margin 166
max-width 176, 179
Menus déroulants Suckerfish 120
Microformats 14
hCalendar 14
XFN 28
MIME 18
Mises en page 161
centrer 165
élastique 176
flottantes 167
grille à 1080 pixels 226
largeur fixe 172
liquide 173
Modèle de boîte 43
Internet Explorer 45
Modes
Quirks 21, 45, 166, 198
Standards 21, 199
moz 73

N

Navigateurs
bordures 151
Firefox 96, 108
legend 147
modes 21
Opera 110
prise en charge graduelle 217
Safari 96, 106
versions 216
zoom 237

Navigation
barre graphique 116
barre horizontale 114
barre verticale 110
liste 261
onglets 118
coulissants 68
séparateurs 117
signaler la page courante 113, 260

nofollow 28
Nouveautés
attributs 97
bords arrondis 72
box-sizing 45
cible des liens 95
colonnes 187
fonctionnalités avancées 230
images d'arrière-plan multiples 71
images de bordure 74
ombres 78
opacity 235
ponctuation hors justification 238
selecteurs avancés 25
survols 104
transparence alpha 270

O

Ombres 269
couleurs 78
portées 75
Onglets 118
coulissants 118
portes coulissantes 68
Opacité 80
opacity 80, 235
PNG 82
RGBa 81
Opera
Cufón 246
first-child 268
legend 147
remplissage 110
Orchard, Dunstan 75
outline 44
overflow 57

P

padding 44
Paragraphes vides 47

Parallaxe 83
Passe-bande 209
Phark 87
Photoshop 75
Pixy 102
PNG
 AlphaImageLoader 82
 arrière-plan 210
 transparence 82, 126, 230

Police

 eot 243
 liaison 237
 taille 176

Portes coulissantes 68, 118

Positionnement 48

 absolu 50
 fixe 52
 Internet Explorer 52
 listes 275
 pourcentage 182
 relatif 49, 124

Préfixes 73

Pseudo-classes 24

Q

Quirks 21, 45, 166, 198

R

Règles

 cascade 30
 guides 40
 priorité 31
rel 28, 114
Remerciements XIII
Remplissage 44
 arrière-plan 43
 gouttière 43
 traitement des navigateurs
 110

Rendu

 bordures 151

legend 147
modes 21
remplissage 110
Résolution 165
RGBa 234, 235, 270
Roma Italia 221
Rundle, Mike 87
Rutter, Richard 77

S

Safari

 arrière-plan 232
 Cufón 246
 dégradés 105
 extension CSS 106
 first-child 268
 legend 147
 ombres portées 274
 remplissage 110
 sélecteur de cible 96
 text-overflow 236
 versions 216
 zoom 237

screen.css 256

Sélecteurs 23

 avancés 25
 d'attribut 27, 232
 de cible 95
 de classe 24
 de frères adjacents 26, 268
 d'enfants 26, 210, 266
 de pseudo-classe 24
 descendants 23, 264
 d'ID 24
 universel 25

Snook, Jonathan 52

Soulignement 93

span 14, 132

Spécifications 17

 unités 112

Spécifité 30, 195

Sprites 103

Stackoverflow 202
Standards 21, 199
Staní ek, Petr 102
Styles 35
 réinitialiser 225, 257

submit 157

summary 141

Survols 91

 CSS 3 104

 distant 132

 GIF animée 94

 images 101

 image unique 102

 Pixy 102

Systèmes CSS 188

T

Tableaux

 bordures 143
 de données 139
 GIF d'espacement 7
 mise en forme 143

target 95

tbody 141

Téléchargements 98

text-align 166

textarea 150

Texte

 caractères dupliqués 213

 citations 263

 Cufón 245

 décalage de 3 pixels 211

 Flash 87

 format eot 243

 liaison des polices 237

 multicolonne 240

 ponctuation hors justification 238

 remplacer par des images 85

 taille 176

 zone 150

text-overflow 234, 236

tfoot 141
thead 141
title 27, 124
Transparence 269
 alpha 270
 AlphaImageLoader 82
 GIF 7, 126
 Internet Explorer 82
 PNG 82, 126
 RGBa 270

Typekit 241

Typographie

 Cufón 245
 liaison des polices 237
 ponctuation hors justification
 238

U

ul 109
Unités 112

V

Validation 19, 194
 bookmarklet 19
Veen, Jeffrey 241
visibility 58
visited 91, 95

W

W3C 17
Web Developer 194, 199, 200
webkit 73
width 44
Willison, Simon 99
WYSIWYG 6

X

XFN 28
XHTML 18
 1.0 Strict 224
XML 18
 DTD 19
XMLHttpRequest 247

Y

YUI Grids 188

Z

Zones
 de texte 150
 réactives 133

Le Campus

Maîtrise des CSS

2^e édition

CSS Mastery enfin traduit en français !

Véritable référence sur le sujet, ce livre contient tout ce que vous devez connaître pour passer maître dans l'art des CSS. Il regroupe les techniques les plus utiles, traite les problèmes concrets de navigateurs et aide à combler les lacunes les plus courantes.

Si la plupart des ouvrages se concentrent habituellement sur les compétences de base, celui-ci présuppose en revanche que vous avez déjà acquis les notions fondamentales et souhaitez approfondir vos connaissances afin de passer au stade supérieur en termes de programmation.

Vous apprendrez à :

- Planifier, organiser et gérer plus efficacement vos feuilles de styles CSS ;
- Mettre à profit les secrets des mises en page liquides, élastiques et hybrides ;
- Créer des boîtes à bords arrondis, des ombres portées et des reflets uniquement en CSS ;
- Maîtriser l'art de la mise en page des formulaires ;
- Pister et corriger les bogues de navigateurs les plus courants.

Enfin, toutes les techniques expliquées sont mises en pratique dans deux études de cas de haute facture, élaborées par deux des meilleurs concepteurs CSS de notre temps : Simon Collison et Cameron Moll.

Niveau : Intermédiaire / Avancé

Catégorie : Développement web

Configuration : Multiplate-forme

Table des matières

- Les fondations
- Des styles qui atteignent leur cible
- Vue d'ensemble du modèle de formatage visuel
- Utilisation des effets d'arrière plan
- Mise en forme des liens
- Mise en forme des listes et création de barres de navigation
- Mise en forme des formulaires et des tableaux de données
- Mise en page
- Bogues et correctifs
- Étude de cas : Roma Italia
- Étude de cas : Climb the Mountains

À propos des auteurs

Andy Budd est un designer web, développeur et auteur de blog de renommée internationale. Il s'est spécialisé dans la création de solutions web conformes aux standards, attrayants et accessibles.

Simon Collison est développeur en chef pour le web chez Agenzia (www.agenzia.co.uk). Il a travaillé sur de nombreux projets web pour des maisons de disques, des artistes de renommée, des graphistes et des illustrateurs majeurs.

Cameron Moll est un célèbre designer dédié aux nouveaux médias (web, terminaux mobiles).

PEARSON

Pearson Education France
47 bis rue des Vinaigriers
75010 Paris
Tél. : 01 72 74 90 00
Fax : 01 42 05 22 17
www.pearson.fr

friends of 
DESIGNER TO DESIGNER™
an Apress® company

ISBN : 978-2-7440-4130-3

