

Kaelig **DELOUMEAU-PRIGENT**

CSS maintenables

avec Sass/Compass

OUTILS ET BONNES PRATIQUES POUR L'INTÉGRATEUR WEB

Préface de Chris **Heilmann**

© Groupe Eyrolles, 2012, ISBN : 978-2-212-13417-9

EYROLLES



Préface

Lorsque les navigateurs ont commencé à interpréter les feuilles de style en cascade, bien des choses ont changé. Je me souviens encore du premier article sur les CSS, qui disait qu'on allait pouvoir multiplier par dix la vitesse des sites web et diviser par deux leur temps de maintenance et de développement. C'était vrai, à l'époque. Une connexion à Internet faisait le bruit crépitant d'un vieux fax, et un débit de 56 kilobits était encore un luxe. Quant à la mise en page, il fallait bricoler à l'aide d'images, de couleurs de fond, de tableaux et de balises ``. Et bien sûr, on espaçait les éléments à coups de gifs transparents et d'espaces insécables ` `.

Rien à voir avec aujourd'hui, où il est impossible de penser le design d'un site web sans CSS. Rien que l'année dernière, il y eut tant de nouveautés excitantes que même les animations et les transformations JavaScript seront bientôt aussi obsolètes que les gifs d'espacement d'autrefois. On peut désormais, en CSS, spécifier différents formats de police, ajouter nos propres polices, créer des motifs, des dégradés et, bien sûr, des transformations, des transitions et des animations. On peut même générer du contenu uniquement à des fins visuelles, et nous disposons d'une panoplie de sélecteurs dédiés aux actions de l'utilisateur – clic, survol à la souris, sélection, etc.

Évidemment, tout se paie... Il faut notamment, pour tenir compte de tous les navigateurs existants (et à venir), recopier de nombreuses fois les définitions de styles, à cause des différences d'implémentation qui nous empoisonnent la vie. Même si ce problème est censé disparaître un jour, cela n'empêche malheureusement pas les gens de persévérer dans leurs erreurs passées, en privilégiant un navigateur particulier sans prévoir de roue de secours pour les plus anciens. C'est l'erreur qui a été faite avec IE6, dont on paie le prix aujourd'hui avec des produits très difficiles à maintenir et à faire évoluer.

D'une certaine façon, c'est l'omniprésence des CSS qui en a ôté le côté magique. Elles sont vues aujourd'hui comme allant de soi, et c'est tout juste s'il existe de la documentation, ou même une réflexion, sur la façon de bien les structurer, de les écrire aussi propres et simples que possible. Nos très hauts débits et nos systèmes de cache performants ne sont pas les seuls en cause. C'est aussi le mépris des développeurs de langages « de haut niveau » qui a déchu les CSS au rang de bidonvilles de code, alors qu'il faudrait au contraire leur donner le même soin qu'à nos programmes en Python ou JavaScript.

La preuve : GitHub, dont le design reste somme toute assez simple, charge 400 Ko de CSS et plus de 2 000 lignes de code ! Quant à Facebook, le site en était à plus de 2 Mo de CSS avant que certains de mes amis y fassent le ménage et réécrivent l'ensemble avec une approche orientée objet. Enfin, on vient de découvrir un nouveau bogue d'Internet Explorer : il ne peut charger *que* 32 feuilles de style externes...

En tant que développeur un peu « vieille école » et artisan, l'idée même de *32 feuilles de style* m'a laissé dubitatif. Il faut reconnaître que la plupart des CMS créent une CSS par module y compris pour des gros sites, et qu'au lieu de les concaténer, ils se contentent de les ajouter aux templates...

J'ai autrefois écrit un document CSS de 300 lignes pour un gros site international. Six mois après avoir quitté la société, je suis retourné y jeter un coup d'œil. Le document avait enflé au point d'atteindre 3 800 lignes, avec des enchaînements à rallonge de sélecteurs et des noms de classes ultraspécifiques tels que `headingright3c6bold` ! Les malheureux développeurs chargés de faire évoluer l'apparence du site ne comprenaient visiblement rien au fonctionnement des CSS (ou ne s'en souciaient guère) et avaient ajouté, au petit bonheur la chance, des éléments HTML aux templates, pour pouvoir sélectionner facilement certains éléments spécifiques. Et ce n'était qu'un des nombreux crimes de lèse-CSS commis sous mes yeux.

Voilà pourquoi je me réjouis tant de voir apparaître une nouvelle tendance parmi les développeurs web : le désir de « nettoyer » les CSS existantes, pour revenir à la promesse initiale de légèreté et de rapidité.

Le livre que vous tenez entre les mains s'inscrit dans ce mouvement. C'est avec compétence, profondeur et vision que l'auteur traite un sujet que la plupart trouvent trop confus pour s'y intéresser. Je suis très heureux que cet ouvrage voie le jour et je souhaite que sa lecture vous donne l'envie de remettre l'art et la magie des CSS au cœur de vos feuilles de style.

Chris Heilmann,
développeur-évangéliste en chef du *Mozilla Developer Network*

Table des matières

Avant-propos	1	Les sites et blogs spécialisés	24
Pourquoi ce livre ?	1	<i>Smashing Magazine</i>	24
À qui est destiné ce livre ?	2	<i>A List Apart</i>	24
Structure du livre	2	<i>Alsacrations</i>	25
Remerciements	4	<i>Pompage.net</i>	26
 		<i>OpenWeb</i>	27
CHAPITRE 1		<i>Web Design Friday</i>	27
De à @font-face :		<i>24 Ways</i>	27
une problématique nouvelle	5	<i>CSS4 Design</i>	28
Aux débuts : pas de CSS,		<i>Mais aussi</i>	29
ni d'intégrateurs web	5	Aide entre développeurs	30
L'apparition de la feuille de style		À l'aide, j'ai un problème insoluble ! . . .	30
comme livrable	6	<i>Isoler le problème</i>	31
CSS aujourd'hui : l'importance des standards,		<i>Outils, partage et entraide</i>	32
l'impact des navigateurs	8	<i>Documenter son code pour référence</i> . . .	35
Le futur des CSS	11	Les logiciels libres : un gisement inépuisable	
Un Web adaptatif	11	d'exemples	36
Les régions CSS	12	 	
Les exclusions	14	CHAPITRE 3	
La liberté typographique	14	Bonnes pratiques pour un code	
Nouvelles structures de langage	17	lisible et maintenable	39
 		Un <head> à toute épreuve	39
CHAPITRE 2		Le doctype	39
Première plongée dans CSS	19	Un élément <html> polyvalent	
Apprendre les CSS :		avec Modernizr	40
l'incontournable théorie	19	Le mode compatibilité	
Comprendre les CSS, c'est comprendre		d'Internet Explorer	45
les navigateurs	21	L'encodage Unicode	47
Une affaire d'expérience	22	Récapitulatif	48
Être en veille constante	24	reset.css : réinitialisation des styles	50

Le reset universel : à éviter	51
Les reset.css sélectifs	52
Bien ranger ses fichiers	55
Un seul fichier ? Attention	
aux débordements	55
À chaque fichier sa fonction	55
<i>reset.css</i>	55
<i>typo.css</i>	56
<i>layout.css</i>	56
<i>forms.css</i>	57
<i>global.css</i>	57
<i>application.css</i>	59
<i>print.css</i>	59
Écrire un code lisible	60
Conventions syntaxiques	60
Notation multiligne ou monoligne ?	61
<i>Round 1 : lisibilité</i>	
<i>et maintenabilité</i>	61
<i>Round 2 : CSS3</i>	63
<i>Round 3 : gérer le suivi des versions</i>	64
<i>Vainqueur du match</i>	65
L'indentation : la clé de la lisibilité	65
<i>Au sein d'un sélecteur</i>	66
<i>Relations entre les sélecteurs</i>	67
<i>Indenter les valeurs CSS3</i>	68
Classer les propriétés	68
<i>Par ordre alphabétique</i>	69
<i>Par type de déclaration</i>	70
Utiliser les raccourcis	71
Documenter son code	71
Avant même de documenter,	
écrire un code explicite	71
Syntaxe des commentaires CSS	73
Quand ne pas (trop) documenter ?	74
Dégradation gracieuse et amélioration progressive	75
Dégradation gracieuse : la tolérance	
à l'erreur	75
Amélioration progressive	77
Simplifier un design trop complexe	79
Tester, encore et encore	80

Tests d'interopérabilité	
(multinavigateur)	80
Tests de non-régression	81
Une page pour tester tous les styles :	
le pattern portfolio	82

CHAPITRE 4

Pragmatisme : démystifier certaines bonnes pratiques 85

Mythe n° 1 – Pas d'informations de présentation dans le HTML	86
Mythe n° 2 – Ne pas utiliser d'éléments non sémantiques	90
Mythe n° 3 – On ne doit pas créer de classes non sémantiques	91
Mythe n° 4 – Le rendu doit être le même dans tous les navigateurs	92
Mythe n° 5 – La taille du texte ne doit pas être exprimée en pixels	94

CHAPITRE 5

Les frameworks CSS 97

Pourquoi utiliser un framework CSS ?	97
Ne pas réinventer la roue	97
Travailler en équipe	98
Prototyper rapidement	98
Découvrir pour apprendre	99
Quand ne pas s'en servir	100
Frameworks les plus populaires	100
Blueprint, le « tout-en-un »	100
Twitter Bootstrap, adaptatif et responsable	102
960.gs : une simple grille de mise en page	103
HTML5 Boilerplate : un concentré de bonnes pratiques	104
OOCSS : CSS orientées objet	105
Objets CSS	106
Séparation de la structure et de l'apparence	108
<i>Exemple : boutons d'action</i>	108

<i>Arguments</i>	164
Héritage de classe : @extend	166
Inclure un fichier : @import	167
Le module de couleurs	168
<i>Normalisation automatique</i>	168
<i>Opérations sur les couleurs</i>	168
<i>Transparence, opacité</i>	169
<i>Saturation</i>	170
<i>Variation de la luminosité</i>	170
Bonnes pratiques en Sass	171
Mixins : attention au surpoids	171
Imbrications : la règle de l'inception	172
<i>Un problème répandu</i>	172
<i>Le cauchemar de la surimbrication</i>	174
<i>La règle de l'inception</i>	174
Performances : concaténer et compacter les CSS	175
Compass : un métasframework	179
Premiers pas avec Compass	180
Compass dans la pratique	184
Reset.css en une ligne	184
Plus jamais d'images manquantes	187
Des images et des fontes embarquées	188
Dimensions d'une image	189
Des sprites sans aucun effort	190
<i>Rappels sur les sprites CSS</i>	190
<i>Présentation du module de sprites de Compass</i>	193
<i>Créer un bouton avec Compass</i>	194
CSS3 facile sans préfixe	196
<i>Qu'est ce qu'un préfixe constructeur ?</i>	196
<i>Box-shadow</i>	197
<i>Border-radius</i>	198
<i>Des dégradés sous tous les navigateurs</i>	199
<i>Transformations</i>	200
<i>Transitions</i>	201

CHAPITRE 9

Erreurs de conception : comment les débiter 203

L'usage des CSS sur le Web aujourd'hui	203
Les différents producteurs de feuilles de style	205
Outils pour diagnostiquer	206
L'inspecteur web et Firebug	206
Diagnostic en ligne avec CSS Lint	208
Les styles inline (dans la source du HTML)	209
La guerre des règles !important	210
Les règles surspécifiées	210
La multiplication des propriétés	211

CHAPITRE 10

Méthode : coder un design de zéro 213

Étudier le design	213
Déduire des motifs récurrents	214
Définir la base typographique	215
Coder la structure générale	218
Coder les modules	219
Étape finale : les exceptions	221

CHAPITRE 11

Méthode : faire le ménage dans des CSS 223

À partir de quand, et jusqu'à quel point optimiser ?	223
Étape 1 : utiliser les variables à bon escient	224
Les couleurs	224
Les jeux de polices	225
La grille de mise en page	227
<i>Générer une grille de mise en page simple et maintenable</i>	228
Étape 2 : séparation en plusieurs feuilles partielles	231
Étape 3 : imbrication des sélecteurs	232

Étape 4 : création de mixins	235
Idées de mixins	236
Étape 5 : extensions de classes	237
Exemple : étendre une boîte arrondie	237
Exemple : optimiser un mixin	238
Bonus : débogagecross-browser	
sans hacks	240
Méthode 1 : classes et commentaires	
conditionnels	242
Méthode 2 : feuilles de style séparées	
avec Sass	243
<i>Le principe</i>	243
<i>En application</i>	244
Index	249

Avant-propos

Pourquoi ce livre ?

Au cours de mes années de pratique en tant qu'intégrateur indépendant et en agence web, j'ai remarqué que la formation des développeurs de systèmes en HTML/CSS pouvait laisser à désirer. Pendant trop longtemps, HTML et CSS sont restés des langages de seconde zone, dont on confie le codage à un stagiaire ou au premier développeur venu.

Les usages actuels du Web sont tellement divers et omniprésents que l'utilisateur passe de plus en plus de temps devant son navigateur. Or, la couche applicative la plus proche des utilisateurs est justement le trio HTML/CSS/JavaScript. Et plus ça va, plus les utilisateurs sont exigeants sur l'ergonomie, la rapidité d'exécution et la compatibilité d'un même site entre leurs différents périphériques (mobile, ordinateur de bureau, tablette, téléviseur...). Plus question de laisser une tâche aussi sensible entre les mains d'un néophyte !

Le métier d'intégrateur répond à ce besoin de développeurs spécialistes de l'affichage des pages web dans le navigateur. Malheureusement, le manque de conventions universelles et de barèmes dans le métier mène à des conceptions inégales en termes de qualité, de productivité et de maintenabilité. Les workflows existants ne permettent plus de faire face à l'incroyable complexité et à l'étendue des possibilités offertes par CSS.

Que vous soyez en train de concevoir le prochain Twitter ou votre site personnel, les bonnes pratiques de maintenabilité, de performance et de qualité constituent toujours le meilleur chemin à suivre.

Ce livre tente de répondre aux questions que se posent les développeurs et designers web.

- Comment apprendre CSS et rester au courant des dernières nouveautés en la matière ?
- Comment organiser son travail pour collaborer en harmonie avec son équipe ?
- Comment produire un code maintenable pour un site qui évoluera ?
- Comment améliorer les performances d'affichage et de chargement d'un site existant ?
- Comment se servir des préprocesseurs ? Quels sont leurs bénéfices (et dangers) sur la qualité de notre travail ?

À qui est destiné ce livre ?

Ce livre est destiné à toute personne qui sera amenée à toucher du code sur des projets web d'une certaine envergure.

- Les intégrateurs web, développeurs, webmestres, designers web...
- Les développeurs qui aimeraient améliorer la qualité de leur code sans sacrifier leur productivité.
- Les designers web qui en ont assez de « bidouiller » en HTML/CSS et souhaitent passer à la vitesse supérieure en comprenant les enjeux du Web d'aujourd'hui.
- Les amoureux du code élégant, maintenable et collaboratif, désireux de découvrir de nouvelles manières d'affiner leurs connaissances dans des domaines aussi variés que les préprocesseurs CSS, le travail en équipe et les méthodes de nettoyage du code dans les gros projets.

Structure du livre

Ce livre est résolument tourné vers la démonstration par l'exemple. Dans la mesure du possible, chaque exemple est contextualisé, analysé et expliqué. Vous trouverez donc régulièrement des morceaux de code rencontrés dans la vie réelle, et des conseils que vous pourrez directement appliquer dans vos futures productions.

Dans un premier temps, nous balaierons les bases pour aller vers des étapes plus techniques et plus élaborées à la fin. Ainsi, la première moitié du livre aborde les principes de la coopération en équipe et les bonnes pratiques pour créer un code maintenable et élégant.

La seconde partie du livre traite des préprocesseurs CSS, plus spécifiquement Sass (et Compass), outils très pratiques pour produire une intégration toujours plus maintenable et améliorer sa productivité.

QUI SUIS-JE ?

J'ai commencé la création de sites web au collège par la fonctionnalité d'export HTML de Microsoft Publisher. Mon travail de fin d'année de quatrième, rendu sur CD-Rom, ne sera jamais publié sur le Web. Les professeurs sont dépaysés par l'initiative et je trouve un nouveau média universel pour faire circuler l'information.

Ensuite, je découvre Dreamweaver et CSS en recopiant les feuilles de style trouvées çà et là dans la source des sites que je visite. Avec un ami, nous concevons un site consacré au jeu vidéo Starcraft : l'époque de Multmania, des modems 56k et des écrans 14 pouces nous pousse à optimiser le poids des pages web. À cette époque, le moindre octet est précieux. Je n'aurais jamais imaginé que ce souci du détail pour les performances me rattraperait lors de la conception de sites mobiles !

C'est après un baccalauréat scientifique obtenu en 2003 tant bien que mal que je rentre à l'université pour étudier la psychologie. Passant plus de temps devant mon ordinateur qu'en cours, je me lance dans un projet web : Placebo City, qui deviendra plus tard ma carte de visite pour postuler à LunaWeb. Ce site communautaire consacré au groupe de musique Placebo m'a permis d'expérimenter de nombreux domaines liés à la création d'un site : développement, hébergement, graphisme, référencement, gestion de communauté, publicité, e-réputation...

Par la suite, je fais quelques à-côtés en produisant des sites à titre amateur pour de petites sociétés ou associations, puis je postule en 2007 chez LunaWeb comme développeur web. C'est au sein de cette agence web que j'ai pu faire mes armes en tant qu'intégrateur, exerçant enfin ma passion à titre professionnel. Dans le cadre de ce travail, j'ai eu la chance d'être entouré de gens très compétents qui m'ont permis d'apprendre ce métier dans des conditions que je qualifierais d'idéales, en travaillant aussi bien sur des projets de grande envergure que sur des petites créations expérimentales. Au fil des années, je me suis donc éloigné des logiciels de mise en page pour me tourner vers une conception au plus proche de la base technique des sites web à l'aide d'un éditeur de texte brut et de préprocesseurs CSS.

Aujourd'hui j'anime un blog, *Le ministère de l'intégration*, sur lequel on cause de design adaptatif, ou *responsive web design*, de HTML, CSS, Sass et Compass.

En juin 2012, j'intègre une nouvelle équipe en qualité de développeur d'interfaces à la division « Design adaptatif » de la BBC à Londres.

Quand je ne suis pas devant un écran pour faire ma veille sur Twitter (@kaelig), créer des sites web ou regarder une bonne série, j'aime écouter de la musique, déguster un bon vin entre amis et encore plus faire les deux à la fois.

► <http://blog.kaelig.fr>

Remerciements

Mes premières pensées vont à mes parents qui m'ont beaucoup aidé lors de l'écriture de ce livre, et tout particulièrement mon père pour sa relecture minutieuse. J'aimerais aussi remercier pour leur rôle (parfois indirect) dans l'aventure que ce livre représente : Muriel des éditions Eyrolles, les deux Chris, Hampton, Nathan, toute l'équipe de LunaWeb présente et passée, Pierre, Boris, Ugo, et Nicolas mon parrain qui m'a fait découvrir *Star Wars* et permis de mettre un pied dans le monde professionnel. Enfin, un salut respectueux aux bêta-lecteurs techniques : Raphaël, Gaël-Ian et Mehdi, je vous dois une bière (ou quatre), les gars !

De à @font-face : une problématique nouvelle

1

Après des prémices chaotiques, le Web arrive peu à peu à maturité. D'un monde très artisanal, on se dirige vers une industrie aux codes de plus en plus précis : métiers, standards, organisation, homogénéisation... Retraçons l'histoire du métier de développeur web côté client, aussi appelé intégrateur web.

Aux débuts : pas de CSS, ni d'intégrateurs web

Le métier d'intégrateur n'est pas né en même temps que le Web, loin s'en faut. À l'origine, le *World Wide Web* devait permettre à des chercheurs de partager de l'information scientifique entre eux ; que les pages soient ou non stylées n'entrait pas dans le champ de leurs préoccupations.

Ceux qui ont vécu cette époque se souviennent que dans les années 1990, les navigateurs étaient des logiciels rudimentaires (et parfois payants), les connexions très lentes et les écrans bien moins larges que ceux d'aujourd'hui.

En 1994, le *World Wide Web Consortium* (W3C) est fondé par un héros des temps modernes, l'inventeur du Web Tim Berners-Lee. Le W3C a pour mission de promouvoir un Web pour tous, se basant sur des technologies universelles validées par le même organisme, et que l'on appelle les standards du Web.

La Toile à ses prémices peut être décrite comme un outil à l'usage d'une élite intellectuelle désireuse de partager le fruit de sa pensée, notamment à l'aide de HTML, un langage inventé pour lier entre elles différentes ressources d'Internet.

Aux balbutiements du Web, styler un document se limite grosso modo à modifier le corps du texte et à insérer une image. À cette époque, ni le grand public, ni le monde du graphisme ne s'intéressent encore au Web. Heureusement, une nouveauté technologique va venir bouleverser l'austérité ambiante : les feuilles de style en cascade, plus connues sous le nom de CSS (*Cascading Style Sheets*).

ANECDOTE Le premier site web : août 1991

Le premier site web mis en ligne a été publié le 6 août 1991. Intitulé *The WWW Project*, ses pages étaient hébergées sur info.cern.ch par le premier serveur web, un ordinateur de marque NeXT, la compagnie fondée par Steve Jobs, suite à son départ forcé d'Apple. Aujourd'hui, vous pouvez en trouver une copie sur le site du W3C.

▶ <http://www.w3.org/History/19921103-hypertext/hypertext/WWW/TheProject.html>

L'apparition de la feuille de style comme livrable

Au milieu des années 1990, alors que les sites commencent à proliférer sur la Toile, une question épineuse émerge : « Comment produire une charte graphique qui vaille pour l'ensemble d'un site, si ce dernier contient des dizaines et des dizaines de documents ? »

En effet, d'une page à l'autre, le créateur d'un site doit répéter les informations de présentation dans le code HTML lui-même, cette redondance posant un évident souci de maintenabilité. Les développeurs web de l'époque expriment en outre le souhait de pouvoir mieux contrôler l'apparence de leurs pages.

PAROLE D'EXPERT HTML, la victoire de la pratique sur la règle

Frédéric Kaplan nous livre une version courte de l'histoire du langage HTML, truffée d'anecdotes, dans laquelle il prend le recul nécessaire pour nous dévoiler les principes qui ont fait de HTML ce qu'il est aujourd'hui.

▶ <http://fkaplan.wordpress.com/2012/01/18/html-limprobable-resurrection-dune-langue-vivante/>

C'est en décembre 1996, après moult délibérations, que le W3C publie la recommandation *CSS Level 1*. C'est une révolution : les feuilles de style permettent alors de cibler un ou plusieurs éléments du document et de leur appliquer des styles sans avoir recours à des informations de présentation au sein du code HTML.

Pour colorer le texte des cellules d'un tableau, on n'écrit donc plus :

```
<table>
<tr>
  <td><font color="red">Une ligne</font></td>
  <td><font color="red">Encore une</font></td>
  <td><font color="red">Jean-Luc est dans le coin</font></td>
</tr>
</table>
```

Mais plutôt :

CSS pour styler les cellules de tableau

```
td { color: red; }
```

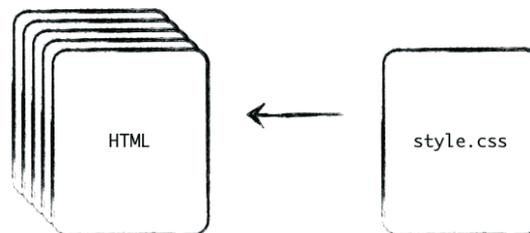
Code HTML du tableau

```
<table>
<tr>
  <td>Une ligne</td>
  <td>Encore une</td>
  <td>Jean-Luc est dans le coin</td>
</tr>
</table>
```

Et surtout, les CSS s'appliquent à un site entier là où chaque page devait être stylée indépendamment des autres auparavant. On imagine le gain de temps pour effectuer la maintenance et les évolutions du design !

Figure 1-1

Comme le dirait ce cher J. R. R. Tolkien :
« une feuille de style pour les gouverner tous ».



CSS aujourd'hui : l'importance des standards, l'impact des navigateurs

Hélas, il faut attendre que l'an 2000 pointe son nez pour que cette spécification soit implémentée correctement par les éditeurs de navigateurs. C'est Internet Explorer 5 Mac qui obtient le premier un taux de conformité frôlant les 100 %. Une belle prouesse... qui survient avec quatre ans de retard si l'on se fie au calendrier du W3C, avec des bogues persistants et pas d'implémentation des CSS niveau 2, dont la recommandation par le W3C date de 1998.

L'industrie du butineur est alors à la traîne et freine l'adoption des dernières avancées du Web par les professionnels de l'Internet : pour supporter les navigateurs de générations précédentes, les développeurs conservent leurs bonnes vieilles habitudes de codage en tableaux, aboutissant à des sites peu sémantiques, truffés d'informations de présentation. En raison du lourd retard technologique accusé par les navigateurs, CSS a donc du mal à s'imposer. Certains développeurs voient même dans ce nouvel apprentissage une perte de temps, considérant qu'un standard aussi mal implémenté dans les navigateurs est, de fait, voué à l'échec.

Pour des raisons de productivité, la tendance s'oriente vers une pratique qui sera fort dommageable au Web des années 2000 : développer ses sites pour un seul navigateur à la fois, celui qui domine le marché. D'abord Netscape (plus de 80 % de parts de marché à son apogée), puis Internet Explorer. Cette pratique donne lieu à un désastre général : des sites entiers cessent de fonctionner lorsqu'ils sont visités sur une plate-forme différente de celle pour laquelle ils ont été développés. Qui n'a pas en mémoire la mention apposée sur de nombreux sites web de l'époque : « Site optimisé pour Internet Explorer 6 » ? Devant une telle débâcle, les designers désireux d'exprimer leur talent graphique se tournent alors vers des technologies propriétaires comme Flash...

Malgré toutes ces difficultés, les recommandations du W3C s'imposent petit à petit comme la voie à suivre par les fabricants de navigateurs.

Les développeurs ne sont plus les seuls à toucher au HTML et certains designers commencent à mettre la main à la pâte. Le 7 mai 2003, Dave Shea dévoile CSS Zen Garden, un projet personnel lancé dans le but de montrer au monde des concepteurs web encore sceptiques que l'on peut réaliser des choses fantastiques à l'aide du langage CSS. L'idée est matérialisée sous la forme d'une page web codée en XHTML valide, associée aux feuilles de style de talentueux designers. Le principe est simple : chaque designer peut composer le design d'un document (dont la structure est imposée) grâce aux feuilles de style.

Figure 1-2
<http://www.csszengarden.com/>



Avec son « jardin zen », Dave ouvre la porte des standards du Web aux artistes. Il pave la route du futur en incitant les techniciens à faire fi des pratiques du passé pour se diriger vers un monde excitant et en pleine ébullition.

Le succès de ce site participatif est immédiat. On recense à ce jour 210 designs acceptés, sans compter les 1 000 designs « rejetés ». Excellent laboratoire de techniques et savoirs variés, CSS Zen Garden reste en 2012 une source inspiration pour quiconque souhaite appréhender sereinement le langage CSS.

Mais revenons à 2001 et à la guerre des navigateurs qui fait rage, Internet Explorer grignotant des parts de marché à Netscape qui peine à conserver son avance. Le navigateur de Microsoft doit alors faire face à un nouveau challenger : Firefox (qui a porté d'autres noms avant celui-ci : Phoenix puis Firebird). Le navigateur au panda roux, développé par la fondation Mozilla, devient de plus en plus populaire auprès des professionnels du Web jusqu'à toucher le grand public et atteindre de fortes parts de marché dans certains pays.

À l'époque, Firefox est le premier navigateur moderne capable de rivaliser avec Internet Explorer. Libre et gratuit, couramment mis à jour, plus rapide, multi-plates-formes et proposant un système de modules, Firefox présente de nombreux avantages comparé au navigateur au « e » bleu. Petit à petit, l'oiseau fait son nid : ce navigateur alternatif gagne la confiance des internautes. Le poids d'Internet devient tel dans les foyers que Microsoft doit réagir pour ne pas se faire devancer par ses concurrents.

Figure 1-3

Le fameux panda roux,
emblème de Firefox



Les ingénieurs de Redmond dégagent un peu tard, et sortent en 2011 la première version d'Internet Explorer qui soit respectueuse des standards, même si cette version 9 du navigateur ne supporte que partiellement les CSS3.

Pendant ce temps, Google lui aussi est entré dans la course avec Chrome, un navigateur basé sur le moteur de rendu WebKit, plus connu pour être à la base de Safari, le navigateur par défaut des ordinateurs Apple et périphériques iOS (iPad, iPhone, iPod touch).

La course est désormais à qui de Chrome, Firefox ou Internet Explorer proposera les avancées technologiques les plus spectaculaires. La cadence de mise à jour s'accélère pour suivre le train de l'innovation : Chrome, dont la première version a vu le jour en 2008, en est déjà à sa version 18 au moment où ce livre est publié. Mozilla a aussi revu sa feuille de route à partir de sa version 4 (2011) pour délivrer des mises à jour majeures quasiment tous les mois. Internet Explorer 10 doit voir le jour en 2012, après un cycle court, comparé aux nombreuses années qui ont séparé IE6 (2001) de IE7 (2006). Ce rythme soutenu dénote un fort désir d'avancer et trahit l'importance de l'innovation sur ce terrain.

Bien que certains modules de CSS3 ne soient pas encore passés au stade de recommandation par le W3C, les navigateurs implémentent déjà des propositions de nouveautés comme les animations, les transitions, les coins arrondis, les dégradés de couleurs... C'est avec les contributions du moteur de rendu WebKit que de nombreuses nouveautés de CSS3 sont apparues.

On peut dire que les navigateurs dits « modernes » ont devancé le W3C sur la plupart des questions techniques liées au HTML5 (CSS3, JavaScript, HTML), et que celui-ci devient en quelque sorte l'arbitre du match que se livrent les fabricants de navigateurs à coups de propositions innovatrices.

Si tout n'est pas rose dans le monde des navigateurs, on apprécie de voir une industrie entière se diriger vers une harmonisation technologique globale. Une belle vic-

toire pour les standards qui profite tout d'abord à l'utilisateur, le réel bénéficiaire d'un Web où tous les contenus sont accessibles quel que soit son terminal.

Le futur des CSS

L'avenir de l'Internet est tout tracé : des utilisateurs qui se connectent à de multiples services, sites et applications via une interface web. Pour satisfaire des besoins croissants en interactivité et rapidité, les navigateurs se perfectionnent, deviennent de plus en plus agiles au profit d'une expérience utilisateur accrue.

Pour relever les défis de demain, notamment l'utilisation sur différents terminaux (TV, mobile, tablette), il faut exploiter à bon escient les toutes dernières innovations en termes de conception web.

Jusqu'ici, on s'est contenté de la propriété `float` pour concevoir des mises en page à l'aide de blocs flottants... Mais après toutes ces années, il est vraiment temps que CSS offre un terrain de jeu bien plus adapté à ce type d'exercice.

Un Web adaptatif

Le Web se dirige clairement vers une consommation des contenus très orientée vers la mobilité. Pour permettre la consultation d'informations personnalisées sur tous nos périphériques high-tech, que l'on soit au bureau devant son écran d'ordinateur, dans le bus avec son mobile ou dans son salon avec sa tablette sur les genoux, les feuilles de style permettront bientôt d'adapter encore mieux la mise en page à chaque usage en particulier. Certaines techniques sont déjà disponibles comme les *Media Queries*. De gros efforts sont fournis sur ce module, tant par le W3C que par les fabricants de navigateurs.

La communauté se pose actuellement des questions relatives aux images adaptatives : comment servir une image de petite taille à un mobile sans pénaliser le rendu et les performances sur le même site visité depuis un ordinateur de bureau ? Actuellement, plusieurs solutions existent, à base de JavaScript et/ou de détection d'agent utilisateur, mais aucune d'entre elles n'est viable. Le W3C a lancé un appel à la communauté pour discuter de ce sujet entre professionnels du Web.

Cette discussion peut être suivie sur :

▶ <http://www.w3.org/community/respimg/>

Figure 1-4

Un site, plusieurs apparences.
Ici, le même blog sous
différents périphériques iOS.
<http://blog.kaelig.fr/>



Les régions CSS

Le futur de CSS est aussi du côté de la mise en page : nous verrons naître des innovations et de sensibles améliorations sur les modules de mise en page. Je pense notamment aux modules **Layout**, **Regions** et **Flexbox** (boîtes flexibles) qui sont trois des nouveautés CSS3 les plus prometteuses pour les éditeurs de contenus.

EN SAVOIR PLUS **Positionnement avancé en CSS**

Pour en savoir plus sur les nouvelles façons de pratiquer le positionnement des blocs avec CSS3, lisez le livre *CSS avancées* de Raphaël Goetter. Le chapitre 5 y est entièrement dédié.

📖 Raphaël Goetter, *CSS avancées*, Eyrolles 2012

À propos du module **Regions**, c'est en mai 2011 qu'Adobe a soumis au W3C une proposition de module dédié au « zoning » de page. Nous avons déjà à notre disposition des outils de mise en page, dont le positionnement absolu, les éléments flottants et les tables, alors pourquoi chercher plus loin ?

Passé au statut de brouillon (*Editor's Draft*) le 6 avril 2012, le module **Regions** CSS est un moyen avancé d'organiser un document selon un flux de contenu non-conventionnel, proche des possibilités de mise en page du monde du papier.

Le principe des régions est simple et se base sur un constat bien réel : aujourd'hui, nous savons étaler un texte sur plusieurs zones ou colonnes, mais pour cela, nous devons mettre en œuvre un découpage HTML complexe et quasi impossible à maintenir (il faut notamment calculer le nombre de caractères pouvant tenir dans chaque zone...). Avec les régions CSS, le document comporte une mise en page prédéfinie, à laquelle on fournit du contenu de manière brute. En s'adaptant au flux de texte délivré, les régions incorporent le contenu.

HTML

```

<div id="texte-source">
  Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor
  incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis
  nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
  Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu
  fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in
  culpa qui officia deserunt mollit anim id est laborum.
</div>
<div class="css-region" id="zone-1"></div>
<div class="css-region" id="zone-2"></div>
<div class="css-region" id="zone-3"></div>

```

CSS

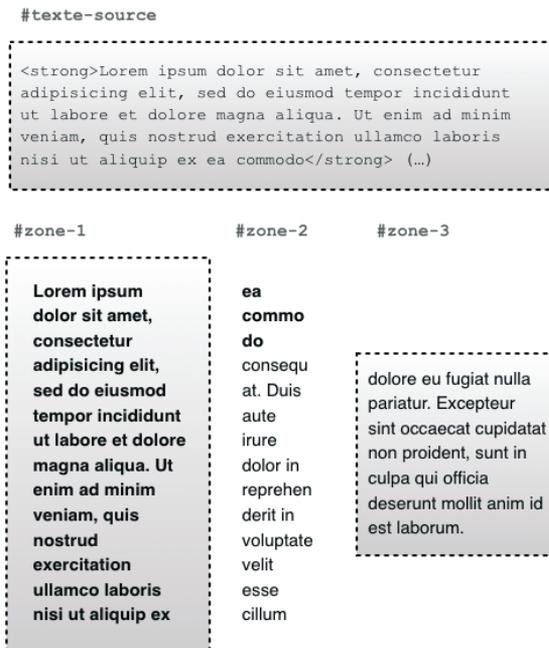
```

#texte-source {
  flow-into: contenu-principal;
}
.region {
  flow-from: contenu-principal;
}

```

Figure 1–5

Des colonnes de taille
différente avec les régions CSS



NORMALISATION Propositions d'Adobe au W3C sur les régions CSS

Adobe propose sur son site le détail de sa proposition au W3C ainsi qu'un descriptif du fonctionnement des régions. Vous trouverez aussi une documentation étendue et à jour sur le site du W3C.

- « *CSS Regions Module Level 3 Editor's Draft* » (Vincent Hardy, Alex Mogilevsky)
 - ▶ <http://dev.w3.org/csswg/css3-regions/>
- « *CSS3 regions: Rich page layout with HTML and CSS3* » (Arno Gourdol)
 - ▶ <http://www.adobe.com/devnet/html5/articles/css3-regions.html>
- « *CSS regions: Build complex, magazine-like CSS layouts using web standards* »
 - ▶ <http://labs.adobe.com/technologies/cssregions/>

Les exclusions

Aujourd'hui en CSS, nous n'avons pas la possibilité de commander au texte d'épouser les contours d'une image centrée dans un paragraphe. Nous sommes limités aux positionnements flottants à droite ou à gauche (`float: left;` et `float: right;`).

Les exclusions CSS3 viennent combler un manque dans le paysage des positionnements flottants : positionner un élément de nombreuses manières dans le flux du contenu. Il faudra encore attendre quelques mois avant de pouvoir en profiter. De plus, il n'est pas dit que des *polyfills* (équivalents pour navigateurs qui ne supportent pas cette fonctionnalité) existeront pour ce type de besoin. Quoi qu'il en soit, cette nouveauté est plutôt excitante : imaginez ce qu'elle peut apporter aux éditeurs dans la mise en page de leurs contenus !

NORMALISATION Brouillon Adobe et Microsoft sur les exclusions

Le *CSS Working Group* a publié un brouillon à l'initiative de deux ingénieurs de chez Adobe et Microsoft, Vincent Hardy et Rossen Atanassov.

- ▶ <http://dev.w3.org/csswg/css3-exclusions/>

La liberté typographique

L'industrie de l'impression papier a toujours possédé une avance considérable sur son concurrent numérique pour ce qui est de l'affichage typographique. Entre les polices crénelées (les fameux effets d'escaliers qui tendent à disparaître grâce aux écrans à haute densité de type Retina depuis l'iPhone 4), la mauvaise gestion des polices et le rendu disparate d'une plate-forme à l'autre, on peut dire que l'informatique a encore des efforts à fournir avant de rattraper le papier.

Toutefois, le contrôle de l'affichage typographique est de mieux en mieux supporté par les différents périphériques connectés. Nous avons déjà de magnifiques exemples sur la Toile conçus par des designers expérimentés dans le domaine de la typographie.

Figure 1–6

Une mise en page vivante grâce aux exclusions CSS
<http://dev.w3.org/csswg/css3-exclusions/>



Figure 1–7

Un usage intensif de @font-face sur le site de l'agence Yebocreative
<http://www.yebocreative.com>



Figure 1-8

Simple as Milk, un studio de design qui montre ce dont il est capable dès la page d'accueil.

<http://simpleasmilk.co.uk/>



Figure 1-9

Le site de la conférence Ampersand et ses invités de marque

<http://2012.ampersand-conf.com/>



Les nouveautés relatives à la typographie qui profiteront à la créativité sur le Web sont :

- @font-face : dites adieu à la domination d'Arial, Georgia et Verdana sur tous les sites web. Importez les polices qui vous plaisent pour aller plus loin dans le design ;
- la césure automatique (*hyphenation* en anglais) : pour une justification et un colonnage du texte toujours lisible ;

Index

l'important 209
@font-face 16
960 grid system 128, 179, 219,
227, 230

A

accessibilité 76
adaptatif 11
Adobe 12, 14
amélioration progressive 20, 42,
44, 75, 77, 89
Apple Store 151
application.css 59

B

Blueprint 23, 100, 121, 128,
179, 219, 227
Boilerplate 104
Bootstrap 102
box model *voir* modèle de boîtes

C

césure 16
Christophe Porteneuve 133
Chrome 43
classe
 extension 237
Clearleft 82
Closure Stylesheets 141
CMS 31, 81, 125, 130, 205, 211
CodeKit 148
coins arrondis 77
commentaires

conditionnels 241, 242
 syntaxe 73
commentaires
 conditionnels 241, 242
Compass 179
compilation Sass 146, 156, 159
conventions
 d'écriture 125
 de nommage 73, 126, 130
 syntaxiques 60
couleurs 224
cross-browser 240
CSS 6
 CSS Zen Garden 9
 documentation 20
 histoire 8
 Level 1 6
 régions 14
 Scaffold 142
 théorie 19
CSS3 21, 124, 196
 border-radius 198
 box-shadow 197
 coins arrondis 238
 dégradés 199, 244
 préfixe constructeur 196
 transformations 200
 vendor-prefixes 196, 197
CSS3Pie 93

D

débogage 240

dégradation 75
 gracieuse 89, 93
design
 conception 213
doctype 39
documentation 71, 128
DRY 109, 136, 160, 170, 172,
196, 214, 227

E

em 94
en-tête 39
exception 221
exclusions CSS 14

F

Facebook 90, 92, 138, 204
fichiers
 organiser 55
Firebug 206
float
 exclusions 14
flottant 14
fonds multiples 78
forms.css 57
framework 23, 97, 99, 227
 960 grid system 103
 Blueprint 100
 bootstrap 98, 102
 HTML5 Boilerplate 104
 pertinence 100

G

gestion de versions 64, 99, 132
Git 99, 128, 132, 133
GitHub 36, 104, 120
global.css 58
grep 212
grille 227

H

hack 240
Haml 65
head 39
HTML5 Boilerplate 23, 37,
104, 180
hyphenation 16

I

imbrication 156, 157, 172
indentation 65
InstaCSS 20
intégrateur web 5
intégration 88
Internet Explorer 8, 17, 40, 54,
76, 81, 93, 130, 208, 240,
242
émulation 46
mode compatibilité 45
interopérabilité 80, 92, 168
ISO-8859-1 47

J

JavaScript 76
jQuery 44

L

layout.css 56
Least 235
lisibilité 65
LiveReload 148

M

maintenabilité 241
media 106
Media Queries 11
meta X-UA-Compatible 45
microdata 88

microformats 90
mixins 236
mobile 51, 77
mode compatibilité 40
modèle de boîtes 40, 54
Modernizr 42, 77, 94
personnaliser 42
moteurs de recherche 112

N

Natalie Downe 82
navigateur 92
Netscape 23
Nicole Sullivan 30
Normalize.css 53, 55, 120
notation
classer les propriétés 68
CSS3 63
multiligne ou monoligne 61
sélecteur 66

O

ombré 77
ombres 42
OOCSS 30, 74, 105, 217
Opquast 85
optimisation pour les moteurs de
recherche 23, 79
optimiser 223
Orange 73, 211
Orange.fr 204

P

pattern library 128, 129
pattern portfolio 81, 82
Paul Irish 30
performances 2, 44, 51, 52, 76,
79, 237, 241, 243
pixel 94
polices 225
polyfills 14, 77
pragmatisme 21, 85, 86
print.css 59
productivité 130, 145, 150, 190
propriété

classer 68
multiplication 211
raccourcis 71

Q

quirks mode 40

R

raccourcis 71
Raphaël Goetter 20
référencement naturel 76
régions CSS 14
régions CSS3 12
rem 95
reset.css 50, 55, 184
responsive web design 11
Ruby 152
RubyGems 151, 152, 182

S

Sass 145
bonnes pratiques 171
commentaires 159
couleurs 168
extend 166, 227, 237
héritage 166
inception 174
installation 150
mixins 17, 143, 162, 235,
236, 238
surimbrication 174
variable 17, 160
Scaffold 142
Scout 148
sélecteur universel 52
sémantique
débat 86
microdonnées 87
sprites 175, 190
style guide 82, 128, 129
Stylus 141

T

taille
em 94

- pixel 94
- rem 95
- tests 31, 80, 81, 130
 - interopérabilité 80, 240
 - Litmus 82
 - non-régression 81
 - pattern portfolio 82
- Tim Berners-Lee 5
- Twitter 24, 29
- Twitter Bootstrap 23, 98, 120, 180
- typo.css 56

- typographie 14, 124, 215
 - Cleartype 17
 - @font-face 15
 - Windows 17

U

- Unicode 47
- UTF-8 47

V

- variable 224
- veille 24

W

- W3C
 - Adobe 14
- width 40
- wireframes 214

X

- Xcode 151

Z

- ZURB Foundation 180

- les ligatures entre caractères : l'élégance typographique à l'état brut.

Figure 1–10

Les ligatures, marques d'une typographie sophistiquée



Il ne s'agit pas tant d'attendre des avancées du côté du W3C, mais plutôt que ces fonctionnalités soient implémentées dans les navigateurs.

CONSEIL DE PRO La typographie sous Windows

Quand vous implémentez de telles nouveautés, n'oubliez pas de tester leur rendu dans les navigateurs et systèmes d'exploitation les plus courants. En effet, Windows propose différents modes de rendu typographique en apposant un traitement différent d'une machine à l'autre selon que la technologie de lissage ClearType est activée ou non. Attention aussi à vérifier l'apparence du texte sous Windows 7 qui lui-même apporte un moteur de rendu complètement revu (notamment sous Internet Explorer 9).

Nouvelles structures de langage

Les préprocesseurs CSS (pour en savoir plus sur ce sujet, consultez le chapitre 7) influencent fortement la qualité et la rapidité de développement, tout en ajoutant une couche de complexité pour le novice. CSS se veut être un langage démocratique, accessible à tous. Toutefois, il a été proposé au W3C d'intégrer des fonctionnalités présentes dans Sass (et WebKit pourrait les incorporer très bientôt de son côté) :

- les mixins, sortes de fonctions réutilisables (abordés dans le chapitre 7) ;
- les variables (déjà à l'état de brouillon W3C au moment de la publication de ce livre : <http://dev.w3.org/csswg/css-variables/>) ;
- les sélecteurs imbriqués (abordés dans le chapitre 7) ;
- les opérateurs : addition, soustraction, multiplication, division, trigonométrie...