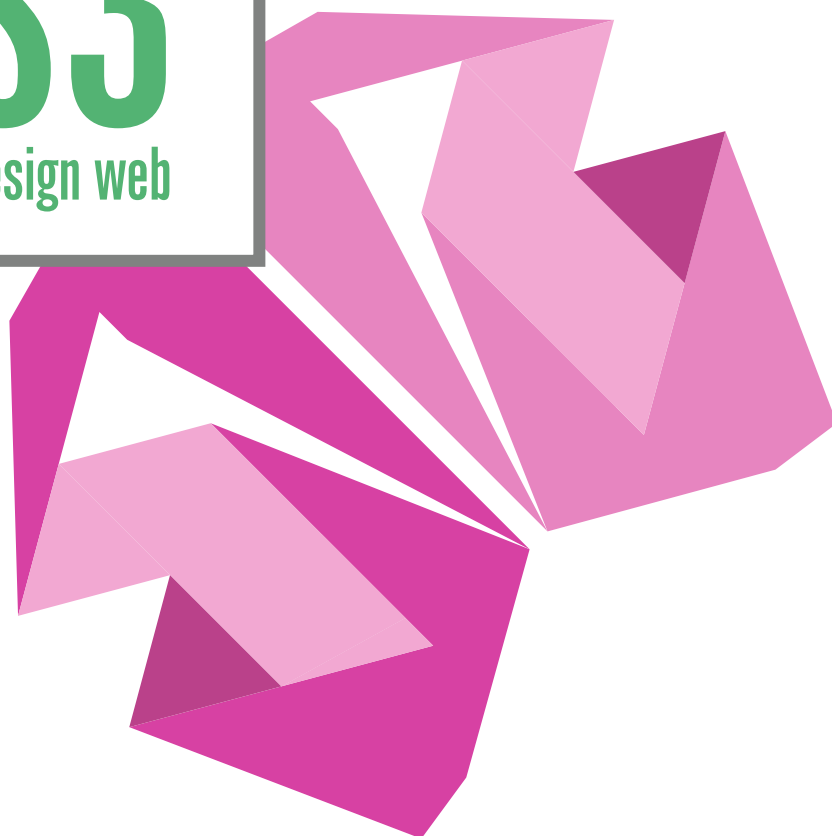


Hugo Giraudel
Raphaël Goetter

Préface de Chris Coyier

CSS3

Pratique du design web



BONUS EN LIGNE - 1 heure de formation video pour :

- Apprendre à créer une feuille de styles de A à Z
- Débugger votre feuille de styles

● Éditions
EYROLLES



Vingt ans après sa conception, le langage CSS n'en est plus à ses balbutiements et n'est plus optionnel en ce qui concerne la conception web moderne. Sans le moindre concurrent en vue, CSS a encore de belles années devant lui. Et pour cause, il est toujours en perpétuelle évolution !

Ce livre n'a pas pour prétention d'être le guide ultime de l'intégrateur dans la mesure où il ne reprend pas les bases. Il offre simplement une mise à niveau en levant le voile sur tous les modules CSS, afin d'offrir dès aujourd'hui les connaissances nécessaires à la réalisation de sites et d'applications web. En effet, les enjeux comme les objectifs ne sont plus les mêmes qu'il y a quelques années, aussi est-il important que les intégrateurs, designers et développeurs s'arment face aux nouvelles problématiques que sont le Responsive Web Design, le rétrécissement de l'écart entre le Web et le natif, et la course à la performance.

Qu'il s'agisse de mise en page avec Flexbox ou Grid Layout, d'embellissement des interfaces, d'élaboration d'animations ou même de design fluide avec les Media Queries, vous devriez être capable de maîtriser tous ces sujets au sortir de votre lecture.

Au-delà de l'aspect purement didactique de l'ouvrage, vous trouverez un grand nombre d'exemples et de mises en pratique, ainsi que tout ce que vous devez savoir vis-à-vis du support des fonctionnalités par les navigateurs. Pour finir, vous découvrirez dans les annexes la liste des valeurs par défaut des propriétés CSS, celle des propriétés que l'on peut animer et une bibliographie pour aller plus loin.

À qui s'adresse cet ouvrage ?

- Aux intégrateurs désireux d'aller plus loin avec CSS
- Aux designers souhaitant se mettre au design *in the browser*
- À tous les concepteurs de sites et d'applications voulant se mettre à niveau vis-à-vis des nouveautés du langage

Au sommaire

L'état actuel du W3C et des standards CSS • Les sélecteurs : l'accès au DOM • Positionnement et layout : les nouvelles techniques de mise en page • Interfaces graphiques et amélioration visuelle • De nouvelles unités et valeurs • Contrôle du texte • Variables natives • Styles conditionnels • Transformations : un nouveau monde en 2D et en 3D • Animations et transitions : pour des interfaces moins statiques

Développeur front-end passionné par CSS et auteur du site Browserhacks, **Hugo Giraudel** fait part de son expertise sur son propre blog ainsi que sur les sites SitePoint, CSS-Tricks, The Sass Way et Tuts+, entre autres. Enthousiasmé par le préprocesseur Sass, il a su s'imposer comme référence mondiale sur le sujet. Il est d'ailleurs l'auteur de SassDoc, un outil de documentation pour Sass.

Webdesigner et gérant d'une agence web strasbourgeoise, **Raphaël Goetter** partage ses connaissances à travers son site Alsacréations.com, et s'intéresse de près aux domaines des normes du Web et de l'accessibilité. Il fait partie du collectif Openweb.eu.org, référence francophone en matière de standards du Web.

www.editions-eyrolles.com
Éditions Eyrolles | Diffusion Geodif

Studio Eyrolles © Éditions Eyrolles

Code éditeur : 067896
ISBN : 978-2-212-67896-3

Hugo Giraudel

Raphaël Goetter

Préface de Chris Coyier

The logo for CSS3, featuring the letters 'C', 'S', 'S', and '3' in a bold, green, sans-serif font. The 'C' and 'S' are connected, and the '3' has a distinctive shape with a curved top.

CSS3

Pratique du design web

DANS LA MÊME COLLECTION

C. DELANNOY. – **Le guide complet du langage C.**

N°14012, 2014, 844 pages.

K. AYARI. – **Scripting avancé avec Windows PowerShell.**

N°13788, 2013, 358 pages.

W. BORIES, O. MIRIAL, S. PAPP. – **Déploiement et migration Windows 8.**

N°13645, 2013, 480 pages.

W. BORIES, A. LAACHIR, D. THIBLEMONT, P. LAFEIL, F.-X. VITRANT. – **Virtualisation du poste de travail Windows 7 et 8 avec Windows Server 2012.**

N°13644, 2013, 218 pages.

J.-M. DEFRANCE. – **jQuery-Ajax avec PHP.**

N°13720, 4e édition, 2013, 488 pages.

L.G. MORAND, L. VO VAN, A. ZANCHETTA. – **Développement Windows 8 - Créer des applications pour le Windows Store.**

N°13643, 2013, 284 pages.

Y. GABORY, N. FERRARI, T. PETILLON. – **Django avancé.**

N°13415, 2013, 402 pages.

P. ROQUES. – **Modélisation de systèmes complexes avec SysML.**

N°13641, 2013, 188 pages.

SUR LE MÊME THÈME

R. RIMELÉ, R. GOETTER. – **HTML 5 – Une référence pour le développeur web.**

N°13638, 2e édition, 2013, 752 pages.

C. SCHILLINGER. – **Intégration web – Les bonnes pratiques.**

N°13370, 2012, 390 pages.

S. POLLET-VILLARD. – **Créer un seul site pour toutes les plates-formes.**

N°13986, 2014, 144 pages.

E. MARCOTTE. – **Responsive web design.**

N°13331, 2011, 160 pages.

F. DRAILLARD. – **Premiers pas en CSS 3 et HTML 5.**

N°13944, 6e édition, 2015, 472 pages.

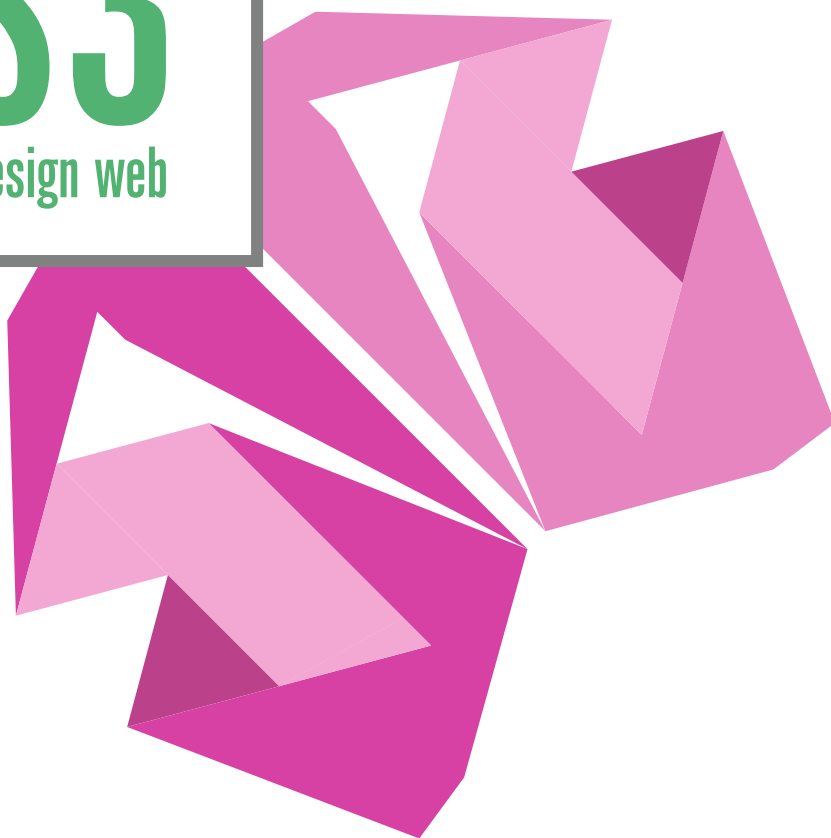
M. KABAB, R. GOETTER. – **Sass et Compass avancé.**

N°13677, 2013, 280 pages.

Hugo Giraudel
Raphaël Goetter
Préface de Chris Coyier

CSS3

Pratique du design web



ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris cedex 05

www.editions-eyrolles.com

En application de la loi du 11 mars 1957, il est interdit de reproduire intégralement ou partiellement le présent ouvrage, sur quelque support que ce soit, sans autorisation de l'éditeur ou du Centre français d'exploitation du droit de copie, 20, rue des Grands-Augustins, 75006 Paris.

© Groupe Eyrolles, 2015 - © Éditions Eyrolles, 2019 - ISBN : 978-2-212-67896-3
Crédits photographiques : © Alexandra Lucas

Préface

CSS occupe une place étrange dans le monde du développement. Il y a des développeurs qui dédaignent ce langage, ne le jugeant pas digne de leur temps ou de leurs compétences. Et, en même temps, ils en ont peut-être peur. Voyez-vous, CSS est à la fois très simple et très compliqué. Après tout, c'est essentiellement une suite de déclarations composées de paires de clés/valeurs : `color: red, width: 50%, display: inline-block`. Rien de bien compliqué en soi. Mais la façon dont ces déclarations interagissent les unes avec les autres peut être aussi complexe et casse-tête que n'importe quel autre langage de programmation. Ajoutez à cela les problèmes d'incompatibilité entre les navigateurs, et vous pourriez presque en venir à dire que le rôle d'intégrateur est l'un des plus difficiles qui soit !

Il y a aussi le fait que CSS évolue à une vitesse sans précédent. Ce que nous connaissons sous le terme de « CSS 3 » est à la fois amusant et puissant, mais rend aussi les choses plus complexes. Il existe désormais de nouveaux systèmes de mise en page comme Flexbox, des possibilités d'animation, et même de la 3D entièrement réalisée avec CSS ! Êtes-vous parfaitement au fait de `position: sticky` ? Qu'en est-il des unités de dimensionnement vis-à-vis du viewport ? Savez-vous que l'on peut tester le support des propriétés directement au sein des feuilles de styles ? C'est un nouveau monde plein de possibilités qui s'offre à vous !

Hugo Giraudel est la personne idéale pour vous présenter ces nouveaux concepts. Il parvient en effet à appréhender ces idées, puis à les expliquer de manière pédagogique et compréhensible. Je peux vous l'affirmer, non seulement parce que c'est ce qu'il fait dans ce livre, mais aussi parce que je lis ses articles techniques depuis des années. Il est parfait quand il s'agit d'expliquer des concepts bien spécifiques, et parvient à couvrir tous les détails nécessaires à la compréhension de ceux-ci. J'ai même fait appel à ses services pour m'aider à rechercher et à écrire des contenus pour mon propre site web (CSS-Tricks) ! C'est une chance que d'avoir Hugo à la tête de ce livre, réunissant autant d'informations intéressantes et pertinentes en un même ouvrage.

Chris Coyier,
auteur du site CSS-Tricks,
développeur en chef à CodePen,
podcasteur à Shop Talk Show

Table des matières

Avant-propos	1
CSS, un langage en perpétuelle évolution	1
Pourquoi cet ouvrage ?	1
À qui s'adresse cet ouvrage ?	2
La structure de l'ouvrage	2
Compléments vidéo à consulter en ligne	3
Remerciements	4
CHAPITRE 1	
L'état actuel du W3C et des standards CSS	5
Une évolution implacable	5
Un tour d'horizon des navigateurs d'aujourd'hui	7
L'état actuel des standards	8
L'aventure des préfixes constructeurs	9
La standardisation des CSS	11
Étape 1. Trouver une idée	12
Étape 2. Editor's Draft (ED)	12
Étape 3. Working Draft (WD)	12
Étape 4. Candidate Recommendation (CR)	12
Étape 5. Recommendation (REC)	13
CHAPITRE 2	
Les sélecteurs : l'accès au DOM	15
Les opérateurs	15
Opérateur d'adjacence directe (CSS 2)	16
<i>Compatibilité des navigateurs pour l'opérateur d'adjacence directe</i>	16
Opérateur d'adjacence générale	17
<i>Compatibilité des navigateurs pour l'opérateur d'adjacence générale</i>	18
Les sélecteurs d'attribut	18
Sélecteur d'attribut simple (CSS 2)	18
<i>Compatibilité des navigateurs pour les sélecteurs d'attribut</i>	19

Sélecteur d'attribut modulé	19
<i>Attribut dont la valeur est comprise dans une liste séparée par des espaces avec [attr~="value"] (CSS 2).</i>	20
<i>Attribut dont la valeur est exacte ou démarre par une chaîne précédée du caractère - avec [attr = "value"] (CSS 2)</i>	20
<i>Attribut dont la valeur débute par une chaîne avec [attr^="value"].</i>	21
<i>Attribut dont la valeur termine par une chaîne avec [attr\$="value"].</i>	22
<i>Attribut dont la valeur contient une chaîne avec [attr*="value"].</i>	22
Les pseudo-classes de position	22
Premier et dernier enfant avec :first-child & :last-child	23
<i>Compatibilité des navigateurs pour :first-child.</i>	23
<i>Compatibilité des navigateurs pour :last-child</i>	24
Énièmes enfants avec :nth-child et :nth-last-child	24
<i>Styler en fonction du nombre d'éléments dans le parent.</i>	28
<i>Compatibilité des navigateurs pour :nth-child et :nth-last-child</i>	30
Enfant unique avec :only-child	30
<i>Compatibilité des navigateurs pour :only-child.</i>	31
Propriétés *-of-type	31
<i>Compatibilité des navigateurs pour les pseudo-classes :*-of-type</i>	32
Les pseudo-classes de contexte	32
Ciblage par ancre avec :target	32
<i>Compatibilité des navigateurs pour :target.</i>	33
Éléments vides avec :empty et :blank	33
<i>Compatibilité des navigateurs pour :empty.</i>	34
Gestion de la langue avec :lang (CSS 2)	35
<i>Compatibilité des navigateurs pour :lang</i>	35
Élément racine avec :root	36
<i>Compatibilité des navigateurs pour :root</i>	36
Négation avec :not	36
<i>Compatibilité des navigateurs pour :not</i>	37
Simplification des sélecteurs avec :matches	37
<i>Compatibilité des navigateurs pour :matches</i>	39
Les pseudo-classes de formulaire	39
Focus avec :focus (en toute simplicité)	39
<i>Compatibilité des navigateurs pour :focus.</i>	40
État des champs de formulaire avec :enabled et :disabled	40
<i>Compatibilité des navigateurs pour :enabled et :disabled</i>	41
Modes d'écriture avec :read-write et :read-only	41
<i>Compatibilité des navigateurs pour :read-write et :read-only.</i>	43
Validité des champs de formulaire avec :valid et :invalid	43
<i>Compatibilité des navigateurs pour :valid et :invalid.</i>	44

Statut des champs de formulaire avec <code>:optional</code> et <code>:required</code>	45
<i>Compatibilité des navigateurs pour <code>:optional</code> et <code>:required</code>.</i>	45
Précisions sur les checkboxes et les boutons radio avec <code>:checked</code> et <code>:indeterminate</code>	45
<i>Compatibilité des navigateurs pour <code>:checked</code>.</i>	47
<i>Compatibilité des navigateurs pour <code>:indeterminate</code></i>	47
Valeur par défaut des boutons radio avec <code>:default</code>	47
<i>Compatibilité des navigateurs pour <code>:default</code>.</i>	48
Gestion de l'amplitude des champs de type number avec <code>:in-range</code> et <code>:out-of-range</code>	49
<i>Compatibilité des navigateurs pour <code>:in-range</code> et <code>:out-of-range</code></i>	49
Les pseudo-éléments	49
Évolution de la syntaxe	50
Contrôle de la sélection du texte avec <code>::selection</code>	50
<i>Compatibilité des navigateurs pour <code>::selection</code></i>	51

CHAPITRE 3

Positionnement et layout : les nouvelles techniques

de mise en page	53
Le modèle de boîte : retour aux sources avec <code>box-sizing</code>	54
Cas pratique : simplifier les calculs de dimensions	56
Compatibilité des navigateurs pour <code>box-sizing</code>	57
Le multicolonne	58
Comment ça marche ?	58
Syntaxe	59
<i>Largeur des colonnes avec <code>column-width</code></i>	59
<i>Nombre de colonnes avec <code>column-count</code>.</i>	60
<i>Déclaration raccourcie avec <code>columns</code></i>	61
<i>Gestion de la gouttière avec <code>column-gap</code>.</i>	61
<i>Séparation des colonnes avec <code>column-rule</code>.</i>	62
<i>Interruption des colonnes avec <code>break-*</code></i>	63
<i>Envahissement des colonnes avec <code>column-span</code>.</i>	64
<i>Équilibrage des colonnes avec <code>column-fill</code>.</i>	65
Cas pratique : alléger une liste chargée	66
Cas pratique : utiliser les colonnes comme grille	66
Compatibilité des navigateurs pour le multicolonne	69
Les modèles de boîtes flexibles avec Flexbox	70
Comment ça marche ?	70
Syntaxe	72
<i>Initialisation avec <code>display: flex inline-flex</code>.</i>	72
<i>Direction générale avec <code>flex-direction</code>.</i>	72

<i>Gestion des retours à la ligne avec flex-wrap</i>	73
<i>Déclaration raccourcie avec flex-flow</i>	74
<i>Justification avec justify-content</i>	74
<i>Alignement du contenu avec align-items</i>	75
<i>Répartition des lignes avec align-content</i>	76
<i>Gestion de l'ordre d'apparition avec order</i>	76
<i>Accroissement avec flex-grow</i>	77
<i>Rétrécissement avec flex-shrink</i>	77
<i>Dimensions par défaut avec flex-basis</i>	78
<i>Déclaration raccourcie avec flex</i>	78
<i>Alignement particulier avec align-self</i>	79
Cas pratique : un menu responsive	79
Cas pratique : un layout mobile first	81
Cas pratique : centrage absolu	83
Cas pratique : un formulaire fluide	84
Compatibilité des navigateurs pour Flexbox	85
<i>Mettre en place une solution pour tous les navigateurs</i>	86
<i>Un préprocesseur pour simplifier les préfixes (Sass)</i>	86
Le Grid Layout	87
Comment ça marche ?	88
Une introduction par l'exemple	89
Initialiser une grille	92
Définir la grille	93
<i>Simplifier les définitions avec la fonction repeat()</i>	94
<i>Fractions de l'espace restant avec l'unité fr</i>	94
Nommage	95
<i>Nommer les lignes</i>	95
<i>Nommer les zones</i>	96
Placer les éléments	97
<i>Le placement avec grid-row-start, grid-row-end, grid-column-start</i> <i>et grid-column-end</i>	98
<i>Le positionnement simplifié avec grid-row et grid-column</i>	101
<i>Le positionnement encore plus simplifié avec grid-area</i>	101
Placement automatique des éléments avec grid-auto-flow	102
Gestion des erreurs de placement	103
Le terme subgrid	103
Cas pratique : réaliser une galerie d'images	104
<i>Simplifier les choses avec un préprocesseur (Sass)</i>	107
Cas pratique : créer un layout simple	108
Compatibilité des navigateurs pour Grid Layout	111
La position « sticky »	111

Cas pratique : un header fixe	112
Compatibilité des navigateurs pour position: sticky	112
Une solution de repli en JavaScript	112
Les régions	113
Terminologie	113
Comment ça marche ?	114
Injecter du contenu dans un flux	114
Réclamer le contenu d'un flux	115
Gérer la fin d'un flux	115
Quelques informations complémentaires	116
<i>Boucles infinies</i>	116
<i>Arborescence HTML spécifique</i>	116
Cas pratique : une gestion des encarts publicitaires différente selon la taille de l'écran	117
Compatibilité des navigateurs pour les régions CSS	120
Les masques de forme en CSS	121
Comment ça marche ?	122
Zone de flottement	122
Formes de base	122
<i>circle</i>	122
<i>ellipse</i>	123
<i>inset</i>	123
<i>polygon</i>	124
Syntaxe	124
<i>Déclaration d'une forme avec shape-outside</i>	125
<i>Seuil d'opacité avec shape-image-threshold</i>	125
<i>Marge avec shape-margin</i>	125
Cas pratique : une bio avec un avatar circulaire	126
Compatibilité des navigateurs pour les masques de forme	127

CHAPITRE 4

Interfaces graphiques et amélioration visuelle 129

Des couleurs plus colorées	130
De nouvelles couleurs	130
La notation rgba	130
<i>Compatibilité des navigateurs pour rgba</i>	131
La notation hsla	131
<i>Compatibilité des navigateurs pour hsla</i>	133
L'opacité	133
opacity et l'héritage	134
opacity et les contextes d'empilement	135

Compatibilité des navigateurs pour opacity	135
Émuler le support sur Internet Explorer grâce aux filtres propriétaires de Microsoft	136
Des bords arrondis en CSS	137
La version simple	137
Dissocier les axes X et Y	139
Bordures et bords arrondis	142
Compatibilité des navigateurs pour border-radius	143
Des ombres avec box-shadow	144
À propos des performances	146
Cas pratique : donner du sens aux interfaces	147
Compatibilité des navigateurs pour box-shadow	149
Des ombres de texte avec text-shadow	149
Compatibilité des navigateurs pour text-shadow	151
Des dégradés	151
Syntaxe	151
<i>Dégradés linéaires</i>	152
<i>Dégradés radiaux</i>	153
Les dégradés au quotidien	155
Compatibilité des navigateurs pour les dégradés	155
Un meilleur contrôle des arrière-plans	157
Plusieurs arrière-plans sur un même élément	157
<i>Compatibilité des navigateurs pour les arrière-plans multiples</i>	158
De nouvelles valeurs pour background-repeat	159
<i>Cas pratique : placeholders dans une galerie d'images</i>	159
<i>Compatibilité des navigateurs pour background-repeat: space et background-repeat: round</i>	160
De nouvelles valeurs pour background-size	161
<i>Compatibilité des navigateurs pour background-size: cover et background-size: contain</i>	162
<i>Émuler le support sur Internet Explorer grâce aux filtres propriétaires de Microsoft</i>	162
Un nouveau système de valeurs pour background-position	163
<i>Compatibilité des navigateurs pour le nouveau système de background-position</i>	164
Origine d'arrière-plan avec background-origin	164
<i>Compatibilité des navigateurs pour background-origin</i>	165
Fixation de l'arrière-plan avec background-attachment: local	165
<i>Cas pratique : effet d'ombre sur un élément scrollable</i>	166
<i>Compatibilité des navigateurs pour background-attachment: local</i>	167
Zone d'arrière-plan avec background-clip	167

<i>Cas pratique : pop-up et bordure semi-transparente</i>	168
<i>Compatibilité des navigateurs pour background-clip</i>	169
Les filtres CSS	170
Comment fonctionnent-ils ?	170
Syntaxe	170
<i>blur</i>	171
<i>brightness</i>	172
<i>contrast</i>	172
<i>drop-shadow</i>	173
<i>grayscale</i>	174
<i>hue-rotate</i>	175
<i>invert</i>	175
<i>opacity</i>	176
<i>saturate</i>	176
<i>sepia</i>	177
Cas pratique : différents coloris d'image	177
Compatibilité des navigateurs pour les filtres CSS	179
Pointer-events	179
Cas pratique : améliorer les performances durant le scroll	180
Compatibilité des navigateurs pour les pointer-events	181
Des images comme bordures	181
Comment ça marche ?	181
Syntaxe	182
<i>border-image-source</i>	182
<i>border-image-slice</i>	182
<i>border-image-width</i>	183
<i>border-image-outset</i>	183
<i>border-image-repeat</i>	183
Compatibilité des navigateurs pour border-image	184

CHAPITRE 5

De nouvelles unités et valeurs 185

Le Saint-Graal des calculs : calc	186
À propos de la syntaxe	186
Pourquoi pas un préprocesseur ?	186
Cas pratique : utiliser calc pour le layout	187
Cas pratique : position de l'arrière-plan	187
Cas pratique : centrage absolu	188
Compatibilité des navigateurs pour calc	189
Root em : l'évolution de l'unité em	189
Pourquoi rem et pas simplement em ?	190

Cas pratique : une grille typographique	191
Compatibilité des navigateurs pour l'unité rem	193
<i>Une solution de repli avec un préprocesseur (Sass)</i>	193
Démystification : à propos de 62.5 %	194
Une unité pour la largeur d'un caractère : ch	196
Compatibilité des navigateurs pour l'unité ch	197
Les unités relatives au viewport	197
Pourcentage de la largeur/hauteur avec vw et vh	197
<i>Cas pratique : limiter la hauteur des images à la hauteur du viewport</i>	198
<i>Cas pratique : typographie responsive</i>	198
<i>Cas pratique : typographie responsive et rem</i>	200
<i>Cas pratique : conserver un ratio relatif au viewport</i>	200
<i>Compatibilité des navigateurs pour les unités vw et vh</i>	202
Pourcentage de la largeur/hauteur avec vmin et vmax	203
<i>Cas pratique : ratio 16:9 occupant toute la largeur du viewport</i>	203
<i>Compatibilité des navigateurs pour les unités vmin et vmax</i>	204
Les dimensions intrinsèques	204
Largeur minimale avec min-content	204
<i>Cas pratique : figure dimensionnée selon la largeur de l'image</i>	204
Largeur maximale avec max-content	206
Comportement de type block avec fill	206
Largeur optimale avec fit-content	207
<i>Cas pratique : liste centrée mais dimensionnée selon son contenu</i>	207
Compatibilité des navigateurs pour les dimensions intrinsèques	208
<i>Faciliter l'utilisation des valeurs intrinsèques avec un préprocesseur (Sass)</i> ...	208

CHAPITRE 6

Contrôle du texte **211**

La gestion des débordements avec overflow-wrap	212
Compatibilité des navigateurs pour overflow-wrap	212
La gestion des espaces avec white-space	213
Cas pratique : affichage de code	214
Compatibilité des navigateurs pour white-space	216
Les débordements de texte et text-overflow	217
Cas pratique : des lignes de tableau de même hauteur	218
Compatibilité des navigateurs pour text-overflow	221
Les césures avec hyphens	221
Syntaxe	222
<i>none</i>	222
<i>manual (valeur initiale)</i>	222
<i>auto</i>	223

Compatibilité des navigateurs pour hyphens	223
Les césures agressives avec word-break	224
Compatibilité des navigateurs pour word-break	225
La gestion des tabulations avec tab-size	225
Compatibilité des navigateurs pour tab-size	226
Une ponctuation plus élégante avec hanging-punctuation	226
Compatibilité des navigateurs pour hanging-punctuation	228
De meilleurs alignements avec text-align-last	228
Compatibilité des navigateurs pour text-align-last	229
La restriction de caractères avec unicode-range	229
Cas pratique : embellir les esperluettes	232
Compatibilité des navigateurs pour unicode-range	234
Cas pratique : des blocs de code qui donnent envie	235

CHAPITRE 7

Variables natives **237**

Comment ça marche ?	238
La syntaxe	238
Les variables invalides	239
Les valeurs de recours en cas d'invalidité	240
Cas particuliers et clarifications	241
Variables qui se référencent mutuellement	241
Propriété all et propriétés personnalisées	241
Animations	242
La compatibilité des navigateurs pour les variables natives	242

CHAPITRE 8

Styles conditionnels **243**

Feature Queries	243
De la notion de « support »	244
Syntaxe	245
API JavaScript	246
@supports et préfixes constructeurs	247
Cas pratique : header fixe à compter d'une certaine position avec position: sticky	248
Cas pratique : carrousel de Bootstrap 3	248
Compatibilité des navigateurs pour les Feature Queries	250
Media Queries	250
Syntaxe	251
<i>Conjonction</i>	251

<i>Condition</i>	252
<i>Négation</i>	252
<i>Restriction</i>	252
Que peut-on détecter ?	252
<i>Les dimensions : width et height</i>	252
<i>L'orientation de l'appareil</i>	253
<i>Le ratio hauteur/largeur de l'écran</i>	253
<i>La résolution de l'écran</i>	254
Futur : davantage de contrôle	255
<i>Présence de JavaScript</i>	255
<i>Luminosité ambiante</i>	256
<i>Système de pointage</i>	257
<i>Capacité de survol</i>	258
<i>Fréquence de mise à jour</i>	259
<i>Gestion des débordements</i>	260
Futur : Media Queries et variables	260
Cas pratique : le design Mobile First	261
Compatibilité des navigateurs pour les Media Queries	262

CHAPITRE 9

Transformations : un nouveau monde en 2D et en 3D..... 263

À quoi servent les transformations CSS ?	264
Les transformations 2D	265
Rotation	265
<i>Cas pratique : réaliser des flèches</i>	267
<i>Émulation de rotation avec les filtres propriétaires</i>	269
<i>Conversion d'angle avec un préprocesseur (Sass)</i>	269
Translation	270
<i>À propos des offsets</i>	271
<i>Cas pratique : centrage absolu</i>	271
Mise à l'échelle	272
<i>Cas pratique : agrandissement au survol</i>	273
<i>Émulation de mise à l'échelle avec zoom sur Internet Explorer 8</i>	273
<i>Bug d'Android 2.3</i>	274
Inclinaison	274
<i>Cas pratique : un menu incliné</i>	275
matrix	276
Compatibilité des navigateurs pour les transformations 2D	277
L'origine de transformation	277
Tester et déboguer transform-origin	278
L'ordre des transformations	279

Les transformations 3D	281
Compatibilité des navigateurs pour les transformations 3D	282
Environnement 3D avec perspective	282
Origine de perspective	284
En résumé	284
Contexte et transform-style	284
Notion de faces avec backface-visibility	285
Cas pratique : effet card flip	285

CHAPITRE 10

Animations et transitions : pour des interfaces

moins statiques **289**

À quoi servent les animations ?	290
Animation ou transition ?	291
À propos de l'accélération matérielle	292
<i>Compatibilité des navigateurs pour will-change</i>	293
JavaScript ou CSS ?	294
Les transitions	295
« Quoi ? » avec transition-property	296
« Comment ? » avec transition-timing-function	296
« Combien de temps ? » avec transition-duration	299
« Quand ? » avec transition-delay	300
Utilisation	301
Quelques informations complémentaires	301
Cas pratique : une sidebar plus légère	302
Cas pratique : changement de vue sur mobile	304
Cas pratique : animation d'une pop-up	307
Compatibilité des navigateurs pour les transitions	309
<i>Transitions et pseudo-éléments</i>	309
Les animations	310
« Quoi ? » avec animation-name	311
« Comment ? » avec animation-timing-function	311
« Combien de temps ? » avec animation-duration	312
« Combien de fois ? » avec animation-iteration-count	313
« Dans quel sens ? » avec animation-direction	313
« Quel état ? » avec animation-play-state	314
« Quand ? » avec animation-delay	315
« Quel impact ? » avec animation-fill-mode	315
La définition d'une animation : @keyframes	316
Animations et performance	317

Cas pratique : animation d'un loader	318
Cas pratique : animation d'un carrousel	319
Cas pratique : animation de l'affichage d'une galerie d'images	323
<i>Faciliter la génération des styles avec un préprocesseur (Sass)</i>	324
Cas pratique : animations comme feedbacks	324
Cas pratique : mise en place d'un système anti-spoiler	327
Compatibilité des navigateurs pour les animations	334

ANNEXE A

Liste des propriétés CSS et de leur valeur par défaut.....	335
---	------------

ANNEXE B

Liste des propriétés CSS qui peuvent être animées	341
--	------------

Notes	344
Ce que l'on aimerait pouvoir animer	344

ANNEXE C

Ressources et liens	345
----------------------------------	------------

Sites francophones	345
Sites anglophones	346
Newsletters	347
Comptes Twitter	347
Bibliographie	348

Index.....	349
-------------------	------------

Avant-propos

CSS, un langage en perpétuelle évolution

Au cours de cette dernière décennie, peu de langages peuvent se vanter d'avoir connu une révolution aussi exceptionnelle que celle vécue par CSS. Pour ceux d'entre vous qui réalisaient déjà des sites web au début des années 2000, vous n'êtes pas sans vous souvenir, peut-être avec nostalgie, que la mise en page en tableaux et les `spacer.gif` ne sont pas si lointains.

Aujourd'hui, tout cela est terminé (du moins je l'espère...). CSS est en effet le langage destiné à l'habillage des pages web, et cela n'est pas près de changer. D'autant plus qu'il sert aux sites web, aux applications mais aussi aux livres. Autrement dit, apprendre CSS n'est plus une option, c'est même une nécessité, tout du moins pour les personnes qui ont vocation à travailler dans le domaine du développement web.

Heureusement, les ressources ne se font pas aussi rares qu'il y a dix ans, y compris celles francophones. Aussi apprendre CSS, ou plutôt améliorer la qualité de ses feuilles de styles, est-il tout à fait envisageable, et j'espère que ce livre vous aidera dans cette tâche.

Pourquoi cet ouvrage ?

Si vous êtes un lecteur assidu des ouvrages des éditions Eyrolles, vous avez peut-être (ou avez eu) dans votre bibliothèque le livre intitulé *CSS 2 – Pratique du design web*, écrit par Raphaël Goetter. Sachez que l'ouvrage que vous avez présentement entre vos mains en est son digne successeur, du moins je l'espère !

La première édition de son aïeul date de juin 2005, soit il y a presque dix ans. Inutile de vous dire à quel point CSS a changé depuis. C'est d'ailleurs pour cette raison que Raphaël l'a mis à jour à trois reprises.

Seulement aujourd'hui, la mise à jour ne suffit plus. L'ouvrage initial étant trop vieux et pour l'essentiel dépassé, il a été décidé de le reprendre depuis ses fondations, afin de le mettre au goût du jour pour de bon et de présenter un livre dédié aux nouveautés dans le domaine des CSS.

Comme son fier ancêtre, cet ouvrage n'a pas de prétentions audacieuses ; il ne cherche qu'à apporter des éléments de réponse et des bases solides pour que vous soyez en mesure d'attaquer vos projets courants et futurs avec les technologies actuelles.

À qui s'adresse cet ouvrage ?

Quand nous avons lancé le projet d'écriture de ce livre, il a rapidement été question de déterminer sa cible. Or, comme beaucoup de langages, CSS couvre un spectre très large de niveaux de compétences. Il existe en effet des développeurs absolument inexpérimentés, des instruits, des novices, des confirmés, des experts, des légendes, et peut-être même que vous ne figurez dans aucune de ces catégories ! Autant dire qu'il est difficile de contenter tout le monde et, fort heureusement, ce n'est pas la prétention de cet ouvrage.

Nous allons aborder de nombreuses notions, certaines relativement simples comme les sélecteurs, d'autres beaucoup plus complexes comme les nouveaux modules de mise en page. Quoiqu'il en soit, nous ne reprendrons pas les bases, sans quoi ce livre ferait quelques milliers de pages !

Aussi dirais-je que cet ouvrage s'adresse à des développeurs qui ont déjà utilisé CSS à maintes reprises et qui sont en parfaite mesure de s'en sortir avec les bases qu'ils ont apprises, mais pour qui « s'en sortir » ne suffit plus ; en d'autres termes, ceux qui désirent aller plus loin. De fait, si vous êtes un fanatique de CSS, utilisateur de post/préprocesseurs, mordu de veille techno et rock star du DevTools, vous risquez de sortir de cet ouvrage tel que vous vous y êtes plongé, et possiblement un peu déçu.

Ce livre est destiné aux développeurs désireux d'apprendre et d'aller de l'avant, et non aux néophytes les plus complets, ni aux experts en la profession.

Idéalement, j'aimerais donc que vous terminiez la lecture de cet ouvrage en étant plus instruit, et pourquoi pas un brin inspiré !

La structure de l'ouvrage

Ce livre a pour vocation de faire un tour d'horizon des modules CSS émergents ; autant dire qu'il y a de quoi faire ! Du coup, il n'a pas été simple de définir un ordre logique à tout cela. Et pourtant, il y en a bien un !

Tout d'abord, Raphaël nous réglera d'une introduction sur l'état actuel des CSS, des standards et du marché des navigateurs. On y abordera la formidable épopée des préfixes constructeurs, l'arrivée astucieuse des *flags* et le chemin parsemé d'embûches qu'emprunte une fonctionnalité avant d'être standardisée.

Ensuite, nous débiterons en douceur avec un chapitre intégralement dédié aux nouvelles façons d'accéder au DOM (*Document Object Model*), c'est-à-dire aux éléments HTML. Effectivement, avant de pouvoir appliquer des styles aux éléments, il faut pouvoir les cibler !

Une fois que vous serez en pleine mesure de cibler n'importe quel élément dans le DOM grâce aux sélecteurs avancés, il sera temps de s'attaquer au plat de résistance avec les nouvelles techniques de mise en page : *Flexbox*, *Grid Layout*, multicolonne... Il y a beaucoup de choses à voir, et ce ne sont pas les plus simples.

La structure mise en place, on pourra alors se pencher sur l'habillage des documents. Il s'avère que CSS 3 apporte énormément de nouveautés en la matière : ombres, bords arrondis, dégradés, manipulation des arrière-plans...

Après quoi, nous irons faire un tour du côté des nouvelles unités et fonctions, qui peuvent s'avérer très pratiques en association avec les notions abordées dans les deux chapitres précédents.

L'un des aspects souvent bien méconnus des CSS concerne le niveau de contrôle que l'on peut avoir sur les contenus textuels. Il s'avère que nous avons à notre disposition de plus en plus d'outils pour s'approcher de la typographie « print », à défaut d'être en mesure de véritablement rivaliser. Ce sera justement le sujet de notre sixième chapitre.

Ensuite, j'ai souhaité que vous vous familiarisiez avec les variables natives qui sont tout ce qu'il y a d'imminentes. Vu qu'il s'agit d'un élément incontournable du futur de CSS, j'ai jugé bon de lui dédier un chapitre, somme toute assez court.

Le chapitre suivant ne sera sûrement pas une totale découverte pour vous puisqu'il traitera des styles conditionnels : les *Feature Queries* (via la directive `@supports`) et les fameuses *Media Queries*. Nous en profiterons même pour faire un tour d'horizon du futur de ces dernières et, qui sait, peut-être serez-vous surpris ?!

Pour la fin de l'ouvrage, j'ai décidé d'aborder des notions un peu plus originales, avec notamment les transformations CSS, y compris celles 3D, et la façon dont vous pouvez en tirer parti dans vos créations. Seront également traitées les animations et comment les utiliser à bon escient pour améliorer vos interfaces.

Dans les annexes, vous trouverez un certain nombre d'informations supplémentaires pour aller plus loin, notamment une bibliographie ou encore la liste des propriétés CSS pouvant faire l'objet d'une animation.



Compléments vidéo à consulter en ligne

Cet ouvrage a été conçu pour être aussi interactif que possible. C'est pourquoi, en plus du livre que vous tenez entre les mains, nous vous invitons à consulter des vidéos de formation associées que j'ai réalisées et accessibles à l'adresse suivante :

<http://www.editions-eyrolles.com/go/videoslivreCSS3>.

Remerciements

Avant tout, je souhaite remercier infiniment ma compagne, Alexandra, pour avoir fait preuve d'une patience à toute épreuve et d'un soutien sans faille au cours de ces longs mois d'écriture à l'humeur parfois difficile, et aussi pour ses belles photographies qui illustrent ce livre.

Un grand merci également aux personnes que j'ai sollicitées au cours de la rédaction de cet ouvrage pour des avis et des questions techniques : Rachel Andrew, Tab Atkins Jr., Dave DeSandro, Paul Irish, Paul Lewis, Rachel Nabors, Sara Soueidan et Ana Tudor, ainsi que Chris Coyier qui nous fait l'honneur de sa préface.

Bien évidemment, je souhaite remercier chaleureusement les relecteurs techniques Geoffrey Crofte et Thomas Zilliox, et surtout Raphaël Goetter qui a su m'accompagner et me faire part de son expérience au cours de cette aventure.

Enfin, merci à toute l'équipe des éditions Eyrolles, notamment Alexandre Habian et Muriel Shan Sei Fan qui ont su m'aiguiller dans la rédaction de ce livre, mais aussi tous les relecteurs qui sont parvenus à corriger mes tournures pas toujours heureuses, de sorte que ce livre soit agréable à lire.

1

L'état actuel du W3C et des standards CSS

Inventé au début des années 1990, CSS a connu une route longue et parsemée d'embûches pour arriver au langage de styles tel qu'on le connaît aujourd'hui. Débutons avec un état des lieux et une mise en situation.

Une évolution implacable

En 2005, lors de la sortie du livre *CSS 2 – Pratique du design web*, la conception de sites en CSS n'en était qu'à ses balbutiements et la mise en page en tableaux HTML était monnaie courante. Pour faire simple, « passer aux standards » demeurait à cette époque une mission pour le moins périlleuse.

En ces temps troubles, les *doctype* XHTML, les *framesets*, [spacer.gif](#) et Internet Explorer 6 avaient le vent en poupe : imaginez-vous dans un monde où Internet Explorer 7 et Google Chrome n'existaient pas encore ! Si, comme moi, vous avez connu cette époque de la « bidouille », vous ne pouvez que vous réjouir de sa disparition et constater le chemin parcouru en près de dix années.

Si je devais résumer la période web actuelle en seulement quatre mots-clés, je citerais :

- HTML5CSS3 (d'accord, j'ai triché !)
- Web mobile ;

- industrialisation ;
- stabilisation.

HTML 5 et CSS 3, véritables fleurons du Web moderne, apportent de nouvelles couches et améliorations considérables aux versions précédentes. Voici les principales innovations apportées par ces technologies :

- de multiples possibilités graphiques (arrondis, ombrages, opacité, dégradés, arrière-plans multiples, pour ne citer que les plus connus) ;
- de nouveaux types de positionnements (*Flexbox*, *Grid Layout*, multicolonne) ;
- de nombreuses façons de cibler les éléments ;
- des transformations (rotation, zoom, déformation, translation) ;
- enfin, des animations pour que les interfaces deviennent un peu moins figées.

Et je ne viens d'évoquer que des fonctionnalités CSS 3 ! En vérité, ce sont tous les domaines du Web qui ont bénéficié de spectaculaires avancées techniques : HTML 5, mais aussi WebGL, SVG, Canvas, etc.

En parallèle, cette décennie marque l'avènement d'un monde « mobile ». En 2007 en effet, l'iPhone d'Apple débarque et provoque un raz-de-marée : il est enfin possible (et pratique) de se servir d'un téléphone portable pour surfer sur le Net. Le smartphone est né.

Avec le Web mobile s'est ouvert un formidable nouveau marché à conquérir, avec son lot de combats de titans sans merci dans lesquels certains, tels que Google et Apple, en sont sortis grandis et où d'autres, tels que BlackBerry, Motorola et Nokia, y ont laissé des plumes. Ou pire...

Notre activité professionnelle se doit d'accompagner cette révolution mobile : « Responsive Web Design » a été élu mot-clé *buzz* de l'année 2013, il en découle que les notions de performance et de temps d'affichage deviennent essentielles et, enfin, une remise en question radicale de l'ergonomie d'un site web est nécessaire dans un monde connecté que l'on peut tenir dans sa main.

L'industrialisation apparaît elle aussi telle une évidence depuis plusieurs années. Le métier d'intégrateur s'est en effet radicalement transformé en peu de temps, agrégeant des contraintes jusqu'alors inédites. Aujourd'hui, il ne suffit plus de rédiger du code HTML et CSS valide, mais bien de réaliser de nombreuses tâches simultanément :

- produire du code lisible et maintenable dans la durée ;
- tester sur plusieurs périphériques et formats différents ;
- ajouter les préfixes nécessaires aux propriétés CSS 3 non finalisées ;
- versionner ses fichiers de travail ;
- gagner en performance en compressant les images ainsi que CSS et JavaScript ;
- ne pas réinventer la roue à chaque projet et automatiser le maximum de tâches.

Autant dire que les outils mis à notre disposition pour pallier les lacunes de CSS ou accélérer notre méthode de production sont très rapidement devenus une béquille indispensable à notre quotidien. Je pense notamment aux préprocesseurs, dont Sass est l'un des plus célèbres porte-parole, ou encore aux planificateurs de tâches (*task runners*) tels que Grunt ou Gulp. Et, chaque jour, on peut assister à la naissance (ou la mort) de l'un de ces outils.

La bonne nouvelle est que tout cela s'accompagne d'une relative mais générale stabilisation du côté des standards qui régissent le Web. La présence de nouveaux acteurs dans le monde des navigateurs web, notamment Chrome de Google né en 2008, ainsi que la fin de l'hégémonie d'Internet Explorer et la mort programmée de Windows XP favorisent en réalité l'action menée par des organismes mondiaux régulateurs tels que le W3C ou le WHATWG. Les standards, quoi.

D'une manière générale, nous pouvons observer que certains consensus s'établissent plus facilement qu'auparavant entre les forces en présence (par exemple, l'idée de laisser tomber les préfixes constructeurs au profit des *flags* – nous y reviendrons un peu plus loin).

Autre point extrêmement positif dans ce constat : le « dinosaure » Internet Explorer rattrape rapidement son retard et offre enfin la possibilité de concevoir des pages web sans bricolages farfelus et aléatoires. La preuve en est qu'Internet Explorer a désormais un *developer channel* public.

Le design web est dorénavant une affaire de règles HTML et CSS établies, qui s'appliquent à la grande majorité des navigateurs actuels.

Un tour d'horizon des navigateurs d'aujourd'hui

Dans les années 2000, la suprématie d'Internet Explorer était alors établie depuis longtemps et rien ne paraissait pouvoir la faire fléchir. À peine quelques années plus tard, Mozilla Firefox, promu par une communauté libre exceptionnelle, commençait déjà à faire de l'ombre au mastodonte de Microsoft.

Puis Google Chrome débarque en 2008. Aujourd'hui, il emporte tout sur son passage. Rapidement propulsé au troisième rang des navigateurs, il occupe actuellement la tête du classement sans que son hégémonie ne puisse être discutée.

Il ne reste plus guère de place pour les autres navigateurs « alternatifs », ce qui n'empêche pas certains méconnus de s'illustrer de belle manière, par exemple Maxthon (basé sur WebKit) qui détient le plus haut score au classement HTML5Test.

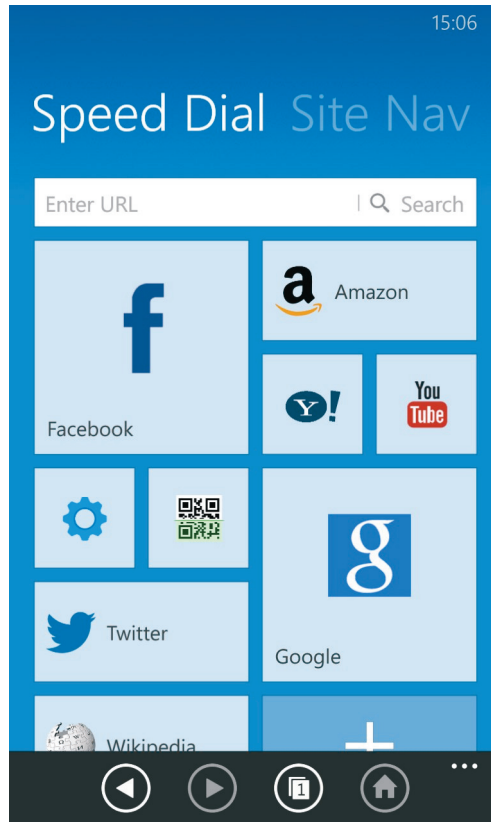
RESSOURCE HTML5Test

HTML5Test est un outil en ligne qui évalue la capacité d'un navigateur (le vôtre) à supporter les fonctionnalités HTML 5 et CSS 3. Il dispose également d'un classement des meilleurs navigateurs en termes de support.

► <http://bit.ly/html5-test>

Les navigateurs mobiles ont également bénéficié d'un large essor durant ces dernières années. Certains, à l'instar de Safari et Chrome (encore lui !), se sont taillé la part du lion, tandis que de nouveaux challengers, comme UC Browser, Silk d'Amazon, Dolphin ou encore Coast d'Opera, débarquent fraîchement dans l'arène.

Figure 1–1
UC Browser



L'état actuel des standards

Le W3C (*World Wide Web Consortium*) est l'organisation mondiale qui fait tourner les standards en coulisses depuis sa création en 1994. C'est à travers son impulsion que se sont construits HTML, XHTML, SVG, PNG et bien d'autres règles d'accessibilité numérique.

Cependant, ce qui fait à la fois la force et la faiblesse du W3C est que ses spécifications sont soumises à un accord entre l'ensemble de ses membres influents. Cela lui confère sa neutralité indispensable, mais c'est également une source de lenteurs pesantes.

Les enjeux économiques sont parfois tels que se mettre au diapason entre des acteurs comme Microsoft, Apple, Adobe ou Google nécessite des processus et des négociations très rigoureux, chronophages, voire carrément figés.

Critiquant ouvertement l'inertie du W3C ainsi que certains de ses choix stratégiques (celui de laisser mourir HTML au profit de XHTML, notamment), certains membres d'Apple, Mozilla et Opera ont décidé en 2004 de collaborer officiellement sur des spécifications qui leur paraissaient « implémentables » rapidement dans les navigateurs web.

Cette nouvelle organisation, le WHATWG (*Web Hypertext Application Technology Working Group*), est composée, entre autres, de membres actifs du W3C. Elle œuvre parallèlement à celui-ci sur des standards HTML 5, Web Workers, Microdata, Web Forms et applications web.

Les résultats du groupe de travail HTML 5 du WHATWG furent si prometteurs qu'ils ont été adoptés officiellement en 2007 par le W3C comme structure de base pour la future version d'HTML. Les deux organismes collaborent depuis sur ce standard, apportant chacun leur pierre à l'édifice.

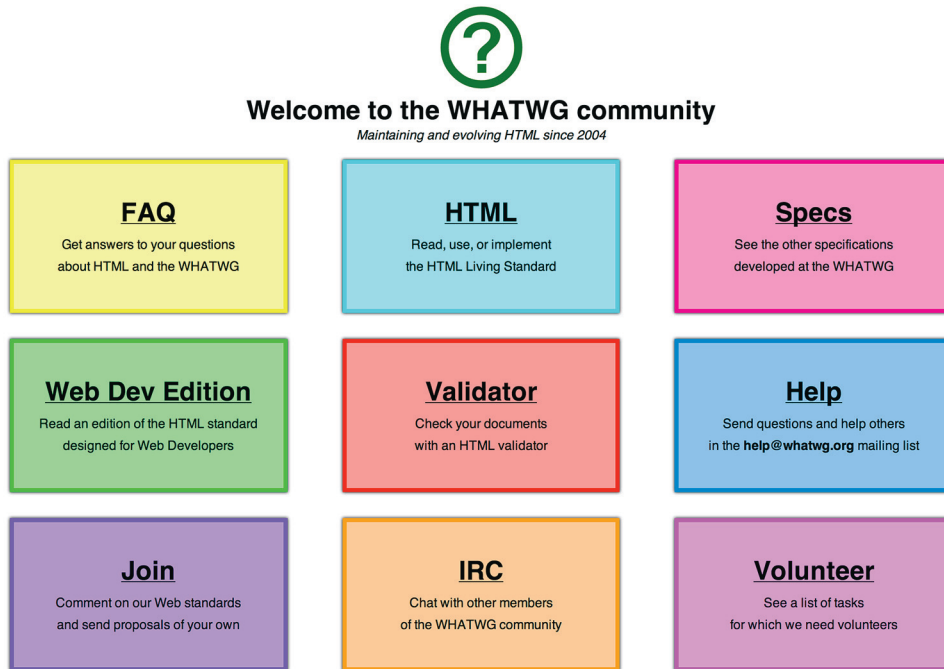


Figure 1–2 Site du WHATWG (<http://whatwg.org>)

L'aventure des préfixes constructeurs

Les désormais célèbres préfixes CSS (`-moz-`, `-webkit-`, `-ms-`, `-o-`, etc.) ont été élaborés par le W3C à l'époque des spécifications CSS 2.1, il y a donc plus de dix ans.

L'objectif était d'offrir aux constructeurs de navigateurs un moyen de tester des propriétés ou des valeurs CSS encore non finalisées et de les proposer à leurs utilisateurs, ce qui semblait être une bénédiction au vu de la lenteur du processus interne de standardisation du W3C.

Chaque navigateur dispose par conséquent de sa propre variante de propriété et peut l'implémenter à sa guise à condition qu'elle soit préfixée comme il se doit. Ainsi `-moz-animation` pourrait en théorie différer complètement de la propriété `animation` officiellement encore en

brouillon. Il s'agit de l'interprétation de la propriété `animation` par le moteur de rendu de Mozilla (Gecko).

Avec le recul, il est indubitable que l'existence des préfixes CSS a rétrospectivement fait plus de mal que de bien, et ce, pour de multiples raisons.

Tout d'abord, les développeurs se sont rués sur cet eldorado sans forcément en comprendre les implications, usant et abusant de propriétés non finalisées sans même le savoir.

Par ailleurs, une tendance globale vise à ne préfixer que pour les navigateurs les plus en vogue et les plus représentatifs. Par exemple, une multitude d'articles et tutoriels exposent des exemples de code où les propriétés ne ciblent que `-webkit-` (notamment sur mobile où ce moteur de rendu est roi), laissant en plan tous les autres navigateurs, soit par oubli, soit par pure fainéantise. À un tel point qu'Opera s'est vu contraint de devoir reconnaître lui aussi le préfixe `-webkit-` pour ne pas être considéré comme un « mauvais navigateur » (le navigateur a par la suite abandonné le moteur de rendu Presto pour joindre Chrome et son moteur Blink) !

Pour éviter de telles dérives mais aussi une maintenabilité complexe, un ensemble de constructeurs a décidé d'abandonner petit à petit les préfixes CSS au profit d'une autre solution plus robuste selon eux : la possibilité d'activer – ou non – certaines fonctionnalités directement au sein du navigateur (via ce que l'on appelle les « flags »). Cela signifie que si vous souhaitez tester un module tel que les Régions CSS (que nous étudierons plus en détail dans ce livre), vous devez l'indiquer à votre navigateur et non plus dans une surcharge de feuille de styles destinée à d'autres usagers.

L'activation de flags est une démarche pour le moins pertinente. En effet, dans l'idéal, ce n'est pas au développeur de tester les propriétés en brouillon, mais bien au navigateur. Cette opération devient alors totalement transparente pour l'utilisateur, mais réduit la portée des toutes nouvelles fonctionnalités. En effet, il n'est dorénavant plus envisageable de « forcer » vos visiteurs à les reconnaître (car vous n'allez pas demander à tous vos clients d'aller activer un flag dans la configuration de leur navigateur).

Nous ne savons pas encore ce que l'avenir nous réserve dans ce vaste et complexe monde de la standardisation : aujourd'hui, les préfixes n'ont pas complètement disparu et les flags navigateurs ne sont encore exploités qu'avec parcimonie.

Une tierce solution vient d'ailleurs de voir le jour récemment. Elle est basée sur la règle conditionnelle `@supports` (que l'on peut comparer au fameux `Modernizr`), permettant de savoir si une propriété ou une valeur est implémentée par un navigateur. Nous reviendrons bien évidemment sur cette fonctionnalité dans le chapitre 8 dédié aux styles conditionnels.



Careful, these experiments may bite

WARNING These experimental features may change, break, or disappear at any time. We make absolutely no guarantees about what may happen if you turn one of these experiments on, and your browser may even spontaneously combust. Jokes aside, your browser may delete all your data, or your security and privacy could be compromised in unexpected ways. Any experiments you enable will be enabled for all users of this browser. Please proceed with caution. Interested in cool new Chrome features? Try our beta channel at chrome.com/beta.

Reset all to default

Override software rendering list Mac, Windows, Linux, Chrome OS, Android <small>Overrides the built-in software rendering list and enables GPU-acceleration on unsupported system configurations. #ignore-gpu-blacklist</small> Enable
Accelerated overflow scroll Mac, Windows, Linux, Chrome OS, Android <small>When possible, puts the scrolling contents of an overflow scrolling element onto a composited layer for faster scrolling. #force-accelerated-composited-scrolling</small> <small>(Default: on)</small>
Universal accelerated overflow scroll Mac, Windows, Linux, Chrome OS, Android <small>Puts scrolling content in composited layers, even in those cases where promoting the overflow scrolling element to a stacking context and a containing block would have broken stacking or clipping. #force-universal-accelerated-composited-scrolling</small> <small>(Default: off)</small>
Disable layer squashing Mac, Windows, Linux, Chrome OS, Android <small>Prevents the automatic combining of composited layers. #disable_layer_squashing</small> Enable
Enable experimental canvas features Mac, Windows, Linux, Chrome OS, Android <small>Enables the use of experimental canvas features which are still in development. #enable-experimental-canvas-features</small> Enable
Disable accelerated 2D canvas Mac, Windows, Linux, Chrome OS, Android <small>Disables the use of the GPU to perform 2d canvas rendering and instead uses software rendering. #disable-accelerated-2d-canvas</small> Enable
Composited render layer borders Mac, Windows, Linux, Chrome OS, Android <small>Forces a border around composited Render Layers to help debug and study layer compositing. #composited-layer-borders</small> Enable
FPS counter Mac, Windows, Linux, Chrome OS, Android <small>Shows a page's actual frame rate, in frames per second, when hardware acceleration is active. #show-fps-counter</small> Enable
Disable WebGL Mac, Windows, Linux, Chrome OS, Android <small>Enabling this option prevents web applications from accessing the WebGL API. #disable-webgl</small> Enable
Disable WebRTC device enumeration Mac, Windows, Linux, Chrome OS, Android <small>Disable support for MediaStreamTrack.getSources(). #disable-device-enumeration</small> Enable
Fixed position elements create stacking contexts Mac, Windows, Linux, Chrome OS, Android <small>Enabling this option makes all fixed position elements create new CSS stacking contexts. #fixed-position-creates-stacking-context</small> <small>(Default: off)</small>
Compositing for fixed position elements Mac, Windows, Linux, Chrome OS, Android <small>Enabling this option will make fixed position elements have their own composited layers. Note that fixed position elements must also create stacking contexts for this to work. #enable-compositing-for-fixed-position</small> <small>(Default: off)</small>
Compositing for RenderLayers with transitions Mac, Windows, Linux, Chrome OS, Android <small>Enabling this option will make RenderLayers with a transition on opacity, transform, or filter have their own composited layer. #enable-compositing-for-transition</small> <small>(Default: off)</small>

Figure 1–3 Page dédiée aux flags dans Google Chrome (<chrome://flags>)

La standardisation des CSS

Je vais vous avouer quelque chose : le titre de ce livre est quelque peu mensonger. En réalité, CSS 3 n'existe pas, et CSS 4 non plus. Il s'agit là d'un abus de langage qui caractérise « tout ce qui vient après CSS 2.1 ».

Voyez-vous, CSS 2.1 est la dernière version monolithique du langage. À compter de celle-ci, les organismes chargés des spécifications CSS se sont accordés sur le fait qu'il devenait très difficile et très long de produire et de maintenir des versions de cette ampleur.

Aussi a-t-on décrété que CSS serait désormais découpé en modules indépendants et susceptibles d'avancer à leur propre vitesse. Beaucoup de modules ont démarré au niveau 3 parce qu'ils étaient dans la continuité des fonctionnalités présentes dans CSS 2.1. Mais les modules émergents tels que Flexbox débutent bien évidemment au niveau 1 alors que d'autres modules sont déjà au niveau 4 (notamment celui des sélecteurs).

En somme, il est temps d'abandonner la notion de version de CSS. CSS 2.1 est dépassé, et CSS 3 n'existe pas. Je suggère simplement d'utiliser CSS, en toute simplicité.

À LIRE **There is no such thing as CSS 4**

L'article « There is no such thing as CSS 4 » par Tab Atkins Jr., éditeur principal des spécifications CSS, explique très bien les raisons pour lesquelles le CSSWG (*CSS Working Group*) a décidé de livrer CSS en modules indépendants.

▶ <http://bit.ly/css-4>

Mais comment un module change-t-il de niveau ? Pour répondre à cette question, il faut regarder du côté de son processus de standardisation, qui est long et parsemé d'embûches. Et pour cause, celui-ci passe par cinq étapes majeures.

Étape 1. Trouver une idée

La toute première étape, qui finalement n'en est pas vraiment une, est de trouver une idée qui tient la route. Celle-ci peut émaner de n'importe qui : un fabricant de navigateurs, une compagnie, vous, moi, etc. Elle est ensuite soumise au CSSWG qui la débat pour arriver à un des aboutissements suivants.

- L'idée est approuvée.
- L'idée est rejetée.
- L'idée est oubliée. Oui, ça arrive...

Si elle est acceptée, elle entre dans la première véritable phase.

Étape 2. Editor's Draft (ED)

Une fois l'idée jugée intéressante, le groupe de travail planche dessus jusqu'à avoir un premier brouillon de spécification : le « brouillon de l'éditeur ». À ce niveau, tout peut arriver, y compris l'abandon. Dans le meilleur des cas, si le groupe de travail décide que l'idée tient la route, un *First Public Working Draft* (FPWD) est publié.

Étape 3. Working Draft (WD)

Tout n'est pas encore joué pour le module, car il peut toujours être abandonné. En effet, si après les différentes itérations sur le brouillon, les acteurs impliqués (internes comme externes) le jugent techniquement trop compliqué ou incomplet, le travail s'arrête ici.

Si, à l'inverse, tout se passe bien, le module finit par atteindre un stade de *Last Call Working Draft* (LCWD), autrement dit un appel aux derniers changements avant que la spécification ne passe en *Candidate Recommendation* (CR).

Étape 4. Candidate Recommendation (CR)

À partir du moment où une spécification passe en *Candidate Recommendation*, vous pouvez commencer à vous pencher dessus sérieusement, car les navigateurs sont susceptibles

d'entamer les premières implémentations. Et pour cause, ceux-ci – entre autres – sont chargés de tester la fonctionnalité de fond en comble. Une fois que le groupe de travail est en mesure de justifier de la robustesse de la spécification grâce à deux implémentations fonctionnelles solides (de la part de deux moteurs de rendu différents), le module est en passe de franchir la dernière étape.

Étape 5. Recommandation (REC)

Curieusement, cette étape n'est pas aussi rose que l'on pourrait bien le croire. En théorie, un module en REC signifie qu'il est validé, approuvé et très certainement implémenté.

Cependant, en pratique, une spécification REC est « morte », car elle n'est plus maintenue (dans la mesure où il est extrêmement difficile de mettre à jour un module en REC) et aussi parce qu'une nouvelle version du module est très probablement en préparation.

POUR EN SAVOIR PLUS

Pour plus d'informations sur le processus de standardisation des spécifications, je ne peux que vous suggérer l'article de Sabastian Ekström, « The CSS Standards Process », publié sur le blog CSS-Tricks, relu et approuvé par Tab Atkins Jr., principal auteur des spécifications CSS.

► <http://bit.ly/css-standards-process>