

# HTML 5

**Une référence  
pour le développeur web**

**Rodolphe Rimelé**

Préface de Raphaël Goetter

© Groupe Eyrolles, 2011, ISBN : 978-2-212-12982-3

**EYROLLES**



# Préface

Nous connaissons tous HTML plus ou moins intimement. Lorsque nous naviguons sur Internet ou que nous écrivons nos pages web, il reste fidèle à nos côtés, malgré les péripéties et déboires qu'il a subis. C'est qu'il en a connu des aventures ! Les derniers arrivés tels que Adobe Flash ou Microsoft Silverlight pensaient l'avoir enterré, en vain... On le retrouvait parfois sous les doux sobriquets de « DHTML » ou de « Web 2.0 ». Créé à l'origine pour tisser des liens et véhiculer du contenu de manière universelle, il se voyait dénaturé dans ses fonctions primaires, comme l'ont montré ses tableaux de mise en page, aujourd'hui diabolisés. À dire vrai et en y regardant de près, HTML a toujours été là, endossant loyalement son rôle d'ossature indispensable du Web.

À ses débuts, sa mission se limitait à structurer des contenus basiques, essentiellement textuels et scientifiques, ordonnés de façon rigoureusement linéaire et assez peu excitante, il faut l'avouer. Puis, au fur et à mesure de l'évolution des usages des internautes, HTML se diversifie, se renforce et s'adapte. Parfois avec un peu de retard sur les besoins, et malgré les batailles que se livrent les navigateurs.

La quatrième mouture de HTML, finalisée en 1998 – la préhistoire du Web, peut-on dire –, paraît bien désuète aujourd'hui, treize années plus tard. Il s'agit pourtant de la plus récente version finalisée ayant reçu l'adoubement officiel du W3C. Ce gigantesque retard accumulé durant ces dix dernières années tend enfin à se combler, petit à petit, grâce au développement et à l'implantation de la version 5 du langage, tant attendue. Aux bons et loyaux services de HTML 4, se substituent de riches univers adaptés aux besoins et usages d'un Web moderne, varié, rapide et mobile.

Nouvelles balises, nouvelles technologies, formulaires avancés, outils multimédias, adaptation aux supports nomades et applications performantes : autant de nouveaux mondes offerts par ce nouvel opus en voie d'adoption. Même si HTML 5, sorte de nouveau « Web 2.0 », est censé annoncer le Web de l'avenir, dans la pratique, ses fonctionnalités répondent tout simplement aux attentes des internautes. L'utilisateur

peut enfin profiter de la lecture audio ou vidéo sans plug-in additionnel, trouver un hôtel ou un emploi proches grâce à la géolocalisation (qui pose certes quelques questions de privauté), bénéficier de support hors-ligne lorsque sa connexion est défaillante ou qu'il se déplace, profiter de fonctions de glisser-déposer, de stockage intelligent ou des web-workers, ces travailleurs de l'ombre qui permettent d'accélérer les performances en parallélisant le traitement des ressources.

Pensez que la version précédente était finalisée depuis plus de dix ans ! La patience de nous autres concepteurs web avait atteint ses limites ; elle se voit enfin récompensée. De grands sites tels que Google, Yahoo!, Twitter, Myspace, Kelkoo, Youtube ou encore Dailymotion intègrent d'ores et déjà une multitude d'applications stables de HTML 5, mais les grands bénéficiaires en sont bien sûr les sites web pour périphériques mobiles qui peuvent déjà exploiter nativement des fonctionnalités telles que l'adaptation automatique des designs aux différentes tailles d'écran.

HTML a bien mûri. La version 5 s'annonce comme une ressource exaltante qui exploitera véritablement les possibilités technologiques contemporaines. Cela inclut la puissance des connexions internet, les périphériques mobiles (smartphones et tablettes) ainsi que le multimédia. Il était temps. Et il est l'heure à présent de révolutionner nos habitudes d'internautes et de concepteurs de sites web.

Cette révolution n'est d'ailleurs pas sans rappeler celle opérée dans mon domaine de prédilection, celui des feuilles de styles CSS, autre univers en refonte depuis la version CSS 3. En tant que designer web et pour avoir rédigé quelques ouvrages sur les feuilles de style CSS, je peux comprendre et partager l'enthousiasme de Rodolphe ainsi que celui de la communauté de développeurs en pleine ébullition.

Si la symbiose entre HTML et CSS semblait parfaite sur le papier – l'un s'attachant à structurer l'information (balises, sémantique), l'autre à lui donner forme (esthétique, positionnements) – je ne vous apprendrai pas que les deux ont longtemps été mal imbriqués : l'on côtoyait fréquemment les styles de mise en forme au cœur des éléments et balises HTML, bien que ce ne fût pas leur place attitrée et que cela nuisît à l'accessibilité et à la compatibilité des documents produits.

À l'ère de HTML 5 et CSS 3, le couple accède à une nouvelle dimension et de nouveaux pouvoirs. Les interactions entre les deux langages n'ont jamais été aussi puissantes : séparation fond-forme renforcée grâce à une pluralité de nouvelles balises sémantiques (`<header>`, `<footer>`, `<article>`, `<section>`, etc.), gestion des médias et périphériques mobiles, prise en charge démultipliée des formulaires (via la notion de formulaire valide, invalide, incomplet), etc.

Cette imbrication va au-delà d'une simple association de langages, une véritable philosophie de conception s'en dégage : par « HTML 5 », on entend désormais « HTML 5

combiné à CSS 3 et JavaScript ». Une avancée considérable pour le Web qui comble enfin de façon extraordinaire les attentes des designers et intégrateurs CSS.

Même le dernier retardataire, Microsoft, suit le mouvement avec entrain voire zèle – fait d’autant plus notable qu’il n’était pas coutumier du fait. En témoignent les premières moutures de Windows 8 et d’Internet Explorer 10, que nous avons pu voir en avant-première.

Le mouvement est en marche, la révolution ne fait que commencer...

Mais revenons à ce livre. HTML 5 se compose en pratique des langages HTML + CSS + JavaScript, autant de domaines dans lesquels l’auteur excelle.

Tout d’abord, cet ouvrage approche l’exhaustivité, compte-tenu des spécifications en cours d’évolution. Meticuleux, Rodolphe n’a pu se restreindre à moins de 600 pages de contenus, codes et illustrations. L’annexe en ligne sur l’accessibilité, notamment, mériterait d’être lue par tout professionnel du Web.

Ensuite, il ne contient pas d’approximation, et l’auteur ne prend aucune liberté avec les standards. Chaque partie est testée, moult fois vérifiée et validée avant de figurer dans l’ouvrage.

Enfin, il est agréable à lire, parsemé d’un humour que l’on pourrait qualifier de « à la Rodolphe »<sup>TM</sup> et qu’il manie avec beaucoup de justesse.

Ce qui m’amène à dire un mot de l’auteur : Rodolphe Rimelé manie avec une désinvolture naturelle les logiciels de graphisme et d’image, les animations Flash, il connaît les arcanes de langages tels que jQuery, Ajax, PHP, MySQL, l’administration de serveurs web et maîtrise encore bien d’autres jargons informatiques. Son *curriculum vitae* déborde allègrement de références en webdesign et développement, et se distingue par la publication d’un DVD d’apprentissage sur jQuery, du fameux lecteur Flash estampillé « Dewplayer » et d’un mémento sur MySQL précédemment publié chez Eyrolles. Ce portrait ne serait pas complet sans évoquer ses évidentes qualités de photographe amateur et d’humoriste tourmenté via son carnet personnel *blup.fr*. Tant de perfection et de diversité à la fois suscitent l’envie pour le commun des mortels que je suis. Dès que Rodolphe touche un clavier d’ordinateur, il semble que tout lui réussit, infailliblement, et à merveille. Et le voilà à présent qui s’attaque à HTML 5 !

Je suis sûr que vous trouverez bien d’autres qualités à son livre.

Bonne lecture.

Raphaël Goetter, fondateur d’Alsacreations.com  
raphael@goetter.fr

# Avant-propos

Ce livre est le fruit de longs mois de travail, de recherches, d'expérimentations, de dissections des spécifications toujours en mouvement, de pittoresques rencontres de bogues, de discussions avec mes collègues. J'espère qu'il sera utile au plus grand nombre de lecteurs et qu'il répantera autour de lui autant de plaisir que j'ai eu à l'écrire.

## Avertissement et conventions

La spécification HTML est un document qui dépasse les 800 pages, et qui fait référence à de nombreux autres grâce à la magie des liens hypertextes. Il est écrit dans un langage peu distrayant, condensé, très technique, à la destination d'un public très spécialisé.

Cet ouvrage se veut didactique et pratique. Il ne couvre pas toutes les mentions, exceptions, et précisions de la spécification, qui nécessiteraient 23 volumes différents et une nouvelle étagère à côté de votre bureau. C'est pourquoi son objectif est d'aller à l'essentiel.

J'ai choisi de faire référence aux éléments HTML en les notant principalement en tant que balises, c'est-à-dire entre les caractères *inférieur* à « < » et *supérieur* à « > », pour faciliter la lecture et l'appropriation du code.

Voici par exemple un élément `<p>` qui correspond à un paragraphe. Fondamentalement, un élément comprend une balise ouvrante `<p>` et une balise fermante `</p>`, éventuellement des attributs et du contenu. Il s'agit aussi d'une entité mixte qui est à la fois présente dans le code HTML, dans le DOM, et à laquelle on peut faire référence en JavaScript et en CSS. On peut donc la décrire de multiples façons selon le contexte. J'espère que les puristes me pardonneront ce trope.

Enfin, cet ouvrage ne traite pas de web design en général, ni de graphisme, ni des feuilles de style CSS (à l'exception des nouveautés CSS 3 mentionnées au sein des chapitres).

#### CONVENTION HTML 5 ou HTML5 ?

Les spécifications officielles prônent l'absence d'espace entre le numéro de version et le nom du langage. Nous avons préféré pour l'heure conserver l'espace encore quelque temps, pour plusieurs raisons. La vérité est que HTML 5 ne serait pas là, s'il n'y avait d'abord eu HTML 4, si essentielle soit leur différence. D'autre part, il nous semblait prématuré de conférer d'emblée à HTML 5 le privilège réservé aux seuls noms propres ou marques. Voyons encore en HTML 5 le résultat d'une démarche rationnelle qui a son histoire.

## Codes source

Les exemples de code sont rédigés en français avec quelques soupçons d'anglais, langage de prédilection pour le Web. Ils doivent être adaptés à vos besoins qui peuvent varier grandement selon le contexte de développement. Certains exemples figurant dans un chapitre ne reprendront pas nécessairement toutes les recommandations présentes dans les exemples précédents par souci de simplicité et de lisibilité. Il vous appartiendra de faire vos choix, au regard des tenants et aboutissants de vos projets.

Le code complet de la page HTML (en-tête et pied de page, propriétés de style) ne sera, la plupart du temps, pas présent, car semblable d'une page à l'autre, mais vous le retrouverez dans les chapitres initiaux et les fichiers à télécharger. Il en sera de même pour les feuilles de style CSS associées à la présentation des documents HTML. Les codes source affichent une fraîcheur maximum au moment de la rédaction.

#### COMPLÉMENTS EN LIGNE Annexes en ligne sur CSS et l'accessibilité (ARIA) et tableaux de prise en charge mis à jour

Vous trouverez sur le site d'accompagnement du livre des ressources et compléments, notamment trois annexes indispensables, mises à disposition gratuitement :

- l'**annexe A**, qui fait un point, au moment de la mise sous presse de l'ouvrage, sur les « **Fonctionnalités modifiées et obsolètes** » entre HTML 4 et HTML 5 ;
- l'**annexe B**, qui donne un précieux rappel sur les « **Feuilles de style CSS** » ;
- l'**annexe C**, enfin, qu'il faudrait mettre entre les mains de tout développeur web, et qui vous guide dans la création de sites accessibles, conformes à la spécification ARIA : « **Accessibilité et ARIA** ».

Vous trouverez également en ligne l'index de l'ouvrage mais aussi, sur le site de l'auteur, la mise à jour des tableaux de prise en charge par les navigateurs.

- ▶ <http://html5.blup.fr/>
- ▶ <http://www.editions-eyrolles.com/>

Quant aux illustrations utilisées dans cet ouvrage, elles sont issues de Fotolia avec leur permission. Les photos des démonstrations, des exemples et des captures d'écran ont été réalisées par l'auteur.

- ▶ <http://www.fotolia.com>

## À propos de l'auteur

Initialement ingénieur *réseaux et télécoms*, j'ai bifurqué quelque peu à l'issue de mes études vers ma passion réelle, qui avait débuté bien avant cela lorsque j'avais tenté de percer plusieurs mystères :

- celui de mon modem RTC qui produisait des bruits étranges ;
- celui de mon modem câble qui `<blink>`clignotait`</blink>` bien souvent orange ;
- celui de la première version de Flash qui m'a valu un stage expérimental auprès de mon propre fournisseur d'accès ;
- celui de toutes les invocations magiques telles qu'IRC, NTTP et IPX qui cachaient un Nouveau Monde désormais en péril.

À cette époque – mode nostalgie *on* – sobre en débit, mais pas en GIF animés, les géants du Web Google, Dailymotion, YouTube, Wikipédia, Facebook n'existaient pas. Point de Chatroulette, d'iPhone ou de Twitter à l'horizon. Les CMS se comp-taient sur les doigts d'une seule main. Cela n'en faisait pas pour autant un monde moins fabuleux – mode nostalgie *off*.

Comme beaucoup d'autres passionnés, je dois tout ce que j'ai appris à la veille techno-logique quotidienne que permet Internet, les échanges sur les forums, la lecture de livres en français ou en anglais, la vie en agence web, la création de projets expérimentaux, les recherches effectuées pour écrire des articles publiés sur Alsacréations.com.

Bien qu'intéressé par mes études menées à leur terme, je me suis aperçu que la confi-guration de routeurs (et d'autres protocoles qui gravitaient dans chaque couche du modèle OSI) était bien moins réjouissante que les séances de `<body>``</body>`, c'est-à-dire la réalisation de sites esthétiques et vraiment utiles à mes semblables. Par ailleurs, je pouvais apprendre en une journée ce que l'on nous inculquait à l'université en un mois et qui était par moments dépassé ou superfétatoire – n'entendez pas là qu'il y avait de *super fêtes*, mais que les applications concrètes se faisaient désirer.

Parmi mes compétences figurent les langages du Web JavaScript, CSS, et bien heu-reusement HTML ; mais aussi PHP, MySQL, Perl, C et tout ce qui concerne l'administration de systèmes et serveurs Linux. Au-delà du code, je manie également le graphisme avec Flash, Photoshop, Illustrator, Inkscape pour concevoir de belles interfaces ergonomiques et de rares Lolcats.

Je m'intéresse aussi à la géographie, l'histoire, l'astronomie, pour me changer les idées depuis que le rayonnement d'un écran illumine mon quotidien.

J'ai publié précédemment à cet ouvrage un Mémento MySQL aux éditions Eyrolles et un DVD d'apprentissage pour le langage jQuery/Ajax.

## À propos d'Alsacréations

Alsacreations.com, site communautaire d'apprentissage des standards web, a été fondé par Raphaël Goetter en 2003. Il a vu sa popularité grandir d'année en année avec la publication de nombreux tutoriels qui défendaient – parfois en avance sur leur temps – une conception web propre et conforme aux normes.

Pour participer à ce mouvement, et après avoir sympathisé autour d'une tarte flambée avec Raphaël, j'ai rédigé des articles, des actualités, développé le forum et d'autres fonctionnalités pour la communauté.

C'est avec plaisir que nous nous sommes associés en 2006 pour lancer Alsacreations.fr, l'agence web, afin de mettre en pratique ce que nous défendions et de pouvoir en faire notre métier. Nous nous attachons à ne pas utiliser des termes comme « Web 2.0 », « rationalisation de process » et « buzz viral ».

Notre petite équipe s'est étoffée progressivement mais sûrement, avec des collaborateurs de qualité en qui je place toute ma confiance. Aujourd'hui, je conduis et développe avec eux des projets pour des clients au niveau national et international.

## Remerciements

Cet ouvrage n'aurait pu voir le jour sans le soutien de tous ceux qui m'entourent au quotidien. Je remercie tout particulièrement...

Mes parents, pour tout ce qu'ils m'ont apporté durant ma jeunesse et mes études, pour tous les recoins du monde qu'ils m'ont fait découvrir : mon père qui nous a quittés durant l'écriture et à qui je dois beaucoup ; ma mère qui a toujours été présente et attentionnée pour moi.

Sarah, pour sa patience de tous les jours, ses relectures, ses muffins délicieux, et la joie de vivre qu'elle insuffle à notre couple.

Raphaël Goetter, pour ses conseils avisés, ses jeux de mots inédits et sa confiance sereine dans l'aventure de notre agence web.

Simon Kern, Philippe Vayssière, et Geoffrey Crofte, pour leur professionnalisme et la réjouissante ambiance qu'ils instaurent dans notre équipe, même après de longues journées.

Christophe Mattera, pour son « coup de pouce » à l'époque où je n'étais qu'un *padawan*.

# Table des matières

## CHAPITRE 1

### Une brève histoire du Web

#### et de ses standards ..... 1

Un successeur pour HTML 4 et XHTML 2

Le rôle du W3C ..... 8

    Une maturation rigoureuse... ..... 8

    ... mais peu véloce ..... 9

Le rôle du WhatWG ..... 10

Les fondements de l'évolution ..... 11

    En quoi consiste réellement HTML 5 ? ... 12

    Différences depuis HTML 4.01

    et XHTML 1.x ..... 13

HTML 5 = HTML + JavaScript + CSS (3) ? .. 14

    Pourquoi des standards pour le Web ? .. 15

## CHAPITRE 2

### HTML en seconde langue ..... 17

La syntaxe HTML 5 ..... 19

    Rappel sur les balises ..... 19

    Imbrications et types de contenu ..... 20

    Structure générale d'un document HTML . 22

    Attributs ..... 23

    Les commentaires ..... 24

L'encodage des caractères ..... 25

Le type MIME ..... 26

Comment le navigateur détermine-t-il

l'encodage des caractères et le type MIME ? . 27

HTML 5 ou XHTML 5 ? ..... 28

    Forme HTML ..... 28

    Forme XHTML ..... 28

Ce que vous savez sur XHTML

est probablement faux ..... 29

Du vrai XHTML 5 ..... 29

Les bons outils ..... 32

    Pour éditer ..... 32

    Pour tester et déboguer ..... 33

*Virtualisez !* ..... 34

*Mozilla Firefox* ..... 34

*Google Chrome* ..... 35

*Safari* ..... 36

*Opera* ..... 36

*Internet Explorer* ..... 37

    Consoles JavaScript pour les API HTML 5 . 37

La validation ..... 38

Rappels sur les styles CSS ..... 39

Rappels sur JavaScript ..... 42

    Frameworks JavaScript ..... 43

    Où placer `<script>` ? ..... 43

Publier un site en ligne ..... 44

    Choisir un hébergeur web ..... 44

    Utiliser un client FTP ..... 45

    Choisir un langage serveur

    et un système de gestion de contenu .. 45

Le protocole HTTP ..... 47

    Requêtes et en-têtes ..... 48

Bonnes pratiques ..... 51

    Organisation du code ..... 51

    Organisation des fichiers ..... 51

    Optimisations en vue des performances 51

CHAPITRE 3

**Navigateurs et support..... 53**

Panorama des navigateurs web  
 et moteurs de rendu ..... 54  
 Prise en charge de HTML 5 ..... 55  
 Bibliothèques de détection  
 et de modernisation ..... 56  
     Modernizr ..... 56  
     html5shiv (ou html5shiv) ..... 58  
 Frameworks HTML ..... 59

CHAPITRE 4

**Éléments et attributs HTML 5 .... 63**

Modèles de contenu ..... 65  
 Le doctype avant tout ..... 66  
     Rappel des précédents doctypes ..... 67  
 Éléments racines et méta-informations .. 68  
     <html> ..... 68  
         *manifest* ..... 69  
     <head> ..... 70  
     <title> ..... 72  
     <meta> ..... 72  
         <meta name> ..... 73  
         <meta charset> ..... 74  
         <meta http-equiv> ..... 75  
     <link> ..... 76  
     <style> ..... 77  
         *media* ..... 79  
     <base> ..... 80  
         *href* ..... 80  
         *target* ..... 80  
     <body> ..... 80  
 Groupement ..... 81  
     <div> ..... 81  
     <span> ..... 83  
 Liens ..... 83  
     <a> ..... 83  
         *href* et *hreflang* ..... 84  
         *rel* ..... 86  
         *id* et *ancres* ..... 86  
         *target* ..... 87

*download* ..... 88  
         *Liens et blocs* ..... 88  
 Sections et titres ..... 89  
     *Le cas Internet Explorer* ..... 93  
     *Le cas Internet Explorer*  
     *sans JavaScript* ..... 96  
     *Le cas Internet Explorer*  
     *sans JavaScript, bis* ..... 97  
     <section> ..... 97  
     <article> ..... 99  
     <header> ..... 100  
     <footer> ..... 102  
     <nav> ..... 103  
     <aside> ..... 104  
     <address> ..... 106  
     <h1> à <h6> ..... 106  
         *Hiérarchie des éléments de sections*  
         *et outline* ..... 108  
     <hgroup> ..... 112  
 Listes ..... 113  
     <ol> ..... 114  
     <ul> ..... 115  
     <li> ..... 116  
     <dl> ..... 117  
     <dt> ..... 119  
     <dd> ..... 119  
 Texte ..... 120  
     <p> ..... 120  
     <blockquote> ..... 121  
     <q> ..... 123  
         *cite* ..... 123  
     <cite> ..... 123  
     <strong> ..... 124  
     <em> ..... 125  
     <b> ..... 125  
     <i> ..... 127  
     <small> ..... 128  
     <dfn> ..... 128  
     <abbr> ..... 129  
     <code> ..... 130  
     <var> ..... 131

<kbd> . . . . .	131
<samp> . . . . .	132
<sub> . . . . .	133
<sup> . . . . .	133
<time> . . . . .	134
<i>datetime</i> . . . . .	135
<i>pubdate</i> . . . . .	136
<hr> . . . . .	137
  . . . . .	138
<wbr> . . . . .	139
<ins> . . . . .	139
<del> . . . . .	140
<s> . . . . .	141
<pre> . . . . .	142
<mark> . . . . .	143
<ruby> . . . . .	144
<rt> . . . . .	145
<rp> . . . . .	146
<bdo> . . . . .	147
<bdi> . . . . .	147
<b>Contenu embarqué . . . . .</b>	<b>148</b>
<img> . . . . .	148
<i>Formats de compression d'images</i> . . . . .	148
<i>Bref comparatif visuel</i> . . . . .	150
<i>Usage des images en HTML</i> . . . . .	151
<i>src</i> . . . . .	152
<i>alt</i> . . . . .	153
<i>width, height</i> . . . . .	155
<i>usemap</i> . . . . .	156
<i>ismap</i> . . . . .	156
<i>Liens sur images</i> . . . . .	157
<i>Positionnement des images</i> . . . . .	158
<map> . . . . .	159
<area> . . . . .	161
<figure> . . . . .	163
<figcaption> . . . . .	166
<iframe> . . . . .	167
<i>src</i> . . . . .	168
<i>width, height</i> . . . . .	168
<i>sandbox</i> . . . . .	169
<i>srcdoc</i> . . . . .	170
<i>seamless</i> . . . . .	171
<embed> . . . . .	171
<i>Imbrications avec &lt;object&gt;</i> <i>et éléments média</i> . . . . .	173
<object> . . . . .	174
<i>Le cas de Flash</i> . . . . .	176
<param> . . . . .	177
<video> . . . . .	178
<audio> . . . . .	178
<source> . . . . .	179
<track> . . . . .	179
<canvas> . . . . .	179
<b>Données tabulaires . . . . .</b>	<b>180</b>
<table> . . . . .	180
<thead> . . . . .	183
<tfoot> . . . . .	183
<tbody> . . . . .	184
<tr> . . . . .	186
<td> . . . . .	187
<th> . . . . .	189
<caption> . . . . .	190
<colgroup> . . . . .	191
<col> . . . . .	193
<b>Éléments interactifs . . . . .</b>	<b>195</b>
<menu> . . . . .	195
<i>type</i> . . . . .	195
<i>label</i> . . . . .	196
<command> . . . . .	197
<details> . . . . .	199
<summary> . . . . .	202
<device> . . . . .	202
<b>Scripting . . . . .</b>	<b>202</b>
<script> . . . . .	202
<i>Script externe à la page HTML</i> . . . . .	204
<i>Exécution asynchrone</i> . . . . .	205
<i>Exécution différée</i> . . . . .	205
<noscript> . . . . .	205
<b>Attributs HTML globaux . . . . .</b>	<b>206</b>
accesskey . . . . .	207
class . . . . .	208
contextmenu . . . . .	209
contenteditable . . . . .	210

data- . . . . .	211	datetime-local . . . . .	251
dir . . . . .	212	month . . . . .	252
draggable . . . . .	213	week . . . . .	253
dropzone . . . . .	213	number . . . . .	254
hidden . . . . .	213	range . . . . .	254
id . . . . .	214	color . . . . .	255
itemid, itemref, itemscope, itemtype,		<datalist> . . . . .	256
itemprop . . . . .	215	<b>Autres éléments de formulaire</b> . . . . .	257
lang . . . . .	215	<textarea> . . . . .	257
tabindex . . . . .	216	<select> . . . . .	258
title . . . . .	218	<option> . . . . .	259
spellcheck . . . . .	219	<optgroup> . . . . .	260
style . . . . .	220	<button> . . . . .	261
<b>Relations des liens</b> . . . . .	221	<output> . . . . .	262
Quelques relations notables . . . . .	222	<keygen> . . . . .	264
<i>nofollow</i> . . . . .	222	<progress> . . . . .	266
<i>noreferrer</i> . . . . .	223	<meter> . . . . .	268
<i>prefetch</i> . . . . .	224	<b>Construction de formulaires</b> . . . . .	270
<i>first, last, prev, next, up</i> . . . . .	225	<form> . . . . .	270
<b>Attributs événements</b> . . . . .	225	<fieldset> . . . . .	272
		<legend> . . . . .	273
		<label> . . . . .	274
<b>CHAPITRE 5</b>		<b>Attributs communs pour</b>	
<b>Les formulaires (Web Forms)...</b>	<b>231</b>	<b>les éléments de formulaire</b> . . . . .	276
<input> et ses variantes . . . . .	233	Quelques nouveaux attributs HTML 5 . . . . .	278
text . . . . .	235	<i>placeholder</i> . . . . .	278
password . . . . .	236	<i>autofocus</i> . . . . .	279
tel . . . . .	236	<i>autocomplete</i> . . . . .	279
url . . . . .	238	<i>required</i> . . . . .	279
email . . . . .	239	<i>multiple</i> . . . . .	279
search . . . . .	240	<i>dirname</i> . . . . .	280
hidden . . . . .	241	<i>pattern</i> . . . . .	280
radio . . . . .	242	<i>min, max, step</i> . . . . .	280
checkbox . . . . .	243	<b>Une touche de style</b> . . . . .	281
button . . . . .	244	<b>Prise en charge</b> . . . . .	283
reset . . . . .	244		
submit . . . . .	245	<b>CHAPITRE 6</b>	
image . . . . .	245	<b>Les microformats (microdata)...</b>	<b>285</b>
file . . . . .	246	Principe des microformats :	
date . . . . .	249	vers le Web sémantique . . . . .	286
time . . . . .	250	Les prémices . . . . .	288
datetime . . . . .	250		

Microdata à la rescousse . . . . .	290	Interface de contrôle et événements . . . . .	321
Attributs globaux et vocabulaires . . . . .	291	<i>Contrôler la lecture</i> . . . . .	323
<i>itemscope</i> . . . . .	291	<i>Surveiller les événements</i> . . . . .	323
<i>itemtype</i> . . . . .	291	Créer une interface graphique personnalisée . . . . .	325
<i>Vocabulaires</i> . . . . .	292	<i>Détecter les erreurs de lecture</i> . . . . .	330
<i>itemprop</i> . . . . .	294	Détection du support avec <code>canPlayType()</code> . . . . .	331
<i>itemid</i> . . . . .	295	<i>Solution de repli avec Flash et Java</i> . . . . .	333
<i>itemref</i> . . . . .	296	Plein écran . . . . .	334
API DOM Microdata . . . . .	297	Aller plus loin avec les API . . . . .	335
<i>document.getItems()</i> . . . . .	298	Prise en charge de <code>&lt;audio&gt;</code> et <code>&lt;video&gt;</code> par les navigateurs : comment choisir ? . . . . .	336
<i>itemType, itemRef, itemId</i> . . . . .	299	Librairies et lecteurs . . . . .	338
<i>properties</i> . . . . .	300		
<i>properties.namedItem()</i> . . . . .	300		
<i>itemValue</i> . . . . .	300		
Rich Snippets . . . . .	302		
<b>CHAPITRE 7</b>		<b>CHAPITRE 8</b>	
<b>Audio et Vidéo . . . . .</b>	<b>305</b>	<b>Dessin avec Canvas . . . . .</b>	<b>339</b>
Conteneurs, codecs, licences et support . . . . .	307	L'élément <code>&lt;canvas&gt;</code> . . . . .	341
Vidéo . . . . .	309	Base de départ et contexte graphique . . . . .	342
<i>Theora</i> . . . . .	309	Coordonnées . . . . .	342
<i>WebM</i> . . . . .	310	Formes géométriques . . . . .	344
<i>H.264</i> . . . . .	310	Chemins . . . . .	345
Audio . . . . .	311	<i>beginPath()</i> et <i>closePath()</i> . . . . .	346
<i>MP3 (Mpeg-1 Audio Layer 3)</i> . . . . .	311	<i>moveTo()</i> et <i>lineTo()</i> . . . . .	347
<i>AAC (Advanced Audio Coding)</i> . . . . .	312	<i>fill()</i> et <i>stroke()</i> . . . . .	348
<i>Vorbis</i> . . . . .	312	<i>rect()</i> . . . . .	348
Les balises <code>media</code> . . . . .	313	<i>arcTo()</i> . . . . .	349
<code>&lt;audio&gt;</code> . . . . .	315	<i>arc()</i> . . . . .	350
<code>&lt;video&gt;</code> . . . . .	315	<i>bezierCurveTo()</i> . . . . .	351
<code>&lt;source&gt;</code> . . . . .	316	<i>quadraticCurveTo()</i> . . . . .	352
<code>&lt;track&gt;</code> . . . . .	317	Styles de traits, remplissages et couleurs . . . . .	353
Attributs pour <code>&lt;track&gt;</code> . . . . .	318	Dégradés . . . . .	354
Attributs pour <code>&lt;audio&gt;</code> et <code>&lt;video&gt;</code> . . . . .	319	Transformations et états du contexte . . . . .	356
<i>src</i> . . . . .	319	<i>save()</i> et <i>restore()</i> . . . . .	358
<i>width</i> et <i>height</i> . . . . .	319	Images . . . . .	360
<i>controls</i> . . . . .	319	Pixels . . . . .	362
<i>poster</i> . . . . .	319	Créer des pixels . . . . .	364
<i>autoplay</i> . . . . .	320	Lire des pixels . . . . .	365
<i>preload</i> . . . . .	320	Modifier des pixels . . . . .	367
<i>loop</i> . . . . .	321	Motifs et sprites . . . . .	370
		Texte . . . . .	375
		Ombrages . . . . .	377
		Transparence, compositions et masques . . . . .	379

Transparence générale . . . . .	379	Lire des fichiers avec <code>FileReader</code> . . . . .	426
Compositions . . . . .	380	Utiliser <code>Canvas</code> . . . . .	430
Masques . . . . .	382	<code>CreateObjectURL</code> . . . . .	432
<b>Contrôle clavier et souris</b> . . . . .	385	<b>Upload simple avec PHP</b> . . . . .	433
Souris . . . . .	386	<b>Upload avec XMLHttpRequest 2</b> . . . . .	434
Clavier . . . . .	389	<code>FormData</code> . . . . .	436
<b>Animation et jeux</b> . . . . .	391	<b>Drag &amp; Drop</b> . . . . .	439
Jeux . . . . .	394	Et bientôt, écrire des fichiers,	
<b>Vidéo et audio</b> . . . . .	395	accéder au système . . . . .	440
<b>Prise en charge</b> . . . . .	397	Prise en charge . . . . .	440
<b>Librairies</b> . . . . .	399		
Et la 3D ? . . . . .	400	<b>CHAPITRE 11</b>	
Et le graphisme vectoriel (SVG) ? . . . . .	401	<b>Gestion du glisser-déposer</b>	
Création au format SVG . . . . .	402	<b>(Drag &amp; Drop) . . . . .</b>	<b>441</b>
Inclusion HTML . . . . .	403	<b>Principe</b> . . . . .	442
Syntaxe . . . . .	405	<b>Événements et attributs mis en œuvre</b> . . . . .	443
Support . . . . .	406	<b>Glisser...</b> . . . . .	444
Alternatives et librairies . . . . .	406	L'attribut <code>draggable</code> . . . . .	444
		Un zeste de CSS . . . . .	445
		<b>Déposer!</b> . . . . .	447
		L'attribut <code>dropzone</code> . . . . .	448
		Les événements <code>dragenter</code> et <code>dragleave</code> . . . . .	449
		L'événement <code>dragover</code> . . . . .	449
		<b>L'interface <code>DataTransfer</code></b> . . . . .	451
		L'événement <code>dragstart</code> . . . . .	452
		<i>effectAllowed</i> et <i>dropEffect</i> . . . . .	452
		<code>setData()</code> . . . . .	453
		L'événement <code>drop</code> et <code>getData()</code> . . . . .	454
		L'événement <code>dragend</code> . . . . .	456
		Aller plus loin . . . . .	456
		Script complet . . . . .	456
		<b>Glisser-déposer de fichiers</b> . . . . .	458
		Dépôt depuis le système ( <code>drag-in</code> ) . . . . .	458
		En symbiose avec <code>FileReader</code>	
		et <code>Data URI</code> . . . . .	460
		Dépose d'éléments	
		hors du navigateur ( <code>drag-out</code> ) . . . . .	464
		<b>Prise en charge du glisser-déposer</b> . . . . .	466
<b>CHAPITRE 9</b>			
<b>Géolocalisation . . . . .</b>	<b>407</b>		
<b>Principe</b> . . . . .	409		
<b>Les mains dans le code</b> . . . . .	410		
Déclencher la localisation . . . . .	411		
Travailler avec la position			
et les coordonnées . . . . .	412		
Gestion des erreurs . . . . .	413		
Options supplémentaires . . . . .	414		
Utiliser une carte . . . . .	416		
<b>Prise en charge de l'API Geolocation</b>			
<b>par les navigateurs</b> . . . . .	418		
Alternative avec <code>geo-location-javascript</code> . . . . .	419		
<b>CHAPITRE 10</b>			
<b>Interactions avec les fichiers</b>			
<b>(File API) . . . . .</b>	<b>421</b>		
<b>Principe</b> . . . . .	422		
<b>Fonctionnement</b> . . . . .	423		
Événement <code>onchange</code> . . . . .	424		
Recueillir les informations			
des fichiers sélectionnés . . . . .	425		

## CHAPITRE 12

**Événements envoyés  
par le serveur (« push ») ..... 467**

Push-toi, j'arrive .....	468
Principe .....	469
Sous le capot .....	469
Côté client (navigateur) .....	469
<i>Propriétés et méthodes</i> .....	470
Côté serveur .....	470
<i>Mise en place d'un flux continu</i> ..	471
Syntaxe des messages source .....	475
<i>data</i> .....	476
<i>id</i> .....	477
<i>event</i> .....	479
<i>retry</i> .....	480
<i>Utiliser JSON</i> .....	481
Prise en charge .....	482

## CHAPITRE 13

**Échange d'informations entre  
documents (Web Messaging) .. 483**

Fonctionnement .....	484
Sécurité .....	488
Vérification de l'origine .....	488
Vérification du contenu .....	488
Données transférées et JSON .....	489
Source et réponse .....	491
Prise en charge .....	494

## CHAPITRE 14

**Communication en temps réel  
(Web Sockets) ..... 495**

Côté serveur .....	497
phpwebsocket .....	497
Côté navigateur .....	500
Application HTML .....	501
Se connecter .....	504
Envoyer des données .....	504
Recevoir des messages .....	505
Gérer les erreurs .....	505
Fermer la connexion .....	505

Aller plus loin ..... 507

Prise en charge ..... 507

## CHAPITRE 15

**Stockage des données locales  
(Web Storage) ..... 509**

Deux espaces de stockage .....	511
Stockage de session .....	511
Local Storage .....	511
L'interface Storage .....	512
Stockage, lecture, suppression .....	512
Un compteur de visites avec localStorage ..	514
Surveiller le dépassement de quota .....	515
Un soupçon de JSON .....	516
Stocker sur un événement .....	519
Stockage à intervalles réguliers .....	519
Événements .....	520
Prise en charge .....	521
Alternatives à Web Storage .....	521

## CHAPITRE 16

**Bases de données (Indexed Database  
& Web SQL Database)..... 523**

L'aube d'IndexedDB .....	525
Philosophie .....	525
Ouvrir une base et créer un catalogue ..	527
Insérer des données dans une transaction ..	530
Afficher le contenu .....	532
Utiliser un index .....	534
Effacer un catalogue .....	536
Perspectives .....	536
Prise en charge .....	537
Le crépuscule de Web SQL Database ? ..	538
Philosophie .....	538
Ouvrir une base .....	538
Initier une transaction .....	539
Créer une table .....	539
Insérer des données .....	540
Exploiter les résultats .....	540
Perspectives .....	541
Prise en charge .....	542

CHAPITRE 17

**Applications web hors ligne .... 543**

Principe ..... 544

En ligne ou hors ligne ? ..... 545

    Structure complète ..... 547

Liste des fichiers à mettre en cache (manifeste) ..... 548

    Syntaxe pour le manifeste ..... 549

*La section CACHE* ..... 550

*La section NETWORK* ..... 550

*La section FALLBACK* ..... 551

    Élaboration et test du cache sous Chrome et Firefox ..... 552

    Mise à jour du cache ..... 554

L'API Application cache ..... 554

    Propriétés ..... 555

    Événements ..... 555

    Méthodes ..... 557

Une application hors ligne ..... 558

Prise en charge ..... 560

CHAPITRE 18

**Historique de navigation..... 561**

Navigation dans l'historique ..... 562

    History ..... 562

    Location ..... 563

Modification dynamique de l'historique ..... 564

    pushState() ..... 565

    replaceState() ..... 566

    The king of popstate ..... 566

Simulation ..... 567

    Cas pratique ..... 568

    Réécriture d'adresse ..... 574

Les ancres et l'événement hashchange .. 575

Détection ..... 576

Prise en charge ..... 576

CHAPITRE 19

**JavaScript en (multi)tâche de fond : les Web Workers ..... 577**

Principe ..... 579

Fonctionnement ..... 580

    Initialisation ..... 580

    Communication ..... 581

    Terminaison ..... 583

    Gestion des erreurs ..... 583

    Contexte ..... 584

    Fonctions complémentaires ..... 584

    Blob à la rescousse ..... 585

Prise en charge ..... 586

CHAPITRE 20

**JavaScript, le DOM et l'API Selectors ..... 587**

Les bases de JavaScript ..... 589

    Variables ..... 589

    Types simples ..... 590

*Objets* ..... 591

*Fonctions* ..... 592

*Boucles* ..... 593

Méthodes de sélection DOM ..... 594

    getElementById() ..... 594

    getElementsByName() ..... 595

    getElementsByClassName() ..... 595

    querySelector() ..... 595

    querySelectorAll() ..... 595

Propriétés et méthodes DOM ..... 595

Méthodes de manipulation DOM ..... 596

    createElement() ..... 596

    appendChild() ..... 597

    removeChild() ..... 597

    insertBefore() ..... 597

    createTextNode() ..... 597

Méthodes pour formulaires ..... 597

Gestionnaires d'événements ..... 598

Autres fonctions ..... 600

Prise en charge ..... 600

**Conclusion et perspectives..... 601**

    Quid des frameworks JavaScript et de Flash ? ..... 601

    Perspectives d'avenir ..... 602

# Fonctionnalités modifiées et obsolètes



## Différences HTML 5 par rapport à HTML 4

Une publication du W3C recense les nouveaux éléments et attributs, ceux qui ont été supprimés, et ceux qui ont été modifiés. D'autres différences y sont mises en évidence notamment pour l'API et la syntaxe en général.

**RESSOURCE** Spécification W3C

HTML 5 differences from HTML 4

▶ <http://www.w3.org/TR/html5-diff/>

## Fonctionnalités obsolètes

Ces pratiques ne généreront pas d'erreur au validateur, mais des avertissements :

- attribut `http-equiv` de l'élément `meta` : l'attribut `lang` doit être utilisé en remplacement ;
- attribut `border` sur l'élément `img` : CSS doit être utilisé en remplacement ;
- attribut `language` sur l'élément `script` : il doit être omis ou remplacé par l'attribut `type` avec la valeur `text/javascript` ;
- attribut `name` sur l'élément `a` : il doit être omis ou remplacé par l'attribut `id`. Une exception cependant : s'il est présent, il ne doit pas être vide, mais posséder la même valeur que l'`id`.

## Fonctionnalités obsolètes non conformes

### Éléments

Ces éléments sont entièrement obsolètes et **ne doivent plus être utilisés** :

- `applet` : remplacé par `embed` ou `object` ;
- `acronym` : remplacé par `abbr` ;
- `bgsound` : remplacé par `audio` ;
- `dir` : remplacé par `ul` ;
- `frame`, `frameset`, `noframes` : remplacé par `iframe` et CSS, ou par les langages interprétés côté serveur pour générer des pages complètes ;
- `isindex` : utiliser un formulaire `form` combiné à un champ texte `input` ;
- `listing`, `xmp` : remplacé par `pre` et `code` ;
- `noembed` : remplacé par `object` quand une alternative est nécessaire ;
- `plaintext` : utiliser le type `MIMEtext/plain` ;
- `strike` : remplacé par `del` ;
- `basefont`, `big`, `blink`, `center`, `font`, `marquee`, `multicol`, `nobr`, `spacer`, `tt`, `u` : remplacé par les propriétés CSS équivalentes.

Précisions au sujet de `tt` :

- utiliser `kbd` pour baliser une entrée au clavier ;
- utiliser `var` pour baliser une variable ;
- utiliser `code` pour baliser un code source ;
- utiliser `samp` pour baliser une sortie de données.

Précisions au sujet de `u` :

- utiliser `em` pour baliser une emphase ;
- utiliser `b` pour baliser des mots-clés ;
- utiliser `mark` pour baliser un texte surligné ou marquant une référence.

### Attributs

Ces attributs sont obsolètes, bien qu'appartenant à des éléments qui font encore partie du langage et **ne doivent plus être utilisés** :

- `charset` sur `a` et `link` : utiliser un en-tête HTTP Content-Type ;
- `coords` et `shape` sur `a` : utiliser l'élément `area` au lieu de `a` ;
- `methods` sur `a` et `link` : utiliser HTTP OPTIONS ;
- `name` sur `a`, `embed`, `img` et `option` : utiliser l'attribut `id` ;

- `rev` sur `a` et `link` : utiliser l'attribut `rel` ;
- `urn` sur `a` et `link` : utiliser l'attribut `href` ;
- `nohref` sur `area` : facultatif, ne plus utiliser ;
- `profile` sur `head` : facultatif, ne plus utiliser ;
- `version` sur `html` : facultatif, ne plus utiliser ;
- `usemap` sur `input` : utiliser l'élément `img` au lieu de `input` ;
- `longdesc` sur `iframe` et `img` : utiliser un élément `a` ou une image map ;
- `lowsrc` sur `img` : utiliser une image compressée progressive (présente dans `src`) ;
- `target` sur `link` : facultatif, ne plus utiliser ;
- `archive`, `classid`, `code`, `codebase`, `codetype` sur `object` : utiliser `data` et `type` pour invoquer des extensions et l'élément `param` pour les paramètres ;
- `declare` sur `object` : répéter l'élément `object` complètement ;
- `standby` sur `object` : optimiser la ressource pour un chargement incrémental ;
- `type` et `valuetype` sur `param` : utiliser `name` et `value` sans déclarer le type de valeur ;
- `event` et `for` sur `script` : utiliser les événements DOM ;
- `datapagesize` sur `table` : facultatif, ne plus utiliser ;
- `abbr` sur `td` et `th` : utiliser un texte explicite ou préciser avec `title` ;
- `axis` sur `td` et `th` : utiliser l'attribut `scope` ;
- `datasrc`, `datafld` et `dataformatas` : utiliser XMLHttpRequest.

Pour les attributs suivants, **utiliser les propriétés CSS en remplacement** :

- `alink`, `link`, `marginbottom`, `marginheight`, `marginleft`, `marginright`, `marginintop`, `marginwidth`, `text`, `vlink` sur `body` ;
- `bgcolor` sur `body`, `table`, `td`, `th`, `tr` ;
- `background` sur `body`, `table`, `thead`, `tbody`, `tfoot`, `tr`, `td`, `th` ;
- `clear` sur `br` ;
- `align` sur `caption`, `div`, `hr`, `h1` à `h6`, `iframe`, `input`, `img`, `legend`, `col`, `embed`, `input`, `img`, `legend`, `object`, `p`, `table`, `tbody`, `thead`, `tfoot`, `td`, `th`, `tr` ;
- `char`, `charoff`, `valign`, `width` sur `col`, `tbody`, `thead`, `tfoot`, `td`, `th`, `tr` ;
- `compact` sur `dl` ;
- `hspace`, `vspace` sur `embed`, `input`, `img`, `legend`, `object` ;
- `color`, `noshade`, `size`, `width` sur `hr` ;
- `allowtransparency`, `frameborder`, `marginheight`, `marginwidth`, `scrolling`, sur `iframe` ;
- `border` sur `img`, `object`, `table` ;

- `type` sur `ul`, `li` ;
- `compact` sur `menu`, `ol`, `ul` ;
- `width` sur `pre`, `table` ;
- `height`, `nowrap` sur `td`, `th` ;
- `cellspacing`, `cellpadding`, `frame`, `rules` sur `table`.

# Feuilles de style CSS

# B

Tout n'est que cascade, style et volupté. Ainsi, du moins, devrait se présenter tout document HTML.



**Figure 2-1** background:red ; color:white

Dans des temps reculés, lors de l'apparition du HTML et des premiers navigateurs, l'esthétisme des pages était très limité. L'essentiel de l'information n'était pas destiné à un large public et on pouvait se contenter d'une mise en pages sommaire. Peu à peu, des balises ont été ajoutées, et certains éléments ont été détournés de leur usage initial par les webdesigners/intégrateurs pour obtenir un rendu plus complexe à l'écran.

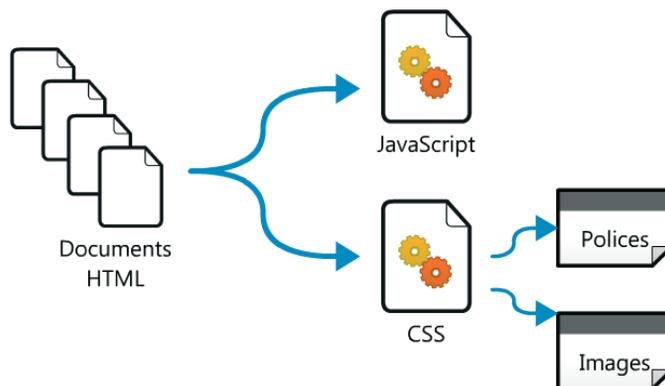
Ainsi, les tables ont été mises à profit pour disposer du contenu en colonnes puis en grilles, on a vu naître des artifices tels que les images de séparation (nommées *spacer.gif*) ou encore l'imbrication de multiples balises de présentation (par exemple `<font>`) aujourd'hui obsolètes.

Ce qui palliait un manque intrinsèque du HTML en capacités de contrôle graphique est devenu au fur et à mesure un cauchemar en termes de maintenance du code, de lisibilité et d'accessibilité du contenu. La sémantique n'était que peu considérée, et la « soupe de tags » au menu de nombreux sites.

C'est pourquoi les feuilles de style en cascade (*Cascading Style Sheets*) ont été adoptées par le W3C pour régir l'apparence des éléments HTML d'une page, concernant la disposition, les dimensions, les couleurs, afin de séparer le contenu et la forme.

Une feuille de style est applicable à une infinité de documents HTML, ce qui en facilite la maintenance et réduit les temps de chargement.

**Figure 2-2**  
Une feuille CSS pour  
de multiples pages HTML



L'objet de ce chapitre n'est pas de dresser un inventaire exhaustif de toutes les techniques afférentes à CSS, mais de proposer une introduction et un petit aide-mémoire vis-à-vis de HTML 5 auquel les feuilles de style sont intimement liées.

### RESSOURCES

- Goetter Raphaël, *CSS 3, Pratique du design web*, Eyrolles 2011
- Goetter Raphaël, *CSS avancées*, Eyrolles 2011

Les spécifications sont nombreuses et pour certaines encore en mouvement.

#### RESSOURCE Spécifications CSS

Cascading Style Sheets Level 2 Revision 1 (CSS 2.1)

▶ <http://www.w3.org/TR/CSS21/>

Tableau récapitulatif des spécifications

▶ <http://www.w3.org/Style/CSS/current-work>

Cascading Style Sheets (CSS)

▶ <http://www.w3.org/TR/CSS/>

## Principe général

L'application d'une feuille de style CSS s'effectue lorsque celle-ci est liée au document par la balise `<link>`, présente dans la section `<head>`.

```
<link rel="stylesheet" href="styles.css">
```

Plusieurs feuilles de style peuvent être chargées de la sorte pour un seul document. Il est aussi envisageable de déclarer directement des instructions CSS entre les balises `<style>` et `</style>`, situées également dans `<head>` ou dans une portion du document, mais cette façon de faire ne facilite pas la maintenance ni la mise en cache.

Dans le fichier de la feuille de style, on retrouve une ou plusieurs déclarations CSS. Elles comprennent un sélecteur dont le rôle est de cibler les éléments concernés par chaque déclaration, suivi d'un bloc entre accolades regroupant les propriétés à appliquer.

```
p {
  text-align: center;
  font-weight: bold;
  color: orange;
}
header img {
  border: 3px solid gray;
  margin-bottom: 3em;
}
```

Dans cet exemple, trois propriétés sont appliquées aux éléments `<p>` (les paragraphes) et deux aux éléments `<img>` situés dans `<header>`.

## Sélecteurs

L'art d'écrire de bons sélecteurs est somme toute un grand jeu d'assemblage parmi les quelques briques de base existantes, pour s'adapter à la structure du document HTML qui doit être stylé. Si HTML et CSS sont issus du même auteur, alors il lui appartient de nommer les éléments en conséquence grâce à des classes, des identifiants ou des attributs pour les lier aux déclarations CSS, et ce en toute simplicité pour faciliter la lecture du code source ainsi que son analyse par le navigateur.

Tableau 2-1 Quelques sélecteurs

Sélecteur	Exemples	Éléments concernés
Sélecteur d'élément	<code>nav { }</code> <code>p { }</code> <code>ul { }</code>	<code>&lt;nav&gt;</code> <code>&lt;p&gt;</code> <code>&lt;ul&gt;</code>
Sélecteur de classe	<code>.remarque { }</code> <code>div.remarque { }</code>	Tout élément portant l'attribut <code>class="remarque"</code> <code>&lt;div class="remarque"&gt;</code> .
Sélecteur d'id	<code>#intro { }</code> <code>header#intro { }</code>	Tout élément portant l'attribut <code>id="intro"</code> <code>&lt;header id="intro"&gt;</code> .
Sélecteur d'attribut	<code>[alt] { }</code> <code>input[type=submit] { }</code> <code>[rel=nofollow] { }</code>	Tout élément possédant un attribut <code>alt</code> <code>&lt;input type="submit"&gt;</code> . Tout élément portant l'attribut <code>rel="nofollow"</code> .
Sélecteur d'enfant direct	<code>ul&gt;li { }</code>	Tout élément <code>&lt;li&gt;</code> enfant direct d'un <code>&lt;ul&gt;</code> .
Sélecteur d'élément adjacent	<code>h1+p { }</code>	Tout élément <code>&lt;p&gt;</code> immédiatement précédé par un élément <code>&lt;h1&gt;</code> .
Sélecteur d'éléments frères	<code>h2~p { }</code>	Un ou plusieurs éléments <code>&lt;p&gt;</code> précédés par un élément <code>&lt;h2&gt;</code> .

Les sélecteurs peuvent s'enchaîner, séparés par des espaces, pour combiner leurs effets.

```
nav ul>li { ... }
```

Ce sélecteur s'adresse aux éléments `<li>` enfants directs de `<ul>`, lui-même situé dans un quelconque `<nav>`.

```
#intro p.remarque { ... }
```

Ce sélecteur s'adresse aux éléments `<p>` de classe `remarque`, situés dans l'élément possédant l'identifiant `intro`.

Séparés par des virgules, les sélecteurs s'appliquent à une énumération de plusieurs familles d'éléments.

```
nav ul>li, #intro p.remarque { ... }
```

Cette déclaration s'adresse aux éléments `<li>` enfants directs de `<ul>`, lui-même situé dans un quelconque `<nav>`... et indépendamment aux éléments `<p>` de classe `remarque`, situés dans l'élément possédant l'identifiant `intro`.

## Propriétés

Les propriétés sont des paires de clés et valeurs. La clé est le nom de la propriété, suivie des deux-points, terminée par la valeur qui peut revêtir différentes formes : mots-clés, valeur numérique simple, valeur numérique avec unité, chaîne de texte, etc.

Les combinaisons offertes par les propriétés CSS, leurs valeurs et les techniques associées sortent du cadre de ce modeste chapitre qui pourrait à lui seul dépasser tous les autres en longueur. Pour en savoir plus, consultez les références proposées dans l'introduction.

**Tableau 2-2** Quelques unités courantes

Unité	Description	Type d'unité
<code>px</code>	Pixels	Relative
<code>%</code>	Pourcentage (en général par rapport à l'élément parent)	Relative
<code>em</code>	Cadratin (relatif à la taille de la police)	Relative
<code>ex</code>	Hauteur de la lettre x	Relative
<code>in</code>	Pouces (= 2,54 cm)	Absolue
<code>cm</code>	Centimètres (= 10 mm)	Absolue
<code>mm</code>	Millimètre	Absolue
<code>pt</code>	Points (= 1/72 in)	Absolue
<code>pc</code>	Picas (= 12 pt)	Absolue

Les unités les plus pertinentes pour un affichage à l'écran restent `px`, `%`, et `em`.

**Tableau 2-3** Texte

Propriété	Rôle
<code>color</code>	Couleur du texte
<code>font</code>	(Déclaration groupée)
<code>font-weight</code>	Graisse
<code>font-size</code>	Taille de police
<code>font-family</code>	Famille de police
<code>font-variant</code>	Variante
<code>font-stretch</code>	Étirement du texte
<code>text-decoration</code>	Décoration de texte

**Tableau 2-3** Texte (suite)

Propriété	Rôle
<code>text-transform</code>	Transformation de texte
<code>text-align</code>	Alignement
<code>text-justify</code>	Justification
<code>text-indent</code>	Indentation
<code>line-height</code>	Hauteur de ligne
<code>word-spacing</code>	Espacement de mot
<code>word-wrap</code>	Retour à la ligne

**Exemple**

```
p {
  font-weight: bold;
  color: #005A9C;
  text-align: left;
  line-height: 150%;
}
```

**Tableau 2-4** Fonds

Propriété	Rôle
<code>background</code>	(Déclaration groupée)
<code>background-color</code>	Couleur de fond
<code>background-image</code>	Image de fond
<code>background-repeat</code>	Répétition de l'image de fond
<code>background-position</code>	Position de l'image de fond
<code>background-attachment</code>	Attache de l'image de fond par rapport à la page
<code>background-origin</code>	Position du fond par rapport à la boîte de l'élément
<code>background-size</code>	Taille de l'image de fond par rapport aux dimensions de l'élément

**Exemple**

```
div {
  background-image: url(image.jpg);
  background-repeat: no-repeat;
  background-size: 100% 100%;
  background-origin: content-box;
}
```

**RESSOURCE Tutoriels CSS sur Alsacreations**

Arrière-plans avec CSS 3 Backgrounds

► <http://www.alsacreations.com/tuto/lire/808-arriere-plans-css3-background.html>

**Tableau 2-5** Ombrages et transparence

Propriété	Rôle
<code>text-shadow</code>	Ombrage du texte
<code>text-outline</code>	Contour du texte
<code>box-shadow</code>	Ombrage d'un élément boîte
<code>opacity</code>	Opacité

**Exemple**

```

footer {
  opacity: 0.5;
  text-shadow: 0px 0px 9px #777;
  box-shadow: rgba(0,0,0,0.4) 10px 10px 0 10px;
}

```

**RESSOURCE Tutoriels CSS sur Alsacreations**

Les ombrages en CSS 3

▶ <http://www.alsacreations.com/tuto/lire/910-creer-des-ombrages-ombres-css-box-shadow-text-shadow.html>

**Tableau 2-6** Bordures

Propriété	Rôle
<code>border</code>	(Déclaration groupée)
<code>border-color</code>	Couleur du bord
<code>border-spacing</code>	Espacement du bord
<code>border-collapse</code>	Fusion du bord
<code>border-style</code>	Style du bord
<code>border-width</code>	Largeur de la bordure
<code>border-radius</code>	Rayon du bord arrondi
<code>border-image</code>	Style de bord avec image
<code>outline</code>	Contour
<code>outline-style</code>	Style du contour
<code>outline-color</code>	Couleur du contour
<code>outline-width</code>	Largeur du contour

**Exemple**

```

.arrondi {
  border: 5px solid yellow;
  border-radius: 20px;
}

```

```
:focus {
  outline: thick solid #fc0;
}
```

**Tableau 2-7** Positionnement et dimensionnement

Propriété	Rôle
<code>display</code>	Mode d'affichage
<code>float, clear</code>	Mode flottant
<code>position</code>	Positionnement
<code>z-index</code>	Ordre de recouvrement « vertical »
<code>overflow</code>	Mode de dépassement de bloc
<code>overflow-y</code>	Mode de dépassement de bloc vertical
<code>overflow-x</code>	Mode de dépassement de bloc horizontal
<code>width</code>	Largeur
<code>height</code>	Hauteur
<code>top</code>	Position par rapport au haut
<code>left</code>	Position par rapport à la gauche
<code>right</code>	Position par rapport à la droite
<code>bottom</code>	Position par rapport au bas
<code>padding</code>	Marge interne
<code>margin</code>	Marge externe
<code>min-height</code>	Hauteur minimale
<code>max-height</code>	Hauteur maximale
<code>min-width</code>	Largeur minimale
<code>max-width</code>	Largeur maximale
<code>vertical-align</code>	Alignement vertical
<code>visibility</code>	Visibilité
<code>rotation</code>	Rotation
<code>rotation-point</code>	Point de rotation

### Exemple

```
img.illustration {
  float: right;
  margin-left: 2em;
  max-width: 50%;
  rotation: 10deg;
  rotation-point: bottom left;
}
```

Tableau 2-8 Listes

Propriété	Rôle
<code>list-style</code>	Style de liste
<code>list-style-type</code>	Type de puce pour les éléments de la liste
<code>list-style-image</code>	Image de puce pour les éléments de la liste
<code>list-style-position</code>	Position de la puce

**Exemple**

```
ul li {
  list-style: disc;
}
```

Tableau 2-9 Autres

Propriété	Rôle
<code>cursor</code>	Apparence du curseur
<code>direction</code>	Direction d'écriture

**Exemple**

```
input[type=submit]
  cursor: pointer;
}
```

Tableau 2-10 Animations

Propriété	Rôle
<code>animation</code>	(Déclaration groupée)
<code>animation-delay</code>	Délai d'animation
<code>animation-direction</code>	Sens d'animation
<code>animation-duration</code>	Durée d'animation
<code>animation-iteration-count</code>	Nombre d'itérations
<code>animation-name</code>	Nom d'animation
<code>animation-timing-function</code>	Fonction d'accélération

**Exemple**

```
div {
  animation-name: 'diagonale';
  animation-duration: 3s;
  animation-iteration-count: 5;
}
```

```
@keyframes 'diagonale' {
  from {
    left: 0;
    top: 0;
  }
  to {
    left: 150px;
    top: 200px;
  }
}
```

**Tableau 2-11** Transformations

Propriété	Rôle
<code>transform</code>	(Déclaration groupée)
<code>transform-origin</code>	Point d'origine de la transformation
<code>transform-style</code>	Style de transformation
<code>perspective</code>	Effet de perspective
<code>perspective-origin</code>	Point d'origine de la perspective
<code>backface-visibility</code>	Visibilité de l'arrière de l'élément transformé

**Exemple**

```
h1 {
  transform: rotate(8deg);
}
```

**Tableau 2-12** Transitions

Propriété	Rôle
<code>transition</code>	(Déclaration groupée)
<code>transition-delay</code>	Délai de transition
<code>transition-duration</code>	Durée de transition
<code>transition-property</code>	Propriété à laquelle appliquer la transition
<code>transition-timing-function</code>	Fonction d'accélération pour la transition

**Exemple**

```
img.transition.couleur {
  transition: transform .5s ease-in;
}
img.transition.couleur:hover {
  transform: rotate(10deg);
}
```

**RESSOURCE Tutoriels CSS sur Alsacreations**

Transitions CSS 3

▶ <http://www.alsacreations.com/tuto/lire/873-transitions-css3-animations.html>

Une petite quantité de nouvelles propriétés CSS 3 ont été implémentées au fur et à mesure dans les moteurs de rendu, officiellement de manière expérimentale, avec des préfixes spécifiques. C'est par exemple le cas de `border-radius` qui existait initialement sous le nom de `-moz-border-radius` pour les navigateurs Gecko (Mozilla), `-webkit-border-radius` pour les navigateurs WebKit (Chrome, Safari). Opera a quant à lui choisi de l'intégrer directement, mais utilise le préfixe `-o-` pour ses propres expérimentations. Une fois tous les tests concluants, la propriété adopte son nom définitif, sans préfixe.

**RESSOURCE Tutoriels CSS sur Alsacreations**

Les préfixes vendeurs en CSS

▶ <http://www.alsacreations.com/article/lire/1159-prefixes-vendeurs-css-proprietaires.html>

## Pseudo-classes et pseudo-éléments

Les pseudo-classes et pseudo-éléments affinent les capacités des sélecteurs. Ils s'écrivent à la suite du sélecteur initial, concaténés par un signe deux-points « : ».

**Tableau 2-13** Quelques pseudo-classes et pseudo-éléments

Pseudo-*	Rôle
<code>:link</code>	Lien
<code>:visited</code>	Lien visité
<code>:hover</code>	Élément survolé
<code>:active</code>	Élément actif
<code>:focus</code>	Élément ayant le focus
<code>:first-child</code>	Premier enfant
<code>:last-child</code>	Dernier enfant
<code>:nth-child(n)</code>	n-ième enfant
<code>:nth-last-of-child(n)</code>	n-ième dernier enfant
<code>:nth-of-type(n)</code>	n-ième type d'élément
<code>:nth-last-of-type(n)</code>	n-ième dernier type d'élément
<code>:first-of-type</code>	Premier type d'élément

**Tableau 2-13** Quelques pseudo-classes et pseudo-éléments (suite)

Pseudo-*	Rôle
<code>:last-of-type</code>	Dernier type d'élément
<code>:only-child</code>	Enfant unique
<code>:only-of-type</code>	Élément de type unique
<code>:checked</code>	État coché
<code>:enabled</code>	État activé
<code>:indeterminate</code>	État indéterminé
<code>:not(expr)</code>	Négation de l'expression
<code>::first-letter</code>	Première lettre
<code>::first-line</code>	Première ligne

**Exemple**

```
a:hover {
  text-decoration:underline;
}
p::first-letter {
  font-size: 2em;
}
```

## Règles @

Les « règles @ » sont des instructions plus évoluées pouvant se retrouver dans les feuilles de style pour moduler leur comportement ou ajouter des informations qui ne pourraient trouver leur place dans des déclarations classiques.

**Tableau 2-14** Quelques règles @

Propriété	Rôle
<code>@import</code>	Importer une autre feuille de style CSS.
<code>@charset</code>	Déclaration de l'encodage des caractères.
<code>@page</code>	Définir des règles générales pour les médias paginés.
<code>@font-face</code>	Importer un fichier de police (fonte) externe.
<code>@media</code>	Définir des requêtes de média.

## Exemple

```
@import "autres_styles.css";
@import url("impression.css") print;

@page {
  size: 15cm 20cm;
  margin: 2cm;
  marks: cross;
}

@font-face {
  font-family: maPolice;
  src: url(ToutSaufComicSans.otf);
  font-weight: bold;
}

p {
  font-family: maPolice;
}

@media print {
  body {
    font-size: 2em;
    background: white;
  }
  nav {
    display: none;
  }
}
```

## Media queries

Les *media queries* (ou requêtes de média) sont une fonctionnalité bien utile de CSS pour adapter le design et la présentation générale d'une page web selon le périphérique de consultation. Depuis la navigation sur mobiles jusqu'aux plages braille, la présentation est ajustée selon les capacités de rendu, par exemple en modifiant la taille du texte ou son agencement.

La syntaxe d'une *media query* est une suite de conditions à satisfaire. Au besoin, le navigateur évaluera l'expression et chargera les styles y correspondant ou les rechargera dynamiquement s'il y a lieu.

### Exemple de media query

```
@media screen and (min-width: 200px) and (max-width: 640px) {  
  section {  
    display: block;  
    clear: both;  
    margin: 0;  
  }  
}
```

Ici, l'on s'adresse à un écran (*screen*) dont la résolution en largeur est comprise entre 200 et 640 pixels. Si le navigateur se retrouve dans cette condition, les éléments de type `<section>` passeront tous en mode bloc, sans aucune marge, pour une meilleure disposition verticale sur un petit écran.

Pour découvrir la magie des *media queries*, consultez l'article en ligne rédigé par votre auteur dévoué.

#### RESSOURCE Tutoriels CSS sur Alsacreations

Les Media Queries CSS 3

► <http://www.alsacreations.com/article/lire/930-css3-media-queries.html>

# Accessibilité et ARIA



Un sage d'Ikea disait : « Un bon design doit toujours rester accessible ». La future recommandation WAI-ARIA permettra, au sein de HTML 5, le respect des principes fondamentaux de l'accessibilité numérique.



**Figure 3-1** Des vitamines pour tous !

L'accessibilité numérique fait l'objet de moult débats dans la communauté des web designers, intégrateurs, experts comme débutants. Sa vocation est de rendre accessibles la technologie et ses bienfaits, notamment Internet, à tous les humains. Cela suppose donc une prise en compte de certains handicaps, voire de caractéristiques purement sociales et matérielles, puisque nous ne sommes pas tous logés à la même enseigne.

Cette intention est louable et reconnue comme une obligation citoyenne par l'Europe. La plupart des gouvernements occidentaux ont voté des mesures destinées à favoriser la prise en compte de l'accessibilité numérique dans la réalisation des projets, produits et services dépendant des TIC (Technologies de l'Information et de la Communication). L'ONU la définit comme un droit universel selon l'article 9 de la Convention relative aux droits des personnes handicapées, adoptée en 2006.

Pourtant, les applications concrètes ne sont pas toujours prises en compte, faute de connaissances, de moyens, ou de temps. Dans le domaine du Web, il appartient à tous de procéder au mieux et de ne pas négliger ce critère de qualité essentiel.

## Qu'est-ce que l'accessibilité du Web ?

Contrairement à beaucoup d'idées reçues, nous sommes tous concernés par l'accessibilité du Web. De façon continue, tout au long de notre vie (parce que nous sommes affectés par un handicap), durant une période déterminée (un bras dans le plâtre), ou à partir d'un certain âge (parce que notre vue baisse). Les exemples sont nombreux et l'on considère que la proportion du public concerné directement par l'accessibilité atteint 15 à 20 % en Europe.

Dans un sens plus général, il s'agit de permettre l'accès aux ressources en ligne, quels que soient le matériel et le réseau utilisés (indirectement liés aux moyens financiers de tout un chacun), quels que soient les aptitudes physiques et mentales, la langue maternelle, l'âge, ou la localisation géographique.

**Figure 3-2**  
Symboles reconnus  
à l'international



Concrètement, une personne non voyante pourra utiliser une synthèse vocale qui lui lira successivement les éléments figurant sur une page web, si le site est bien conçu et ne comporte pas d'erreurs de conception empêchant le logiciel de synthèse de pouvoir comprendre son contenu texte. Une personne handicapée motrice pourra utiliser un périphérique de pointage différent d'une souris, avec un clavier visuel à l'écran et un petit joystick. Il existe des outils de commandes fonctionnant avec les doigts, les paupières, la langue, et même le souffle !

De nombreux hommes et femmes sont dans ce cas et maîtrisent Internet à un haut niveau. Vous pouvez très bien avoir de nombreux échanges avec un non-voyant ou un internaute possédant un handicap moteur, sans que vous puissiez vous apercevoir que votre interlocuteur en ligne utilise une synthèse vocale ou un périphérique d'entrée qui n'est pas un couple souris-clavier.

Les bienfaits d'une prise en compte sérieuse et rationnelle de l'accessibilité d'un site web sont multiples :

- le public pouvant consulter votre site est bien plus large ;
- les personnes handicapées profitent des mêmes services ;
- les moteurs de recherche, dont on dit qu'ils sont les premiers internautes aveugles, comprennent beaucoup mieux le contenu de vos pages et les indexent de façon optimale ;
- l'accès est rendu possible sur des plates-formes techniques limitées ;
- la propriété du code source et sa maintenabilité sont accrues.

Il existe de nombreuses ressources sur le Web pour prendre connaissance des bonnes pratiques et des astuces pour améliorer l'accessibilité d'un site. De plus en plus d'extensions et d'outils voient le jour pour effectuer des diagnostics et des tests.

Les lecteurs d'écran quant à eux évoluent lentement mais sûrement. On peut citer parmi les plus connus VoiceOver (Mac OS X), JAWS, Window-Eyes, ZoomText, NVDA (Windows). Leurs anciennes versions ne sont pas toujours réceptives à ARIA.

Dans tous les cas, on veillera a minima au respect de certaines règles (voir encadré).

HTML 5 est désormais concerné à juste titre, car revêtant un aspect plus dynamique et beaucoup plus riche au travers des API gravitant autour du noyau HTML. Dès lors, il est complexe de pouvoir lire un document de façon linéaire, du début à la fin, alors que des parties dynamiques peuvent changer de contenu à tout moment. Bien qu'un soin certain soit apporté par les rédacteurs des spécifications, rien n'oblige les développeurs web à tout mettre en œuvre pour réussir un site totalement accessible. La diversité des situations et des problématiques pouvant survenir n'ont de limite que l'imagination de tous ceux qui participent à l'élaboration du Web.

## Bonnes pratiques pour un site accessible

Au préalable à toute mise en œuvre technique au niveau du code HTML, des bonnes pratiques existent pour structurer et produire du contenu pour le Web. En voici quelques-unes, ainsi que les cas dans lesquels elles se révèlent salutaires.

- Ne pas baser l'information uniquement sur les couleurs (daltonisme et achromatopsie), par exemple éviter « cliquez sur le bouton rouge ».
- Ne pas baser l'information uniquement sur le son (surdité).
- Permettre l'agrandissement des polices dans le navigateur (myopie, fatigue visuelle).
- Ne pas exiger un court délai de réaction (troubles moteurs), par exemple pour la navigation et les menus déroulants.
- Favoriser le contraste du texte (déficiences visuelles).
- Produire des alternatives telles que l'audiodescription ou la transcription pour des contenus vidéo (surdité partielle ou totale).
- Permettre la navigation au clavier (troubles moteurs) et prévoir des liens d'évitement.
- Ne pas utiliser d'images pour afficher du texte, sans leur adjoindre une alternative texte (attribut `alt`).
- Concevoir une navigation claire et simple (aptitudes mentales, vieillesse).
- Structurer le contenu de façon logique avec une hiérarchie de titres `<hX>` aisément compréhensible.
- Exploiter HTML et CSS pour séparer le fond de la forme, éviter les mises en pages en tableaux, suivre les recommandations du W3C.
- Utiliser des liens d'évitement en début de page pour offrir l'accès direct au contenu, à la navigation, à la recherche.

Flash et Java ont fait également l'objet de débats, car étant embarqués dans de nombreuses pages et faisant appel à une technologie radicalement différente des briques de base du Web, ils ont pu pénaliser la navigation et l'accès (de façon partielle ou totale) à des sites essentiels à tout un chacun. Il existe désormais des techniques, prévues par Adobe ou anticipées par les logiciels d'aide à la navigation, pour améliorer l'accessibilité des animations Flash.

Une démonstration a été publiée en ligne pour prouver que l'on peut rendre un site accessible sans pénaliser son rendu visuel, en respectant les bonnes pratiques WCAG.

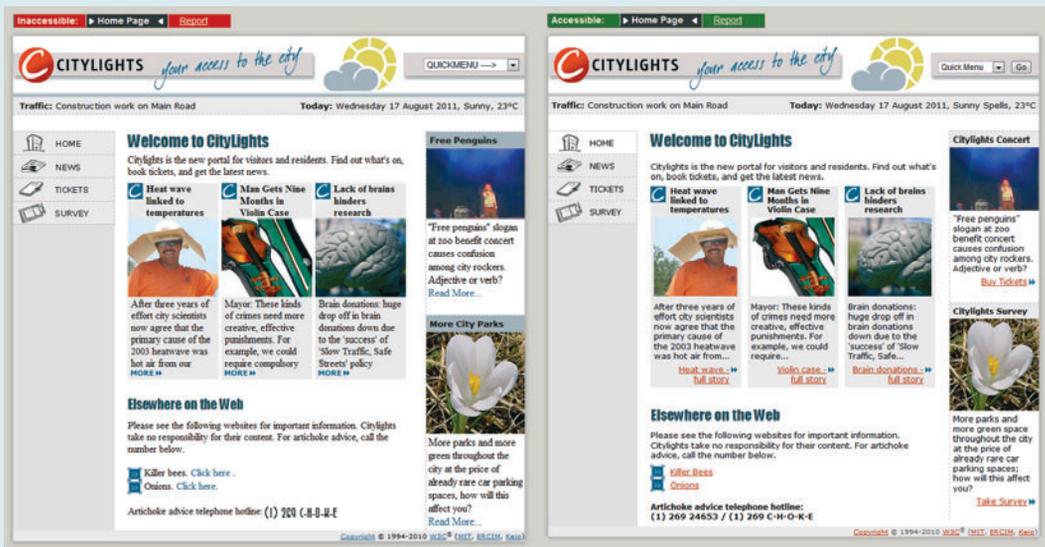


Figure 3-3 Démonstration avant (inaccessible) et après (accessible) <http://www.w3.org/WAI/demos/bad/>

## HTML sémantique

La première bonne pratique en HTML reste celle de respecter la sémantique, et donc d'utiliser les éléments avec ce pour quoi ils ont été prévus. Au début du XXI<sup>e</sup> siècle, beaucoup d'intégrateurs (ainsi qu'Alsacreation.com) se sont fait l'écho d'une conception web conforme aux normes. Ils ont milité pour l'oubli de l'élément `<table>` à des fins de mise en pages, qui était exploité à tort de la sorte. Cet usage détourné avait été initié par le manque de possibilités graphiques des anciens navigateurs, un besoin d'aller plus loin dans le positionnement, et par une certaine ignorance généralisée des bienfaits des feuilles de style CSS.

Pourtant, cette philosophie n'a pas toujours été comprise. Certains ont cru qu'il suffisait de remplacer les `<table>` par des `<div>` pour créer un site propre. Or, l'élément `<table>` n'est pas le grand méchant loup, et est tout à fait justifié dans le cas de données tabulaires. En premier lieu, il faut donc surtout éviter le syndrome de la « divite », c'est-à-dire de considérer `<div>` comme un élément bon à tout faire, et ne l'exploiter qu'en dernier recours lorsqu'aucun autre élément n'existe. Par exemple, il est tout à fait logique qu'un paragraphe de texte soit contenu entre les balises `<p>` et `</p>` qui sont dévolues à cet usage. Il est inutile de lui préférer `<div>` qui n'a aucune valeur sémantique, et qui serait potentiellement associé à des propriétés CSS pour lui conférer l'apparence d'un paragraphe.

Avec HTML 5, les nouvelles balises (par exemple `<article>` et `<nav>`) sont une avancée indéniable en ce sens puisqu'elles facilitent l'interprétation du document, la lisibilité du code par un humain ou une machine, le référencement et l'accessibilité de façon globale. D'autres nouveautés permettent en théorie d'éviter les surcouches JavaScript ou Flash pour les éléments médias (avec `<audio>`, `<video>`) et pour les contrôles de formulaires (avec les nouveaux types et attributs pour `<input>`). Dans la pratique, de nombreux progrès sont encore attendus, car la navigation au clavier dans ces médias n'est pas toujours parfaite.

Leur interprétation par les agents utilisateurs et les logiciels d'assistance a néanmoins été sujette à des bogues et soumise aux « correctifs JavaScript » sur les anciennes versions d'Internet Explorer, ce qui les rend invisibles sans JavaScript jusqu'à la version 8 incluse. La navigation au clavier pour les éléments médias est imparfaite voire inexistante dans les versions des navigateurs qui n'ont qu'un vécu récent avec HTML 5. La transcription texte des fichiers audio et vidéo nécessite encore de grands progrès, car le support de `<track>` reste limité pour ne pas dire parcellaire au moment de la rédaction de cet ouvrage.

En ce qui concerne Canvas, le bilan est plus sombre car cette grille de pixels purement graphique est bien complexe à « accessibiliser ». D'ailleurs, le faut-il vraiment ? Un texte alternatif peut-il faire l'affaire ? Tout dépend de son usage. Canvas sera probablement complété par un DOM ou une surcouche d'accessibilité. Cependant, seront-ils vraiment utilisés ? SVG ne présente pas cet inconvénient, à la condition que le créateur prenne en compte l'accessibilité dès le départ.

Malgré tout, les spécifications HTML sont conçues de façon à ce que tous les éléments et attributs possèdent un rôle mûrement réfléchi, garantissant que son contenu pourra être interprété à bon escient par un agent utilisateur bien conçu. Du point de vue du code, quelques réflexes ne sont jamais superflus.

Pour les tableaux de données :

- Utiliser l'attribut `summary` sur `<table>`.
- Utiliser les attributs `scope` et `headers` sur `<td>` et `<th>`.

Pour les formulaires :

- Utiliser `<fieldset>` et `<legend>`.
- Associer une étiquette `<label>` à tout champ d'entrée `<input>`, `<textarea>`, etc.

Pour le contenu :

- Compléter l'attribut `alt` pour les `<img>` (non décoratives).
- Utiliser un titre pertinent pour la balise de document `<title>`.
- Utiliser une hiérarchie de titres `<h1>` à `<h6>` bien structurée.
- Employer des listes `<ul>`, `<ol>` pour toute énumération.

Ces exemples ne sont qu'un échantillon de ce qu'il est possible de faire pour améliorer considérablement l'accessibilité d'un site, au sens large du terme. Pour épauler HTML et prêcher la bonne parole, il est nécessaire de faire un tour du côté de la WAI, qui n'est pas ce geste très gracieux permettant de saluer en Thaïlande en rapprochant les deux mains de son visage, mais qui mérite tout autant d'attention.

## WAI, WCAG et ARIA

La *Web Accessibility Initiative* est issue du W3C. Dès 1997, s'est fait ressentir le besoin d'établir des recommandations et de développer des outils pour améliorer les technologies du Web. Plusieurs guides ont vu le jour depuis, notamment :

- Les WCAG 1.0 (1999) et WCAG 2.0 (2008) – *Web Content Accessibility Guidelines* – qui sont un ensemble de bonnes pratiques pour le contenu web, destinées aux personnes possédant une vision déficiente, une perte de capacité auditive, un handicap physique ou une déficience cognitive.
- Les ATAG 1.0 (2000) et ATAG 2.0 (2011) – *Authoring Tool Accessibility Guidelines* – ont pour cible les développeurs de logiciels pour l'édition et la création de ressources web, afin de les inciter à produire du contenu accessible respectant au mieux les règles WCAG.
- Les UAAG 1.0 (2002) – *User Agent Accessibility Guidelines* – se concentrent sur les développeurs de navigateurs et de logiciels équivalents permettant la navigation et la consultation web.

- WAI-ARIA – Accessible Rich Internet Applications – définit des méthodes pour améliorer l'accessibilité des applications web en général, notamment lorsqu'il s'agit de faire appel à des comportements dynamiques avec JavaScript et Ajax.

**Figure 3–4**  
Web Accessibility  
Initiative (WAI)

The screenshot shows the WAI website homepage. At the top, there's a header with the W3C logo and the text 'Web Accessibility Initiative' and 'WAI: Strategies, guidelines, resources to make the Web accessible to people with disabilities'. Below the header, there's a navigation menu on the left with links like 'Getting Started', 'Guidelines & Techniques', etc. The main content area is divided into sections: 'Web Accessibility Initiative (WAI) Home', 'Highlights', and 'For Review: ATAG 2.0 and Implementing ATAG 2.0 Working Drafts'. The right sidebar contains sections for 'WAI develops...', 'WAI welcomes...', 'Announcements', and 'Events, Meetings, Presentations'.

En tant que développeur web ou intégrateur, vous êtes directement concerné par WCAG et ARIA. En premier lieu parce que WCAG permet de prendre connaissance des bonnes (et donc des mauvaises) pratiques pour la mise en ligne de contenu web conventionnel, et de respecter différents niveaux de qualité à atteindre (A, AA, AAA) selon les moyens techniques mis à contribution. En second lieu parce que WAI-ARIA porte spécifiquement sur les technologies qui rendent aujourd'hui le Web attractif, entre autres HTML 5 et Ajax pour les applications web que l'on qualifie de « riches » par distinction avec les documents « statiques ».

#### RESSOURCE Recommandations WAI

##### WCAG 2.0

- ▶ <http://www.w3.org/TR/WCAG20/>

##### Techniques pour WCAG 2.0

- ▶ <http://www.w3.org/TR/WCAG20-TECHS/>

##### Mieux comprendre les critères WCAG 2.0

- ▶ <http://www.w3.org/TR/UNDERSTANDING-WCAG20/>

##### Point sur le support de HTML 5 et de son accessibilité

- ▶ <http://www.html5accessibility.com/>

WAI-ARIA a atteint le stade de recommandation candidate le 18 janvier 2011, mais son implémentation est antérieure. Bien que dépendante de la bonne volonté des développeurs d'agents utilisateurs (les navigateurs), elle progresse de façon certaine et l'initiative est soutenue par de grands noms tels qu'IBM, Mozilla, Dojo, jQuery, Microsoft, Google, Opera, Yahoo, Adobe, Apple, Sun.

Les techniques offertes par WAI-ARIA visent à décrire l'état et le comportement d'une application web. Il s'agit par exemple de savoir si un menu est développé, ou si un élément HTML quelconque possède un rôle particulier (et dynamique) au sein du document. Beaucoup d'agréments dynamiques sont développés en JavaScript avec de belles images et font face à deux problèmes : aucune navigation au clavier n'est habituellement prévue pour les contrôler, et les technologies d'assistance ne savent comment les retranscrire (vocalement ou avec une plage braille).

Les applications web sont bien souvent dopées à l'Ajax pour effectuer des appels au serveur HTTP en tâche de fond et mettre à jour des contenus sans recharger la totalité de la page. Cette méthode peut également passer inaperçue pour les technologies d'assistance, à moins qu'elles ne soient notifiées d'un changement et que l'utilisateur puisse être informé de la nature de la mise à jour du DOM et de son emplacement.

Sous Windows, la plupart des navigateurs modernes transmettent ces informations ARIA, incluses dans le code HTML, à l'API MSAA (*Microsoft Active Accessibility*) existant depuis Windows 98. Cette dernière est alors mise à disposition des logiciels d'aide à la navigation pour décrire le contenu d'une façon alternative, par exemple avec une synthèse vocale. Sous Linux on peut citer Gnome ATK/AT-SPI.

La mise en œuvre de WAI-ARIA consiste en l'ajout de quelques attributs au code HTML pour étendre sa valeur sémantique. Le code n'en est pas alourdi. L'effort reste donc minime et toujours bénéfique dans le cas du développement d'applications web « riches ». En théorie, ces techniques ne se limitent pas à HTML et peuvent être intégrées à d'autres langages, c'est d'ailleurs le cas de SVG.

On peut considérer que le début de prise en compte a eu lieu avec Internet Explorer 8, Firefox 1.5 (plus sérieusement avec la version 3), Opera 9.5, Safari 4 et Google Chrome 2. Le niveau de support est cependant différent et croissant au fil du temps.

Les anciens navigateurs qui ne supportent pas ARIA, ou les récents qui ne désirent pas l'utiliser, ignorent tout simplement les attributs HTML qui y font référence, et ne sont en aucun cas gênés. Les frameworks JavaScript les plus célèbres (entre autres jQuery et Dojo) sont déjà férus d'ARIA et font constamment des progrès en ce sens.

En attendant un support parfait de HTML 5 par l'ensemble des navigateurs, qui est certainement une douce utopie, WAI-ARIA apporte du concret et reste certainement la meilleure initiative à considérer, applicable immédiatement.

**RESSOURCE Tout sur WAI-ARIA**

Reportez-vous à la spécification WAI-ARIA elle-même, ainsi qu'à une introduction critique et un guide concret :

- ▶ <http://www.w3.org/TR/wai-aria/>
- ▶ <http://www.w3.org/TR/wai-aria-primer/>
- ▶ <http://www.w3.org/TR/wai-aria-practices/>

## Les rôles et propriétés de WAI-ARIA

Le groupe du WAI préconise l'utilisation du terme complet « WAI-ARIA » pour ne pas confondre cette technologie avec un passage d'opéra (et l'on parle bien ici du genre musical, pas du navigateur éponyme). Les exemples suivants en font honteusement abstraction, par commodité de lecture.

Quels sont les ajouts définis par ARIA ?

- Des rôles pour décrire un type de composant de l'interface graphique (par exemple un menu, un bouton, une barre de progression...).
- Des rôles pour décrire la structure générale du document (par exemple ses régions, sa titraille, etc.).
- Des états pour les composants interactifs du document (par exemple « coché » pour une case à cocher, ou « déployé » pour un menu).
- Des propriétés pour décrire le drag & drop (glisser-déposer).
- Des propriétés pour définir quelles sont les parties du document qui sont susceptibles d'être mises à jour dynamiquement, sans rechargement complet (JavaScript et Ajax), avec des politiques de gestion et de notification de ces mises à jour.
- Une manière de naviguer à travers cet ensemble au clavier : se reporter pour cela à la description de l'attribut `tabindex`. Doter un élément quelconque d'un `tabindex="0"` permet d'y établir le focus au clavier, tandis qu'un `tabindex="-1"` autorise le focus à la souris ou avec un script sans l'ajouter à l'ordre des tabulations. Ce concept est destiné au développement de composants d'interface (ou *widgets*) dotés de navigation au travers de leurs descendants. Il est épaulé par `aria-activedescendant` qui indique pour un parent donné quel descendant reçoit le focus.

L'affectation des rôles et des propriétés passe par deux familles d'attributs en HTML 5 qui sont intégrées nativement à la spécification et donc implicitement valides (contrairement à l'ajout d'attributs ARIA en XHTML 1.0).

La première famille est constituée d'un seul membre : **role** pour l'assignation d'un rôle précis à un élément. Sa valeur est issue d'une liste de rôles définis par la spécification ARIA.

La deuxième famille est une série d'attributs débutant par **aria-** et suivis par un nom de propriété ou d'état relatif à l'élément. Leur valeur est aussi définie par la spécification parmi un ensemble de valeurs.

#### RESSOURCE Rôles et propriétés ARIA

Tous les rôles

▶ <http://www.w3.org/WAI/PF/aria/roles>

Toutes les propriétés et états

▶ [http://www.w3.org/WAI/PF/aria/states\\_and\\_properties](http://www.w3.org/WAI/PF/aria/states_and_properties)

Attributs ARIA et valeurs du point de vue de HTML 5

▶ <http://dev.w3.org/html5/markup/aria/aria.html>

## Rôles avec l'attribut role

Les rôles ARIA peuvent être classés en plusieurs familles. Il existe des rôles abstraits desquels ces familles peuvent hériter de propriétés communes, mais ils ne seront pas décrits, car ils ne sont pas directement exploitables.

### Points de repère (landmark roles)

Les « points de repère » (ou *Landmark Roles*) délimitent les grandes zones du document ou de l'application web.

Tableau 3-1 Rôles ARIA « Points de repère »

Rôle	Description
<a href="#">application</a>	Région correspondant à une application web (à l'inverse d'un document statique).
<a href="#">banner</a>	Typiquement, l'en-tête comprenant un logo et éventuellement un outil de recherche.
<a href="#">complementary</a>	Section complémentaire à la section principale (main), de même niveau, qui peut rester pertinente lorsqu'elle en est détachée.
<a href="#">contentinfo</a>	Région contenant des informations relatives au document.
<a href="#">form</a>	Région contenant des objets qui dans leur ensemble constituent un formulaire.
<a href="#">main</a>	Contenu principal.
<a href="#">navigation</a>	Ensemble de liens de navigation.
<a href="#">search</a>	Une région contenant des objets qui dans leur ensemble constituent un outil de recherche.

Ces repères dans le document autorisent l'utilisateur à naviguer de l'un à l'autre, souvent à l'aide de raccourcis clavier, et à en appréhender de façon plus naturelle la structure générale. Ils sont supportés par JAWS 10+, NVDA 2010+ et VoiceOver..

### Application de rôles ARIA

```
<header role="banner">
  <!-- En-tête et titre principal -->
  <form role="search">
    <label for="q">Mot clé : </label>
    <input type="search" name="q" id="q">
    <input type="submit" value="Lancer la recherche">
  </form>
</header>
<nav role="navigation">
  <!-- Liens de navigation -->
</nav>
<section role="main">
  <!-- Contenu principal -->
</section>
```

Plusieurs éléments HTML 5 possèdent déjà un rôle WAI-ARIA natif (par exemple les éléments de formulaire `input`, les éléments sémantiques comme `<nav>`). Certains d'entre eux peuvent être « surchargés », par exemple `<a>` qui est affublé d'un rôle ARIA `link`, et qui est amené à répondre à d'autres usages que celui d'un lien classique. D'autres éléments en sont totalement dépourvus, notamment les éléments neutres `<div>`, `<span>`, mais peuvent en être équipés. L'essentiel est de ne jamais aller à l'encontre des rôles natifs, ce qui pourrait dérouter ou inhiber les agents utilisateurs. Leur liste complète est présente dans la spécification HTML 5, ainsi que les restrictions en vigueur.

Ainsi les éléments `<article>`, `<aside>`, `<section>`, `<hr>` possèdent respectivement les rôles par défaut `article`, `note`, `region` et `separator`. Si le concepteur choisit de leur affecter un autre rôle, il doit piocher parmi `article`, `document`, `application`, ou `main` pour `<article>` ; `note`, `complementary` ou `search` pour `<aside>` ; et `region`, `document`, `application`, `contentinfo`, `main`, `search`, `alert`, `dialog`, `alertdialog`, `status` ou `log` pour `<section>`.

Pour les listes, `<ul>` et `<ol>` possèdent le rôle implicite `list`, tandis que leurs enfants `<li>` sont catégorisés en `listitem`.

#### RESSOURCE Rôles et propriétés ARIA

Valeurs appliquées implicitement aux éléments en HTML 5

► <http://www.w3.org/TR/html5/content-models.html#annotations-for-assistive-technology-products-aria>

Les éléments de section HTML 5 peuvent sembler redondants avec les *landmarks*. N'est-ce d'ailleurs pas l'objet des nouvelles balises telles que `<header>`, `<nav>` et leurs acolytes de posséder une valeur sémantique ? Lorsque toutes les technologies d'assistance supporteront avec perfection HTML 5 et par exemple efficacement le rôle de navigation véhiculé par `<nav>`, on pourra présumer que l'attribut `role="navigation"` ne sera plus nécessaire. Cependant, ce n'est pas encore le cas, c'est pourquoi il est plus sage de compléter la structure du document avec ces points de repère, et de ne pas supposer qu'ils seront « devinés ».

## Structure de document

En quittant la vision satellite globale, et en traversant les premières couches de nuages pour se rapprocher des zones composant chaque région majeure, les rôles structurant le contenu font leur apparition.

Tableau 3-2 Rôles ARIA « Structure de document »

Rôle	Description
<code>article</code>	Un article formant une partie indépendante du document.
<code>columnheader</code>	Un en-tête de colonne.
<code>definition</code>	Une définition.
<code>directory</code>	Une liste de références aux membres d'un groupe (une table des matières).
<code>document</code>	Une région correspondant à un document (à l'inverse d'une application).
<code>group</code>	Un ensemble d'objets qui ne sont pas destinés à être inclus dans un résumé du document ou dans sa table des matières.
<code>heading</code>	Un en-tête de section.
<code>img</code>	Un conteneur pour un ensemble d'éléments qui forment une image.
<code>list</code>	Une liste d'éléments non interactifs.
<code>listitem</code>	Un élément dans une liste ou une table des matières ( <i>directory</i> ).
<code>math</code>	Une expression mathématique.
<code>note</code>	Une note annexe au contenu principal.
<code>presentation</code>	Un élément de présentation dont les rôles implicites ne seront pas relayés à l'API d'accessibilité.
<code>region</code>	Une grande partie du document assez importante pour être incluse dans le résumé ou la table des matières.
<code>row</code>	Une ligne de cellules dans une grille.
<code>rowheader</code>	Un en-tête de ligne de cellules dans une grille.
<code>separator</code>	Un séparateur de contenu ou de groupes d'éléments de menus.
<code>toolbar</code>	Une collection de contrôles fréquemment utilisés représentés sous une forme compacte.

Grâce à ces rôles et à quelques propriétés (décrites ultérieurement), le balisage complémentaire est très aisé.

### Titrage ARIA

```
<section role="search" aria-labelledby="entete">
  <h1 id="entete" role="heading" aria-level="1">
    Recherche dans les archives
  </h1>
  <!-- Formulaire ... -->
</section>
```

Dans cette situation interviennent également les rôles implicites prévus par HTML 5. Pour ne citer que deux d'entre eux, `columnheader` et `rowheader` correspondent respectivement à des en-têtes de tableau, c'est-à-dire dans la pratique à un élément `<th>`, auquel on aurait affecté l'attribut `scope="col"` ou `scope="row"`.

### Composants graphiques (widgets)

Les rôles pour composants graphiques (ou *widgets*) décrivent des éléments majoritairement interactifs pour l'utilisateur. On y retrouve une panoplie complète de menus, boutons, champs de formulaire, et autres objets qui existent depuis que l'interface graphique utilisateur (en version originale *GUI*) a remplacé l'austère ligne de commande.

**Tableau 3-3** Rôles ARIA « Composants graphiques »

Rôle	Description
<code>alert</code>	Un message d'alerte.
<code>alertdialog</code>	Un dialogue contenant un message d'alerte.
<code>button</code>	Un contrôle d'entrée (un bouton) destiné à déclencher des actions lorsqu'il est cliqué ou pressé.
<code>checkbox</code>	Une case à cocher possédant trois états : vrai, faux ou mixte.
<code>combobox</code>	Une liste de choix (conteneur).
<code>dialog</code>	Une fenêtre de dialogue, usuellement modale, demandant un choix ou une information à l'utilisateur.
<code>grid</code>	Une grille (un tableau) pouvant contenir des cellules organisées en colonnes et en lignes (conteneur).
<code>gridcell</code>	Une cellule appartenant à une grille.
<code>link</code>	Une référence à une ressource externe qui provoque la navigation vers cette ressource lorsqu'elle est activée.
<code>listbox</code>	Une liste déroulante autorisant l'utilisateur à choisir un ou plusieurs éléments parmi une liste de choix (conteneur).

Tableau 3-3 Rôles ARIA « Composants graphiques » (suite)

Rôle	Description
<code>log</code>	Une région contenant des informations ajoutées au fur et à mesure dont les plus anciennes peuvent disparaître.
<code>marquee</code>	Une région dynamique contenant des informations non essentielles qui changent fréquemment.
<code>menu</code>	Un contrôle offrant une liste de choix à l'utilisateur (conteneur).
<code>menubar</code>	Un menu qui est destiné à rester visible et qui est usuellement présenté à l'horizontale (conteneur).
<code>menuitem</code>	Une option parmi un groupe contenu dans menu ou menubar.
<code>menuitemcheckbox</code>	Un élément de menu pouvant être coché et possédant trois états : vrai, faux, mixte.
<code>menuitemradio</code>	Un élément de menu parmi un groupe d'autres congénères, dont un seul peut être coché à la fois.
<code>option</code>	Un élément sélectionnable dans une liste.
<code>progressbar</code>	Une barre de progression pour les tâches qui peuvent nécessiter un certain temps.
<code>radiogroup</code>	Un groupe de boutons radio (conteneur).
<code>radio</code>	Un élément de bouton radio parmi un groupe d'autres congénères, dont un seul peut être coché à la fois.
<code>scrollbar</code>	Un contrôle graphique permettant de faire défiler le contenu dans la zone visible
<code>slider</code>	Un contrôle permettant à l'utilisateur de sélectionner une valeur parmi un intervalle donné.
<code>spinbutton</code>	Un contrôle permettant à l'utilisateur de sélectionner une valeur à intervalles réguliers, parmi un intervalle donné.
<code>status</code>	Un conteneur recueillant une information d'état dont l'importance ne justifie pas l'emploi d'une alerte.
<code>tab</code>	Un onglet parmi un ensemble.
<code>tablist</code>	Une liste d'onglets ( <code>tab</code> ) qui sont des références respectives à des conteneurs ( <code>tabpanel</code> ).
<code>tabpanel</code>	Un conteneur pour les ressources associées à un onglet.
<code>textbox</code>	Un champ d'entrée texte libre.
<code>timer</code>	Un conteneur doté d'un compteur numérique indiquant la quantité de temps écoulé depuis une date, ou restant jusqu'à une date fixée.
<code>tooltip</code>	Un pop-up contextuel affichant la description d'un autre élément.
<code>tree</code>	Une liste contenant une arborescence de groupes pouvant être réduits et redéployés (conteneur).
<code>treegrid</code>	Une grille dont les lignes peuvent être réduites et redéployées (conteneur).
<code>treeitem</code>	Un élément d'arborescence, pouvant en contenir d'autres.

Étant donné la diversité des situations, il est complexe de dresser une liste de pratiques types pouvant répondre aux besoins courants de l'interface utilisateur. Se concentrer sur quelques extraits permet d'appréhender la philosophie globale des rôles que l'on retrouve en général sur les feuilles de l'arbre DOM.

#### Exemple de menu avec trois actions

```
<ul role="menu">
  <li role="menuitem">Nouveau document</li>
  <li role="menuitem">Ouvrir un document</li>
  <li role="menuitem">Fermer</li>
</ul>
```

Ces actions peuvent ensuite être implémentées en JavaScript.

#### Groupe de boutons radio

```
<ul role="radiogroup">
  <li role="radio"><!-- Bouton radio 1 --></li>
  <li role="radio"><!-- Bouton radio 2 --></li>
  <li role="radio"><!-- Bouton radio 3 --></li>
</ul>
```

#### Barre d'outils

```
<div role="toolbar" tabindex="0" aria-activedescendant="btn1">
  
  
  
</div>
```

ARIA autorise le « détournement » d'éléments, puisque cette technologie permet justement d'affecter des rôles précis à des éléments qui n'en ont pas.

#### Span détourné en case à cocher

```
<span tabindex="0" role="checkbox" aria-checked="true"
  onkeydown="return gestionEvenement(event);"
  onclick="return gestionEvenement(event);">
  Je souhaite être notifié des nouveaux commentaires
</span>
```

Dans cet exemple, la possibilité d'obtenir le focus au clavier est donnée par l'attribut `tabindex`. Son rôle est celui d'une case à cocher (`role="checkbox"`), dont l'état par défaut est coché (`aria-checked="true"`). Le texte contenu par l'élément lui sert

d'étiquette. Les attributs `onkeydown` et `onclick` associent des gestionnaires d'événement en JavaScript qui doivent agir en conséquence lorsque l'utilisateur clique ou utilise une touche de son clavier sur cet élément, notamment la barre d'espace qui doit changer l'état de la case. Une technologie d'assistance pourra donc tout à fait interpréter l'ensemble de façon transparente, en tant que case à cocher conventionnelle. Bien entendu, ce type de solution est à réserver lorsqu'il n'a pas été possible d'utiliser les éléments HTML prévus à cet effet.

## Propriétés et états avec les attributs `aria-*`

Les propriétés et états ARIA renseignent sur les caractéristiques intrinsèques d'un élément. Il s'agit de fonctionnalités similaires qui sont très proches d'un point de vue technique puisqu'elles sont toutes mises en œuvre avec des attributs débutant par le préfixe `aria-`.

### Application d'une propriété ARIA

```
<div aria-live="polite">
  <!-- ... -->
</div>
```

Leur différence réside principalement dans le fait que les valeurs des états sont amenées à évoluer au cours du temps, tandis que celles des propriétés restent relativement stables au cours de la navigation dans le document.

Par exemple, la propriété `aria-required` définissant le caractère requis d'un champ d'entrée dans un formulaire devrait a priori rester constante au cours du temps.

### Application d'une propriété ARIA

```
<input type="text" role="textbox" aria-required="true">
```

En revanche, l'état `aria-checked` définissant le statut « coché » ou « décoché » pourra changer après un événement utilisateur, navigateur ou serveur (en Ajax). Il en sera de même pour `aria-valuenow` qui exprime la valeur d'un élément de formulaire à un moment donné.

### Modification dynamique d'une propriété ARIA

```
<!-- « Glisseur » ou potentiomètre linéaire -->
<div class="slidercontrol">
  
```

```

</div>

<script>
// Fonction
function sliderSet(valeur) {
    var curseur = document.getElementById('slider');
    // Définition des attributs aria-*
    curseur.setAttribute('aria-valuenow', valeur);
    curseur.setAttribute('aria-valuetext', valeur+" %");
}

// Position par défaut
sliderSet(50);

// Autres actions éventuelles pour gérer le contrôle graphique à la
// souris et l'appel à la fonction sliderSet
</script>

```

Dans cet exemple, une image sert de curseur et une fonction JavaScript permet de modifier dynamiquement la valeur transmise par ARIA aux technologies d'assistance en jouant sur les attributs `aria-valuenow` et `aria-valuetext`. En bonus, la propriété `aria-orientation` lui conférerait une orientation (verticale ou horizontale).

Il aurait tout aussi bien été possible de remplacer l'image par un des nouveaux types HTML 5 pour la balise `<input>`. L'usage d'ARIA avec cet élément ne ferait alors pas double emploi.

```
<input type="range" min="0" max="100" value="50">
```

Hormis la distinction existant entre les propriétés et les états, leur implémentation dans le code source HTML reste semblable. Certains d'entre eux fonctionnent de concert avec les rôles et n'auront de signification que pour un rôle donné. L'état `aria-pressed` ne sera pertinent qu'avec un rôle `button`.

## Globaux

ARIA définit en premier lieu des attributs globaux, applicables à tous les éléments.

**Tableau 3-4** Attributs ARIA globaux

État ou propriété	Description
<code>aria-atomic</code>	Voir "live"
<code>aria-busy</code>	Voir "live"
<code>aria-controls</code>	Voir "relations"
<code>aria-describedby</code>	Voir "relations"

**Tableau 3-4** Attributs ARIA globaux (suite)

État ou propriété	Description
<code>aria-disabled</code>	Voir "contrôles d'interface"
<code>aria-dropeffect</code>	Voir "drag & drop"
<code>aria-flowto</code>	Voir "relations"
<code>aria-grabbed</code>	Voir "drag & drop"
<code>aria-haspopup</code>	Voir "contrôles d'interface"
<code>aria-hidden</code>	Voir "contrôles d'interface"
<code>aria-invalid</code>	Voir "contrôles d'interface"
<code>aria-label</code>	Voir "contrôles d'interface"
<code>aria-labelledby</code>	Voir "relations"
<code>aria-live</code>	Voir "live"
<code>aria-owns</code>	Voir "relations"
<code>aria-relevant</code>	Voir "live"

Tous ces attributs sont décrits dans chaque section spécifique ci-après.

## Contrôles d'interface

Des attributs spécifiques sont applicables aux composants graphiques interactifs (ou *widgets*), que l'on représente habituellement par des contrôles formant ensemble une application ou un formulaire avec lequel l'utilisateur peut interagir.

**Tableau 3-5** Attributs ARIA pour composants « graphiques »

État ou propriété	Description	Valeurs	Rôles
<code>aria-autocomplete</code>	Indique que des suggestions d'autocomplétion sont fournies à l'utilisateur (en ligne dans le champ, en liste, les deux, ou aucune).	<code>inline</code> , <code>list</code> , <code>both</code> , <code>none</code>	<code>combobox</code> <code>textbox</code>
<code>aria-checked</code>	Statut coché ou décoché (état).	<code>true</code> , <code>false</code> , <code>mixed</code> , <code>undefined</code>	<code>option</code>
<code>aria-disabled</code>	Indique que l'élément est perceptible mais désactivé (état).	<code>true</code> , <code>false</code>	Tous

**Tableau 3-5** Attributs ARIA pour composants « graphiques » (suite)

État ou propriété	Description	Valeurs	Rôles
<code>aria-expanded</code>	Indique que l'élément est déployé ou réduit (état).	<code>true</code> , <code>false</code> , <code>undefined</code>	<code>button</code> <code>document</code> <code>link</code> <code>section</code> <code>sectionhead</code> <code>separator</code> <code>window</code>
<code>aria-haspopup</code>	Indique que l'élément possède un menu contextuel.	<code>true</code> , <code>false</code>	Tous
<code>aria-hidden</code>	Indique que l'élément n'est pas perceptible pour l'utilisateur (état).	<code>true</code> , <code>false</code>	Tous
<code>aria-invalid</code>	Indique que la valeur entrée ne correspond pas à ce qui est attendu (état).	<code>grammar</code> , <code>false</code> , <code>spelling</code> , <code>true</code>	Tous
<code>aria-label</code>	Étiquette associée à l'élément fournissant un nom reconnaissable pour l'utilisateur.	Texte	Tous
<code>aria-level</code>	Définit le niveau d'un élément au sein de sa structure, par exemple dans les arborescences ( <code>tree</code> ) ou les imbrications.	Nombre entier supérieur ou égal à 1	<code>grid</code> <code>heading</code> <code>listitem</code> <code>row</code> <code>tablist</code>
<code>aria-multiline</code>	Spécifie que le champ d'entrée texte accepte (ou non) plusieurs lignes.	<code>true</code> , <code>false</code>	<code>textbox</code>
<code>aria-multiselectable</code>	Spécifie que l'utilisateur peut sélectionner (ou non) plusieurs valeurs parmi un ensemble.	<code>true</code> , <code>false</code>	<code>grid</code> <code>listbox</code> <code>tree</code>
<code>aria-orientation</code>	Indique l'orientation d'un élément, tel qu'une barre de défilement.	<code>vertical</code> , <code>horizontal</code>	<code>scrollbar</code> <code>separator</code> <code>slider</code>
<code>aria-pressed</code>	Statut « pressé » d'un bouton (état).	<code>true</code> , <code>false</code> , <code>mixed</code> , <code>undefined</code>	<code>button</code>
<code>aria-readonly</code>	Indique que l'élément est utilisable, mais qu'il n'est pas éditable.	<code>true</code> , <code>false</code>	<code>grid</code> <code>gridcell</code> <code>textbox</code>

Tableau 3-5 Attributs ARIA pour composants « graphiques » (suite)

État ou propriété	Description	Valeurs	Rôles
<code>aria-required</code>	Indique que l'élément est requis pour la validation du formulaire.	<code>true</code> , <code>false</code>	combobox gridcell listbox radiogroup spinbutton textbox tree
<code>aria-selected</code>	Spécifie que l'élément est sélectionné (état).	<code>true</code> , <code>false</code> , undefined	gridcell option row tab
<code>aria-sort</code>	Indique l'ordre de tri des éléments dans une grille.	ascending, descending, none, other	columnhead er rowheader
<code>aria-valuemax</code>	Spécifie la valeur maximale d'un contrôle d'intervalle.	Nombre	range
<code>aria-valuemin</code>	Spécifie la valeur minimale d'un contrôle d'intervalle.	Nombre	range
<code>aria-valuenow</code>	Spécifie la valeur actuelle d'un contrôle d'intervalle.	Nombre	range
<code>aria-valuetext</code>	Spécifie la valeur texte alternative d' <code>aria-valuenow</code> lisible par un humain.	Texte	range

Ces attributs sont relativement parlants (si l'on ose dire) et s'attarder sur chacun d'entre eux mériterait d'y consacrer un ouvrage complet, tant la multiplicité des situations est grande.

En se concentrant sur `aria-checked`, il est très facile d'imaginer une liste dans laquelle l'utilisateur pourrait « cocher » des choix, sans case à cocher de type `<input type="checkbox">`.

#### Exemple d'application de propriétés et rôles ARIA

```
<ul role="menubar" controls="contenu">
<li role="menuitem" aria-haspopup="true">
  Affichage
  <ul role="menu" aria-hidden="true" hidden>
    <li role="menuitemcheckbox" aria-checked="true">
      
      Trier par ordre alphabétique
    </li>
    <li role="menuitemcheckbox" aria-checked="false">
      Trier par ordre chronologique
    </li>
  </ul>
</li>
```

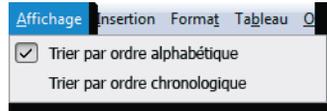
```

    </ul>
  </li>
</ul>

<table id="contenu" role="grid" summary="Liste des articles">
<!-- Contenu du tableau-grille... -->
</table>

```

**Figure 3–5**  
Rendu graphique possible



Nous avons affaire ici à :

- Un menu racine de rôle `menubar`, dont on sait qu'il contrôle un élément possédant l'identifiant `contenu`, c'est-à-dire la grille de données située ultérieurement dans le code source.
- Ce menu racine contient un élément de menu nommé « Affichage » de rôle `menuItem`, et dont on sait qu'il possède un pop-up associé.
- Un sous-menu de rôle `menu`, caché par défaut grâce à l'état `aria-hidden` et l'attribut HTML 5 `hidden`.
- Deux éléments de liste dont le rôle est `menuItemcheckbox`, c'est-à-dire comme faisant partie d'un menu, pouvant être cochés individuellement, et dont l'un d'entre eux est noté comme l'étant effectivement avec `aria-checked="true"`.
- Une image symbolisant l'état de façon graphique, dont le rôle est `presentation`, c'est-à-dire comme devant être ignoré par toute assistance à la navigation ou synthèse vocale.
- Un texte décrivant l'action déclenchée par l'ensemble de cet élément de contrôle.

Des instructions supplémentaires sont nécessaires pour gérer le changement d'état de l'image, que ce soit avec CSS ou avec JavaScript, et pour implémenter l'effet attendu.

## Il y a de la vie sur cette planète

Ainsi que le disait Luke Skywalker, la vie peut se cacher là où on ne l'attend pas. Il en va de même pour les pages web dont le contenu peut être vivant et changer à l'insu de l'utilisateur naviguant sans rendu visuel. En effet, un outil de synthèse vocale procédera à la lecture linéaire du document sans savoir si une partie a été modifiée, avant ou après l'endroit où se trouve son « curseur » de lecture et s'il doit le notifier.

ARIA prévoit de marquer les portions de document, dont le contenu peut être amené à changer au cours du temps de façon dynamique, par les attributs suivants.

Tableau 3-6 Attributs ARIA pour régions « live »

État ou propriété	Description	Valeurs
<code>aria-atomic</code>	Indique la portée de l'information retournée à l'utilisateur lorsque le contenu d'un élément est modifié.	<code>true</code> , <code>false</code>
<code>aria-busy</code>	Indique qu'un élément et ses descendants sont en cours de mise à jour (état).	<code>true</code> , <code>false</code>
<code>aria-live</code>	Indique qu'un élément pourra être mis à jour et décrit le type de mises à jour que les agents utilisateurs peuvent attendre.	<code>off</code> , <code>polite</code> , <code>asservitive</code>
<code>aria-relevant</code>	Indique quelles modifications pourront être opérées.	Une combinaison parmi les termes clés : <code>additions</code> , <code>removals</code> , <code>text</code> , <code>all</code>

Les valeurs pour `aria-live` sont :

- `off` : mise à jour non présentée à l'utilisateur, à moins qu'il n'ait déjà le focus sur cette région.
- `polite` : changement en tâche de fond, l'assistant à la navigation peut l'annoncer au moment qu'il juge opportun (par exemple, à la fin de la lecture d'une phrase ou lorsque l'utilisateur a fait une pause dans son action).
- `asservitive` : haute priorité, l'utilisateur doit en être immédiatement informé. À utiliser avec parcimonie pour ne pas perturber la navigation.

#### Déclaration d'un contenu « live »

```
<div id="news" aria-live="polite">
<!-- Ici les résultats du tournoi Blobby Volley en direct -->
</div>
```

Cet exemple comprend un élément `<div>` neutre, dont on sait que son contenu pourra être mis à jour par JavaScript, probablement avec le concours d'Ajax. L'attribut `aria-live` indique que l'outil d'assistance doit « surveiller » cet élément pour annoncer ce changement.

Les valeurs pour `aria-relevant` conditionnent le niveau de précision des informations transmises par l'agent d'assistance à l'utilisateur selon le type d'événement :

- `additions` : lorsque des nœuds DOM sont ajoutés à la région.
- `removals` : lorsque du texte ou des nœuds sont retirés de la région.
- `text` : lorsque du texte est ajouté à n'importe quel nœud de la région.
- `all` : correspond à l'ensemble des précédentes valeurs.

Si le critère `removals` est utilisé, l'utilisateur devra être informé des suppressions de contenu (texte ou nœuds DOM). Dans le cas d'une synthèse vocale et d'un rempla-

cement de texte « kiwi » par « goyave », le mot « kiwi » sera prononcé, mais il ne sera pas précisé que « goyave » a été retiré. Cette information complète n'est pas toujours utile ou pertinente, il faut donc en user avec parcimonie.

Attention, toutes les régions dynamiques ne se prêtent pas toujours à un emploi d'*aria-live* lorsqu'elles ont une spécialité identifiée, car certains rôles sont déjà prévus à cet effet : *alert*, *status*, *timer*, *marquee* et *log*.

## Drag & drop (glisser-déposer)

Deux attributs existent pour le glisser-déposer.

**Tableau 3-7** Attributs ARIA pour le glisser-déposer « drag & drop »

État ou propriété	Description	Valeurs
<i>aria-dropeffect</i>	Indique quelles fonctions sont déclenchées lorsqu'un objet est déposé sur la cible.	Un ensemble d'un ou plusieurs termes clés : <i>copy</i> , <i>move</i> , <i>link</i> , <i>execute</i> , <i>popup</i> , <i>none</i> .
<i>aria-grabbed</i>	Indique l'état de l'élément saisi, dans une opération de glisser-déposer (état).	<i>true</i> : en train d'être déplacé <i>false</i> : l'élément peut être saisi <i>undefined</i> : ne peut pas être saisi pour un glisser-déposer

Pour en savoir plus, consultez en ligne l'article rédigé par Gez Lemon et traduit par Cédric Trévisan.

### RESSOURCE Zoom sur le drag & drop avec ARIA

Glisser-déposer accessible avec WAI-ARIA

► [http://www.paciellogroup.com/blog/misc/ddfrench/WAI-ARIA\\_DragAndDrop\\_French.html](http://www.paciellogroup.com/blog/misc/ddfrench/WAI-ARIA_DragAndDrop_French.html)

## Relations

Les relations peuvent également être décrites avec des attributs ARIA correspondants.

**Tableau 3-8** Attributs ARIA de relations

État ou propriété	Description	Valeurs	Rôles
<i>aria-activedescendant</i>	Identifie le descendant actif d'un composant.	Identifiant	<i>composite</i> <i>group</i> <i>textbox</i>
<i>aria-controls</i>	Identifie les autres éléments dont la présence ou le contenu sont contrôlés par cet élément.	Liste d'identifiants	
<i>aria-describedby</i>	Identifie le ou les éléments qui décrivent l'objet (complémentaire à <i>aria-labelledby</i> ).	Liste d'identifiants	

Tableau 3-8 Attributs ARIA de relations (suite)

État ou propriété	Description	Valeurs	Rôles
<code>aria-flowto</code>	Identifie le ou les éléments suivants pour l'ordre de lecture, autorisant la technologie d'assistance à prendre le pas sur la hiérarchie naturelle du DOM.	Liste d'identifiants	
<code>aria-labelledby</code>	Identifie le ou les éléments qui confèrent une étiquette à l'objet (similaire à <code>aria-label</code> ).	Liste d'identifiants	
<code>aria-owns</code>	Établit des relations de parenté lorsque le DOM ne permet pas de le faire. Note : un élément ne peut avoir qu'un seul « possesseur ».	Liste d'identifiants	
<code>aria-posinset</code>	Définit la position d'un élément dans un ensemble d'autres éléments de type <code>listitem</code> ou <code>treeitem</code> .	Nombre entier	<code>listitem</code> <code>option</code>
<code>aria-setsize</code>	Définit le nombre total d'éléments dans l'ensemble courant de type <code>listitem</code> ou <code>treeitem</code> .	Nombre entier	<code>listitem</code> <code>option</code>

La plupart des relations établissent un lien entre un élément et un autre dans le document, par exemple `aria-labelledby` et `aria-describedby` à des fins d'étiquetage et de description.

### Exemple de relations ARIA

```
<div role="application" aria-labelledby="todolist" aria-describedby="info">
<h1 id="todolist">Liste de choses à faire</h1>
  <p id="info">
    Ce tableau récapitule toutes les tâches en attente.
  </p>
  <div role="grid">
    <!-- ... -->
  </div>
</div>
```

Le conteneur principal est une application, dont le titre peut être retrouvé dans l'élément d'identifiant `todolist`, et dont la description est présente dans l'élément d'identifiant `info`.

En appliquant ce principe à `<figure>` en HTML 5 nous avons là une manière de lier deux morceaux de contenu d'une manière bien plus forte que leur simple proximité.

### Relation ARIA avec figure

```
<figure>
  
```

```

<figcaption id="legende1">
  <strong>Capture d'écran</strong> : L'extension Firebug pour Firefox propose
des outils de développement web regroupés sous forme d'onglets.
</figcaption>
</figure>

```

Tous ces attributs forment un ensemble cohérent et fourni pour la description avancée des états et propriétés d'une application ou d'un document web. De nombreux exemples sont disponibles en ligne pour explorer toutes les facettes d'ARIA.

#### RESSOURCE Exemples en ligne

Codetalks (notamment la section « Validation and Testing »)

▶ <http://codetalks.org/>

OpenAjax Alliance

▶ <http://www.oaa-accessibility.org/examples/>

Illinois Center for Information Technology and Web Accessibility

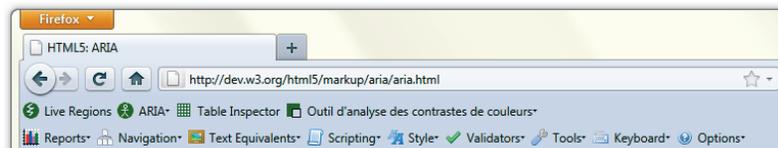
▶ <http://test.cita.illinois.edu/aria/>

## Valider et tester

Le validateur en ligne [validator.nu](http://validator.nu) comporte une option compatible avec la syntaxe HTML 5 + ARIA ou XHTML 5 + ARIA.

La meilleure façon de s'assurer de l'accessibilité d'un site est de le tester avec les outils appropriés, et au minimum avec la navigation au clavier qui est disponible nativement dans tous les navigateurs. Il existe des versions d'évaluation des deux plus célèbres lecteurs d'écran : JAWS et de Window-Eyes. NVDA est quant à lui diffusé en Open Source et téléchargeable librement. Des extensions spécifiques existent également, notamment pour Mozilla Firefox avec la Juicy Studio Accessibility Toolbar et l'Accessibility Evaluation Toolbar.

**Figure 3–6**  
Extensions pour Firefox



L'extension Juicy Studio Accessibility Toolbar produit des résumés, notamment pour les rôles ARIA attribués.

**Figure 3-7**  
Synthèse des rôles ARIA



## Juicy Studio: WAI-ARIA Properties

### Properties

Properties	Element	Parent Nodes	Markup
<ul style="list-style-type: none"> <li>role="navigation"</li> </ul>	DIV	<ul style="list-style-type: none"> <li>HTML</li> <li>BODY</li> </ul>	<code>&lt;div role="navigation"&gt;</code>
<ul style="list-style-type: none"> <li>role="banner"</li> </ul>	H1	<ul style="list-style-type: none"> <li>HTML</li> <li>BODY</li> <li>DIV</li> <li>DIV#header</li> <li>DIV#header-inside</li> </ul>	<code>&lt;h1 role="banner"&gt;</code>
<ul style="list-style-type: none"> <li>role="search"</li> </ul>	FORM	<ul style="list-style-type: none"> <li>HTML</li> <li>BODY</li> <li>DIV</li> <li>DIV#sous-menu</li> </ul>	<code>&lt;form method="post" action="/search/" role="search" id="recherche"&gt;</code>
<ul style="list-style-type: none"> <li>role="main"</li> </ul>	DIV	<ul style="list-style-type: none"> <li>HTML</li> <li>BODY</li> <li>DIV#global</li> <li>DIV#page</li> </ul>	<code>&lt;div role="main" id="content"&gt;</code>
<ul style="list-style-type: none"> <li>role="complementary"</li> </ul>	DIV	<ul style="list-style-type: none"> <li>HTML</li> <li>BODY</li> <li>DIV#global</li> <li>DIV#page</li> </ul>	<code>&lt;div role="complementary" id="sidebar"&gt;</code>
<ul style="list-style-type: none"> <li>role="contentinfo"</li> </ul>	DIV	<ul style="list-style-type: none"> <li>HTML</li> <li>BODY</li> <li>DIV#global</li> <li>DIV#page</li> </ul>	<code>&lt;div role="contentinfo"&gt;</code>

Sous Mac OS X, VoiceOver est intégré nativement depuis la version 10.4 et se retrouve même sur les plates-formes mobiles sous iOS. Sous Linux, Orca est un autre projet libre destiné à la synthèse vocale, aux plages braille et à l'agrandissement.

**Figure 3-8**  
Requin vs Orque



**TÉLÉCHARGER Lecteurs d'écran****JAWS**

- ▶ <http://www.freedomscientific.com/jaws-hq.asp>

**Window-Eyes**

- ▶ <http://www.gwmicro.com/Window-Eyes/>

**NVDA**

- ▶ <http://www.nvda-fr.org/>

**Juicy Studio Accessibility Toolbar**

- ▶ <https://addons.mozilla.org/fr/firefox/addon/juicy-studio-accessibility-too/>

**Accessibility Evaluation Toolbar 1.5.62.0**

- ▶ <https://addons.mozilla.org/fr/firefox/addon/accessibility-evaluation-toolb/>

**Orca**

- ▶ <http://live.gnome.org/Orca>

## L'aide de JavaScript

Le nombre de compléments JavaScript facilitant l'application de WAI-ARIA est croissant. Parmi eux, l'un des plus simples pouvant servir de base à une personnalisation plus avancée est `accessifyhtml5.js` : son action se borne à définir quelques rôles et propriétés par défaut pour des éléments HTML tels que `<header>`, `<footer>`, `<nav>`, `<output>`, `<aside>`, etc. Attention, ce type de script n'est pas miraculeux, il ne pourra deviner automatiquement tous les rôles applicables.

**TÉLÉCHARGER Scripts Accessifyhtml5.js (pour jQuery)**

- ▶ <https://github.com/yatil/accessifyhtml5.js/blob/master/accessifyhtml5.js>

## ANNEXE A

**Fonctionnalités modifiées  
et obsolètes ..... 605**

Différences HTML 5 par rapport à HTML 4 .....	605
Fonctionnalités obsolètes .....	605
Fonctionnalités obsolètes non conformes	606
Éléments .....	606
Attributs .....	606

## ANNEXE B

**Feuilles de style CSS ..... 609**

Principe général .....	611
Sélecteurs .....	612
Propriétés .....	613
Pseudo-classes et pseudo-éléments .....	619
Règles @ .....	620
Media queries .....	621

## ANNEXE C

**Accessibilité et ARIA ..... 623**

Qu'est-ce que l'accessibilité du Web ? .....	624
HTML sémantique .....	627
WAI, WCAG et ARIA .....	628
Les rôles et propriétés de WAI-ARIA .....	631
Rôles avec l'attribut role .....	632
<i>Points de repère (landmark roles)</i> .....	632
<i>Structure de document</i> .....	634
<i>Composants graphiques (widgets)</i> .....	635
Propriétés et états avec les attributs aria-* .....	638
<i>Globaux</i> .....	639
<i>Contrôles d'interface</i> .....	640
Il y a de la vie sur cette planète .....	643
<i>Drag &amp; drop (glisser-déposer)</i> .....	645
<i>Relations</i> .....	645
Valider et tester .....	647
L'aide de JavaScript .....	649

**Index ..... 651**