HTML 5

Une référence pour le développeur web

HTML 5 • CSS 3 • JavaScript • DOM • W3C & WhatWG
Drag & drop • Audio/vidéo • Canvas • Géolocalisation • Hors ligne
Web Sockets • Web Storage • File API • Microformats

2e édition
Traite les dernières évolutions d'HTML 5

Rodolphe Rimelé

Préface de Raphaël Goetter

EYROLLES

 ISBN : 978-2-212-13638-8

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'usager

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. Méticuleux, 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 » ™ 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 répandra 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 plus 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 de nouveautés CSS 3 mentionnées au sein des

chapitres). Ce sont des sujets qui sont eux-mêmes aussi vastes et qui méritent des apports complémentaires à la seule vision du code HTML.

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 ou la mise en page attendue. Certains exemples figurant dans un chapitre ne reprendront pas nécessairement toutes les recommandations présentes dans les 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 repris intégralement, 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

Annexe sur CSS et l'accessibilité (ARIA) et tableaux de prise en charge mis à jour

Cet ouvrage est complété de deux annexes et d'un index :

- l'**annexe A**, qui fait un point sur les « **Fonctionnalités modifiées et obsolètes** » entre HTML 4 et HTML 5 ;
- l'**annexe B**, 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 sur le site d'accompagnement du livre des ressources et compléments, notamment une **annexe C** indispensable donnant un précieux rappel synthétique sur les « **Feuilles de style CSS** », ainsi qu'une collection de liens utiles.

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

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

- http://www.fotolia.com
- http://www.photl.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 comptaient sur les doigts d'une seule main. Cela n'en faisait pas pour autant un monde moins fabuleux ni moins vaste – mode nostalgie *off*.

Comme beaucoup d'autres passionnés, je dois tout ce que j'ai appris à la veille technologique 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 Alsacreations.com.

Bien qu'intéressé par mes études menées à leur terme, je me suis aperçu que la configuration de routeurs (et d'autres protocoles empilés 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, bien heureusement HTML ; mais aussi PHP, MySQL, Perl, C et tout ce qui concerne l'administration de systèmes et serveurs Linux. Connaître se qui se cache plusieurs niveaux en-dessous du navigateur est, à mon sens, primordial pour le dompter. 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 du 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 », mais à adopter des approches pragmatiques.

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.

Philippe Vayssière, Simon Kern, Geoffrey Crofte, Coralie Leveillé-Menez, Guillaume Focheux, Stéphanie Walter et Jennifer Noesser 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* et Benoît Laurenti pour de multiples raisons.

Table des matières

Une brève histoire du Web et de ses standards

1

Pour comprendre la façon dont le langage HTML 5 a été pensé et conçu, il faut se replonger dans son histoire mouvementée au W3C et au WhatWG.

Figure 1–1 Ce manuscrit n'a jamais existé

Le langage HTML *(HyperText Markup Language)* est au Web ce que la portée musicale est à l'orchestre. L'un ne pourrait exister sans l'autre. Les musiciens, quelle que soit leur nationalité, ne pourraient interpréter l'œuvre du compositeur sans cette notation commune, pour jouer de « concert », sans fausses notes et en rythme.

Tout le monde a déjà entendu parler de HTML. Tous les internautes ont déjà vu cette extension dans la barre d'adresses de leur navigateur. Pourtant, très peu savent ce qui se cache réellement derrière ces quatre lettres mystérieuses qui leur permettent d'accéder à leurs sites et services favoris.

En tant que concepteur, designer ou intégrateur web, on croit le maîtriser puis l'on découvre de nouvelles applications chaque jour, de nouvelles subtilités et astuces qui en font un sujet passionnant, voire monstrueux lorsqu'il s'agit de contenter tous les navigateurs sachant l'interpréter avec plus ou moins de virtuosité.

Un successeur pour HTML 4 et XHTML

Au commencement, la Darpa (agence du département de la Défense des États-Unis) crée **Arpanet**. Il s'agit, au début des années 1970, de relier des universités en réseau pour permettre les échanges de données. Par la même occasion, le protocole TCP/IP est inventé pour uniformiser le transit des informations entre les machines. Il est encore utilisé de nos jours. Toute machine ou terminal ayant accès à Internet possède une adresse IP.

Dans les années 1980, le réseau est scindé en deux, d'un côté Milnet (à vocation militaire, ex-DDN) et d'un autre côté NSFnet (à vocation universitaire, ex-Internet). Les applications sont diverses, mais très austères : échange de fichiers (FTP), e-mails, avec des serveurs reliés entre eux à une vitesse fulgurante de 56 kbit/s. Le système des DNS *(Domain Name System)* est inventé à son tour pour permettre de nommer les machines plutôt que d'avoir à mémoriser leur adresse IP.

En 1984, le **Cern** (Organisation européenne pour la recherche nucléaire) adopte le même type de réseau pour ses échanges internes, puis l'étend et le relie à un laboratoire américain via Internet. C'est en son sein que l'équipe de **Tim Berners-Lee**, chargée de réorganiser l'information, invente un nouveau protocole, simple et abordable, destiné à la mise en ligne de pages possédant des liens hypertextes. Dès lors, l'usage devient public et l'on baptise toutes ces pages reliées entre elles, telle une grande toile mondiale : « *World Wide Web* ». Il s'agit d'ailleurs du nom du premier navigateur dont la paternité revient à Tim Berners-Lee.

Les premières versions de HTML voient le jour dans les années 1990, dérivées de la grande famille des langages **SGML** *(Standard Generalized Markup Language)*. Il ne s'agit cependant pas de normes (il n'y a aucune spécification HTML 1.0), car le langage

reste en pleine évolution, principalement motivée par les développements des navigateurs. Les pages utilisent le protocole **HTTP** *(HyperText Transfer Protocol)* pour transiter sur le réseau et établir le dialogue entre le navigateur et le serveur. L'**IETF** *(Internet Engineering Task Force)* héberge les premiers groupes de travail HTML.

En 1993, le navigateur **Mosaic** de NCSA *(National Center for Supercomputer Applications)* développé pour Sun remporte un franc succès sous Unix. Il inaugure l'élément `img` pour l'incorporation d'images et les formulaires pour la saisie de données. L'année suivante, une partie de son équipe prend son envol et fonde Netscape. Le navigateur phare **Netscape Navigator** ajoute de nombreux éléments de présentation (polices des textes, alignement, clignotement) allant totalement à l'encontre du but premier de HTML.

À partir de là, les perspectives d'évolution divergent durant de nombreuses années, chaque navigateur introduisant des balises propriétaires pour satisfaire les besoins de présentation des documents. L'usage de certains éléments (ou *tags*) est détourné de son but initial, notamment l'élément `table` pour la structuration en colonnes et la découpe de la page en zones distinctes. On commence à redouter ce que l'on nomme la *soupe de tags*.

Tim quitte le Cern en 1994 pour rejoindre le MIT *(Massachusetts Institute of Technology)* et fonde le **W3C** *(World Wide Web Consortium)* pour promouvoir la standardisation des langages du Web.

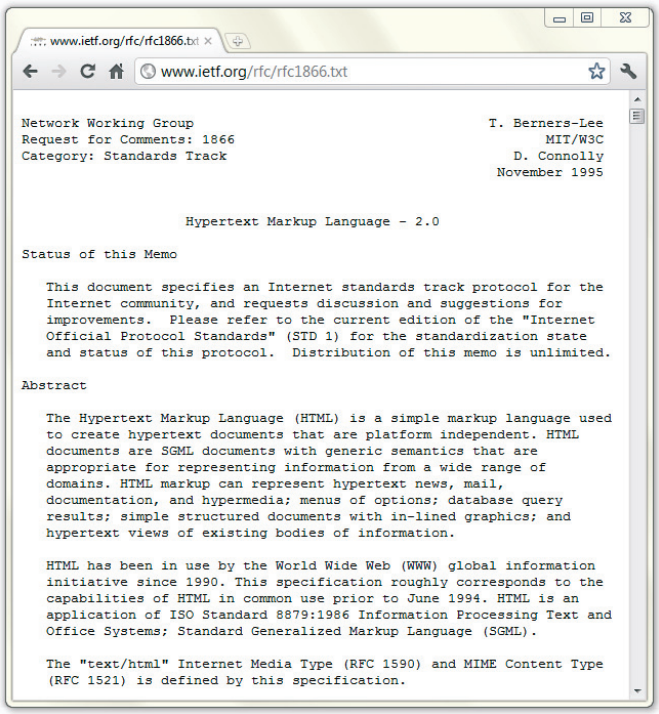


Figure 1–2
RFC 1866 – HTML 2.0

En 1995, le W3C publie HTML 2.0 (sans les ajouts spécifiques à Netscape Navigator 2) et débute le brouillon HTML 3.0 qui ne débouchera pas directement sur des implémentations concrètes. **JavaScript**, langage de programmation créé pour ajouter de l'interactivité aux pages web, est inventé pour le compte de Netscape par Brendan Eich.

Microsoft s'aperçoit à son tour de l'existence d'Internet, crée la première version d'Internet Explorer pour Windows (basée sur Mosaic), suivie de près par sa version 2.0 supportant entre autres une variante de JavaScript nommée JScript, les *frames* et les cookies.

En 1996, un nouveau standard nommé **CSS** (*Cascading Style Sheets* – feuilles de style en cascade) est officialisé pour régir tout ce qui concerne la présentation des documents et séparer la forme du contenu. Il mettra un peu moins de dix années à s'imposer. Internet Explorer passe en version 3.0 en intégrant déjà quelques fonctionnalités des CSS.

En 1997, le W3C publie **HTML 3.2**, officialisant certaines des inventions propriétaires des navigateurs. Internet Explorer 4.0 installé par défaut avec Windows 98 supplante Netscape. Il fait appel malheureusement à des nouveautés dont Microsoft est propriétaire (JScript, VBScript, ActiveX). Le travail continue immédiatement sur HTML 4.0 avec la standardisation de nombreuses fonctionnalités avancées (support des styles, des scripts, des *frames* et des objets) et des améliorations relatives à l'accessibilité.

Après la publication de **HTML 4.0** en décembre 1997, le premier groupe de travail *(HTML Working Group)* cesse son activité. Le second, qui sera en réalité le groupe de travail XHTML *(Extensible HyperText Markup Language)*, est chargé de redéfinir HTML comme une application de XML, avec un rôle limité de maintenance pour HTML 4.0 puis 4.01. Ce groupe est à l'origine de **XHTML 1.0**, mais ne produit aucune avancée concrète pour le langage HTML.

Figure 1–3
W3C – HTML 4.01

La spécification XHTML 1.0 reprend les balises HTML 4.01 à l'identique, il s'agit juste d'adopter une syntaxe plus stricte suivant les règles du XML que l'on promettait à un bel avenir. Cette syntaxe définissant un cadre avec des règles plus structurées et combinées à CSS, conquiert de nombreux auteurs web qui y voient un ensemble de bonnes pratiques à favoriser.

En 1998, **CSS 2** est publié en recommandation, mais sa complexité d'implémentation amènera plus tard le W3C à produire une révision (2.1) avec une approche plus réaliste. En parallèle, le développement de CSS 3 commence, découpé en modules pour pouvoir bénéficier de degrés d'avancement divers.

On entre alors dans une période de stagnation apparente du côté du W3C, dont le fonctionnement trop fermé est critiqué par une partie des développeurs web. Pourtant, quelques avancées voient le jour du côté des navigateurs, par exemple XMLHttpRequest en 1999 avec Internet Explorer 5 pour les prémices de l'**Ajax**.

Les animations **Flash** (promues successivement par FutureWave, Macromedia, et Adobe) remportent un succès indéniable, car il est désormais possible, via l'installation d'une petite extension au navigateur, d'embarquer dans les pages web moult animations, dessins vectoriels, sons, vidéos, éléments d'interaction, le tout complété avec un langage de programmation nommé ActionScript.

En 2000, AOL rachète Netscape, signant par là son déclin progressif. Le navigateur en version 4 est renommé « Communicator » mais affiche de faibles performances et souffre des pratiques commerciales de Microsoft. Le code source de ce qui était espéré devenir la version 5.0 de Netscape passe sous licence libre en 1998 avec la création de l'organisation **Mozilla**, qui s'aperçoit que ce dernier est irrécupérable et débute l'écriture du moteur Gecko.

Microsoft lance **Internet Explorer** 6.0 en 2001 et n'y touche plus durant 6 ans, considérant la suprématie établie (jusqu'à 95 % de parts de marché). Pourtant, la tendance s'infléchit grâce à la concurrence insistante des navigateurs alternatifs qui proposent un bien meilleur support des standards du Web, et quelques inventions bien pratiques en termes de confort de navigation et d'interface utilisateur.

Dans l'élan, le W3C publie **XHTML 1.1** qui se révèle complexe à mettre en place, compte tenu des contraintes imposées par la spécification et des moteurs des navigateurs utilisés par les internautes. En effet, un document créé en XHTML et servi avec le type MIME adéquat, ne pouvait même pas être compris par le navigateur le plus répandu à ce moment-là, Internet Explorer.

En 2002, la fondation Mozilla lance sur les (quelques) restes de Netscape Navigator/Communicator et de la branche principale de Mozilla un nouveau projet de navigateur Open Source nommé successivement Phoenix, Firebird puis **Firefox**, dont les innovations séduisent un public averti, puis de plus en plus large. Son extensibilité et son respect des standards font sa force.

Opera Software défend le navigateur du même nom, initialement payant puis devenu gratuit en 2005. Celui-ci fédère une communauté d'utilisateurs convaincus par sa réactivité et ses innovations. On le retrouve également sur consoles et plates-formes mobiles.

Les soucis commencent lorsque le W3C décide d'évoluer à terme vers **XHTML 2.0** dont le premier brouillon est dévoilé le 5 août 2002. Cette nouvelle version, qui se veut un nouveau départ, allégée de son lourd héritage, est incompatible avec les précédentes. Certaines balises très courantes ne sont plus valides, telles que `img` pour les images. Comment faire alors ? Fournir une version du site en HTML pour les navigateurs actuels et maintenir en parallèle une version XHTML 2.0 pour d'éventuelles implémentations dans de futurs navigateurs ?

C'est une catastrophe tant auprès des éditeurs de navigateurs qui veulent se concentrer sur les applications concrètes répondant aux besoins des développeurs, que du public qui ne comprend pas ce revirement de situation, et tarde à s'intéresser au sujet.

Suite à cette période trouble et à la décision du W3C d'abandonner HTML en faveur de langages basés sur XML, la mailing-list WhatWG est créée le 4 juin 2004. À l'initiative du groupe, Ian Hickson d'Opera travaille sur Web Forms 2.0 pour étendre les possibilités des formulaires HTML – spécification qui sera intégrée à HTML 5 par la suite. Le W3C tente de persévérer sur la voie du XML.

Le **WhatWG** *(Web Hypertext Application Technology Working Group)* est constitué par des passionnés souhaitant améliorer HTML, issus d'équipes de la fondation Mozilla, d'Apple et d'Opera. Leur but est de faire évoluer le langage pour répondre aux besoins concrets de l'explosion du Web, tout en maintenant sa simplicité et sa rétrocompatibilité. Ce groupe communautaire n'a pas d'existence administrative à proprement parler, il ne vit que sur le Web, avec un nombre illimité de membres qui y participent gratuitement.

Le WhatWG débute son travail en juin 2004 sous la dénomination de Web Applications 1.0. Ian Hickson devient éditeur officiel du document HTML 5, et rejoint finalement Google (membre du W3C).

La spécification **HTML 5**, soutenue en avril 2007 devant le *World Wide Web Consortium* par la fondation Mozilla, Apple et Opera, est acceptée comme point de départ du nouveau groupe de travail HTML. Ce dernier publie le premier brouillon *(Working Draft)* le 22 janvier 2008 et admet que XHTML 2.0 est trop ambitieux.

Le processus est alors assez inhabituel puisque les deux entités, W3C et WhatWG collaborent sur le projet HTML 5 avec des approches différentes, parfois quelques accrochages sur la méthode, mais avec un pragmatisme certain grâce à la constitution très forte des équipes par des membres d'éditeurs de navigateurs.

Entre-temps, Apple a lancé **Safari** pour Mac (2003) puis Windows (2009) pour fournir un navigateur de qualité à sa base d'utilisateurs grandissante, en déclinant le moteur KHTML en WebCore/WebKit.

Google, devenu le plus célèbre moteur de recherche et acteur majeur du Web, développe son propre navigateur **Google Chrome** à partir de septembre 2008 sur la base d'un projet libre nommé Chromium, à un rythme extrêmement soutenu. Google souhaite privilégier la simplicité de navigation et les performances à l'exécution.

Ces navigateurs alternatifs s'approprient petit à petit la plus grande part du marché. Mozilla Firefox dans sa version 3.0 devance peu à peu Internet Explorer dans certaines régions du monde et s'affirme comme une alternative de prédilection avec de nombreux projets gravitant autour du moteur Gecko.

Les **navigateurs mobiles**, versions dérivées de leurs grands frères, confèrent aux smartphones et tablettes tactiles la possibilité d'accéder au Web depuis que les capacités des réseaux sans fil permettent un débit d'accès acceptable et que la mobilité entre dans les mœurs. Ils font alors appel à des comportements de navigation différents découlant des temps de chargement et de la navigabilité au doigt.

Microsoft sort finalement de sa léthargie et reprend un rythme de développement honorable avec Internet Explorer 7 en 2006 pour l'occasion de la sortie de Windows Vista.

Internet Explorer 8 pointe le bout de son nez en mars 2009. En juillet, le W3C annonce le non-renouvellement du groupe de travail XHTML 2.0 pour réunir les forces autour du groupe de travail HTML et clarifier sa position. Tandis que CSS 3 voit ses modules complétés progressivement, la spécification HTML 5 est désormais adoptée formellement comme langage de prédilection pour le Web par le W3C et le WhatWG.

Dès lors, tous les navigateurs s'y mettent et dopent chacune des versions publiées. Début 2011, Microsoft publie Internet Explorer 9, précédé de peu par Opera 11. Mozilla décide d'adopter un nouveau modèle de développement basé sur plusieurs canaux (Aurora, version Bêta, version Finale) pour diffuser les nouveautés plus rapidement. La numérotation des versions s'emballe à la façon de Chrome.

À la rentrée 2011, Microsoft surprend tout le monde en confirmant son intérêt pour les standards du web, et sa volonté de les exploiter à un haut niveau via le moteur d'Internet Explorer 10. Windows 8 est distribué fin 2012 avec une nouvelle interface utilisateur, qui interprète directement HTML 5, CSS 3 et JavaScript pour créer des applications intégrées au système.

L'enthousiasme gagne Amazon qui annonce une accointance des tablettes Kindle avec HTML 5. Adobe historiquement ancré dans les suites de logiciels graphiques consacre plus d'efforts au standard, délaissant quelque peu Flash, notamment sur les mobiles. De nouveaux outils sont lancés avec la collection Edge pour générer des ani-

mations dans des formats ouverts et se tourner franchement vers les développeurs-intégrateurs web.

En 2013, Mozilla se lance dans une nouvelle bataille avec Firefox OS et Firefox Marketplace, qui utilisent les technologies web pour développer tout un système d'applications sur périphériques mobiles : l'enjeu stratégique est celui du nomadisme et de l'ubiquité.

CURIOSITÉ HISTORIQUE **Les archives de Tim**

Le premier code source implémentant HTML, écrit par Tim Berners-Lee

- http://www.w3.org/History/1991-WWW-NeXT/Implementation/HyperText.m

La « première page » HTML

- http://www.w3.org/History/19921103-hypertext/hypertext/WWW/Link.html

Une des premières ébauches détaillant les balises

- http://www.w3.org/History/19921103-hypertext/hypertext/WWW/MarkUp/MarkUp.html

Une spécification pour HTML 2.0 (novembre 1995)

- http://www.w3.org/MarkUp/html-spec/index.html
- http://www.ietf.org/rfc/rfc1866.txt

Piste cachée : un historique animé de bonne facture

- http://webdirections.org/history/

Le rôle du W3C

La mission du W3C est de développer et promouvoir des standards pour le Web afin d'en assurer la croissance et l'universalité. Ses membres et experts internationaux sont constitués en groupes de travail qui rédigent ces standards.

Figure 1–4
Logo du W3C

Le W3C est administré par trois entités :

- le MIT *(Massachusetts Institute of Technology)* aux États-Unis ;
- l'université Keio au Japon ;
- l'Ercim *(European Research Consortium for Informatics and Mathematics)* en Europe, remplaçant de l'INRIA français.

Il regroupe environ 400 membres (organisations) et se voit désormais à l'origine de dizaines d'autres spécifications et recommandations, telles que MathML, RDF, Soap, SVG, Smil, PNG, et autres variations autour de XML.

Les groupes de travail (abrégé WG, alias *Working Group*) sont variés, et œuvrent de façon modulaire. Entre autres, le *HTML Working Group* se consacre à HTML, Microdata et Canvas en 2D ; le *Web Apps Working Group* se concentre sur la partie dynamique avec Web Messaging, Web Workers, Web Storage, WebSocket, Server-Sent Events (sachant que l'IETF participe au protocole même de WebSocket) ; et d'autres groupes se focalisent sur WebRTC ou WebVTT.

Une maturation rigoureuse…

Le processus d'écriture et de validation des documents fait mention de statuts précis :

1. *Working Draft :* brouillon de travail (WD).
2. *Last Call Working Draft :* dernier appel pour modifications (LC).
3. *Candidate Recommendation :* candidat à la recommandation (CR).
4. *Proposed Recommendation :* recommandation proposée (PR).
5. *W3C Recommendation :* recommandation W3C officielle (REC).

Un brouillon (WD) est publié pour être débattu par la communauté (membres du W3C, grand public, autres organisations). Une recommandation candidate (CR) est un document dont le W3C pense qu'il a été largement examiné et satisfait aux prérequis techniques du groupe de travail. Il s'agit alors de procéder à des implémentations concrètes pour recueillir des retours d'expérience. Est divulguée ensuite une recommandation proposée (PR), version plus mûre établie après de larges applications techniques et soumise au comité consultatif pour approbation finale. Quand il est clair qu'un support significatif est assuré par le grand public et le W3C, la recommandation (REC) voit le jour, signifiant que le W3C recommande son vaste déploiement et la mise en application de son contenu.

EN SAVOIR PLUS **Au sujet du W3C**

Standards et spécifications (brouillons, recommandations, etc.)

- http://www.w3.org/TR/

HTML Working Group

- http://www.w3.org/html/wg/

… mais peu véloce

Ces stades peuvent nécessiter de longues périodes de relectures, tests et discussions pour évoluer favorablement en vue d'un consensus général.

Il en a été ainsi pour CSS 2.1, qui a mis de nombreuses années à atteindre l'étape de recommandation candidate (CR) en 2009 alors que des navigateurs l'implémentaient bien avant cela à des degrés divers. En automne 2010, le groupe de travail CSS officialise la publication de la suite de tests pour CSS 2.1. Toutes les équipes de développement des navigateurs majeurs acceptent de publier leurs rapports en conséquence. C'est à l'appui de ces rapports une fois publiés que les responsables du groupe de travail peuvent enfin demander à la direction du W3C de passer en recommandation proposée (PR). Ce stade est atteint fin 2010. La tête du consortium annonce alors un appel à validation *(Call for Review of a Proposed Recommendation)* à tous les membres. CSS 2.1 atteint enfin le statut de recommandation du W3C début 2011.

HTML 4.01 était passé au stade de recommandation le 24 décembre 1999, en apportant quelques corrections mineures à la version 4.0. Quant à HTML 5, les navigateurs l'implémentent progressivement sans attendre son accession à l'étape finale ; il est de notoriété publique qu'il n'est pas (absolument) nécessaire d'attendre pour l'exploiter. Ainsi certaines versions de navigateurs l'intègrent de façon partielle, d'autres mieux tout en nécessitant encore des perfectionnements. Microsoft a déjà mis en œuvre quelques fonctions de HTML 5 depuis Internet Explorer 8 et de façon plus complète dans la version 9.

En 2009, Ian Hickson, éditeur de la spécification HTML 5, avait annoncé vouloir atteindre la recommandation candidate (CR) en 2012. Dans les faits et à partir de cette étape, le W3C réclame deux implémentations complètes et interopérables, avec de larges batteries de tests (plus de 20 000), pour aboutir à la recommandation finale, ce qui devait *en théorie*, selon les pronostics (souvent ironiques) de Ian, s'établir aux alentours de 2022. Cette date est effrayante, mais heureusement, HTML 5 sera en pratique depuis longtemps répandu et compris par les navigateurs. La date a, depuis, été ramenée officiellement à l'intervalle 2014-2016.

Début 2011, un logo officiel a été dévoilé pour faire la promotion de HTML 5. Celui-ci comporte toujours un numéro de version et des déclinaisons pour les sous-groupes de technologies : la sémantique *(semantics)*, l'exécution *(offline & storage)*, l'accès au matériel et aux périphériques *(device access)*, la connectivité *(connectivity)*, le multimédia *(multimedia)*, le graphisme et les effets *(3D, graphics & effects)*, la performance et l'intégration *(performance & integration)*, CSS 3.

Le W3C, qui développe déjà des dizaines d'autres spécifications, prévoit de finaliser officiellement HTML5 fin 2014, bien que celui-ci soit déjà exploitable pour beau-

coup de fonctionnalités dans les navigateurs modernes, puis « HTML 5.1 » fin 2016, puis une version « 5.2 » par la suite.

> **RESSOURCE** **Pages intéressantes du W3C**
>
> Le logo du W3C :
> - http://www.w3.org/html/logo/
>
> La feuille de route stratégique pour 2014 et 2016 :
> - http://dev.w3.org/html5/decision-policy/html5-2014-plan.html
>
> L'annonce concernant 2014 puis 2016 :
> - http://lists.w3.org/Archives/Public/public-html/2012Sep/0243.html

Figure 1–5
Logo HTML 5 par le W3C

Figure 1–6
« Badge » HTML et technologies

Le rôle du WhatWG

Durant la même période, le WhatWG a annoncé vouloir désormais parler uniquement de « HTML » sans préciser de numéro de version, en tant que partie des spécifications *Web Applications*, qui sont toujours maintenues et qui comprennent les autres API web décrites dans cet ouvrage (Web Workers, Web Storage, etc.). Le groupe considère que la spécification HTML *made in* WhatWG est désormais mature et qu'elle ne mérite plus le statut de brouillon mais de *norme de fait*, ou *living standard* (standard vivant).

Figure 1–7
Le logo du WhatWG

Le WhatWG travaille en tandem avec le W3C, qui tire des « instantanés » de la spécification pour en fournir une version stable aux éditeurs de navigateurs et aux développeurs qui ont de fortes contraintes d'interopérabilité. À la mi-2012, des divergences de fonctionnement et d'objectifs refont surface : les groupes de travail reprennent un peu plus d'indépendance. Les contributions du WhatWG, voulues rapides et ancrées dans la réalité, implémentées par les navigateurs, viendront certainement rejoindre les prochaines versions de HTML du côté W3C. Les responsables des spécifications sont désormais indépendants.

Quelques différences voient le jour. Par exemple, les éléments `<data>` et `<dialog>` sont désirés par le WhatWG mais absents du côté du W3C, qui ne mentionne pas non plus Microdata.

> **EN SAVOIR PLUS** **WhatWG ou W3C ?**
>
> L'annonce de la modification des relations par Ian Hickson :
> - http://lists.w3.org/Archives/Public/public-whatwg-archive/2012Jul/0119.html
>
> HteuMeuLeu partage sa vision de la situation :
> - http://www.hteumeuleu.fr/le-w3c-le-whatwg-et-html5/
>
> Jean-Pierre Vincent résume HTML5 sur BrainCracking :
> - http://braincracking.org/2012/02/01/html5-aujourdhui/

Les fondements de l'évolution

Comme dans toute évolution de langage, des compromis ont dû être effectués, principalement pour rectifier des erreurs commises par le passé, ou pour asseoir de meilleures pratiques de conception. Ainsi, l'accessibilité joue souvent un grand rôle dans les décisions prises pour la création, la modification ou la suppression d'éléments/attributs. Certains éléments de HTML 4.01 ont été abandonnés (déclarés obsolètes), notamment ceux jouant un rôle de présentation tels que `<font>` et `<center>`, étant avantageusement remplacés par les feuilles de style CSS.

Depuis la spécification de HTML 4, les applications concrètes se sont faites plus nombreuses, et les internautes sont passés du stade de passionnés privilégiés et initiés au high-tech, à un grand public avide de contenus variés et versé dans l'e-commerce ou les réseaux sociaux. Là où l'on pouvait se contenter de pages statiques, il est aujourd'hui impensable de ne pas proposer des sites dynamiques, alliant animations, audio, vidéo, composants interactifs.

HTML 5 se propose de redevenir la référence en termes de standard ouvert pour des applications en lieu et place de technologies telles que Flash (Silverlight, Java, etc.) dont la croissance a été rapide. Avec l'inauguration d'API orientées vers le graphisme, la vidéo, l'audio, et la communication, HTML peut désormais prétendre à la création de jeux de haute qualité et d'applications très riches visuellement.

La **simplification du langage** a pour but de faciliter son apprentissage et de le mettre à portée de tout débutant qui souhaiterait s'initier au HTML. On notera comme exemple parfait le nouveau *doctype* unique (`<!DOCTYPE html>`) qui facilite grandement le choix par rapport aux précédentes versions nécessitant une connaissance des subtilités entre variantes : Strict, Transitionnel, Frameset.

L'approche reste pragmatique au sein du groupe de travail HTML. Le but est de suivre des principes clairs pour favoriser l'adoption de la version 5 du langage :

- grâce au support des contenus existants « *Support Existing Content* » et la défense de l'évolution plutôt que la révolution (*Evolution Not Revolution* pour la rétrocompatibilité HTML4) ;
- grâce au principe de la dégradation gracieuse « *Degrade Gracefully* » : un nouvel élément doit permettre de tolérer une alternative ou une émulation, et de ne pas perdre une fonctionnalité essentielle lorsqu'il n'est pas compris ;
- grâce à la prise en compte des implémentations existantes dans les navigateurs « *Do not Reinvent the Wheel* » (ne pas réinventer la roue) et à leur assimilation par les auteurs web « *Pave the Cowpaths* » (paver le chemin déjà tracé) ;
- grâce à la défense de l'interopérabilité et de l'accessibilité universelle (indépendance du média, support des langues internationales et des contenus alternatifs).

SPÉCIFICATION Principes du W3C

Liste des différences par rapport à HTML4

- http://dev.w3.org/html5/html4-differences/

Les lignes directrices de la conception de HTML

- http://dev.w3.org/HTML5/html-design-principles/

Tout le monde sur la brèche

Les grands éditeurs de logiciels et de navigateurs l'ont compris : il faut être présent en tant qu'acteur du Web pour espérer conquérir des parts de marché ; et il faut démontrer la force des produits pour publier, développer ou exploiter HTML5. Chacun a créé une vitrine référençant des démonstrations, parfois d'avant-garde, mettant en œuvre les différentes technologies web – y compris CSS – pour construire des applications complexes, des jeux, des animations, tout en donnant un excellent aperçu de ce que l'on peut accomplir avec ces langages ouverts.

RESSOURCE Démonstrations technologiques et sites promotionnels

Adobe : The Expressive Web
- http://beta.theexpressiveweb.com/

Mozilla Demo Studio
- https://developer.mozilla.org/en-US/demos/

Chrome Experiments
- http://www.chromeexperiments.com/

Microsoft Test Drive
- http://ie.microsoft.com/testdrive/

Microsoft : The Browser You Loved To Hate
- http://browseryoulovedtohate.com/

Apple HTML5
- http://www.apple.com/html5/

En quoi consiste réellement HTML 5 ?

Du point de vue du **balisage**, HTML 5 inaugure de nouveaux éléments pour de nouveaux usages, afin de répondre aux besoins grandissants des internautes.

- Les premiers permettent d'ajouter une valeur sémantique aux blocs génériques précédemment définis par les éléments `div` et `span`. C'est le cas notamment de `article`, `aside`, `footer`, `header`, `section` et `nav`.
- Les suivants apportent de nouvelles fonctionnalités aux pages web, c'est le cas notamment de `audio` et `video`.

Bien entendu, l'introduction de ces nouvelles balises représente la partie la plus visible et la plus facile à aborder lorsqu'il s'agit de concevoir des pages web. Figurent aussi des **API** *(Application Programming Interfaces)* pour étendre les possibilités dynamiques du langage et d'interaction avec le DOM *(Document Object Model)*, entre autres pour :

- le stockage de données côté client,
- l'élément *canvas* (dessin 2D et 3D),
- les microformats,
- la lecture de médias (audio, vidéo),
- le glisser-déposer *(drag & drop)*,
- la communication bidirectionnelle (*sockets*, *push*, interdocuments),
- la géolocalisation,
- la gestion des applications web hors-ligne,
- la lecture et l'écriture de fichiers locaux,
- la gestion de l'historique du navigateur,
- et bien d'autres à venir...

Survient alors un problème de classification : toutes ces technologies ne sont pas nécessairement incluses dans la spécification HTML 5 du W3C bien qu'elles le soient pour la plupart dans celle, plus généraliste, du WhatWG. Elles font alors l'objet d'une publication séparée. Cependant, elles sont très souvent réunies par les médias par abus de langage sous le terme « HTML 5 », car développées en parallèle et évidemment utilisées en osmose avec HTML.

Certains brouillons sont uniquement présents côté WhatWG et n'ont fait l'objet d'aucune implémentation dans les navigateurs. C'est par exemple le cas, dès 2011, de l'interface *PeerConnection* exploitable pour la vidéoconférence, et d'un gestionnaire d'annulations baptisé *UndoManager*.

Cet ouvrage traite donc volontairement de spécifications réputées ne pas faire partie du document HTML 5 hébergé par le W3C, mais qui sont pourtant exploitées de concert, spécifiées, et implémentées durant la même période par les navigateurs.

SPÉCIFICATION HTML 5 version W3C et WhatWG

La spécification HTML 5 du côté du W3C

- http://www.w3.org/TR/HTML5/
- http://dev.w3.org/HTML5/spec/Overview.html

La spécification HTML du côté du WhatWG et sa FAQ

- http://www.whatwg.org/html
- http://wiki.whatwg.org/wiki/FAQ

Différences depuis HTML 4.01 et XHTML 1.x

Outre la transformation de la philosophie de développement autour de HTML 5, voici un résumé des différences notables :

- de nouvelles règles d'analyse syntaxique ;
- la possibilité d'utiliser SVG ou MathML au sein de HTML ;
- les nouveaux éléments : `article`, `aside`, `audio`, `canvas`, `command`, `datalist`, `details`, `embed`, `figcaption`, `figure`, `footer`, `header`, `hgroup`, `keygen`, `mark`, `meter`, `nav`, `output`, `progress`, `rp`, `rt`, `ruby`, `section`, `source`, `summary`, `time`, `video`, `wbr` ;
- de nouveaux types pour l'élément `<input>` (`color`, `date`, `datetime`, `datetime-local`, `email`, `month`, `number`, `range`, `search`, `tel`, `time`, `url`, `week`) ;
- de nouveaux attributs spécifiques, par exemple ping (pour `<a>` et `<area>`), `charset` (pour `<meta>`), `async` (pour `<script>`) ;
- de nouveaux attributs globaux : `contenteditable`, `contextmenu`, `spellcheck`, `draggable`, `hidden`, `role`, `aria-*`, `data-*` ;
- des éléments dépréciés : `acronym`, `applet`, `basefont`, `big`, `center`, `dir`, `font`, `frame`, `frameset`, `isindex`, `noframes`, `s`, `strike`, `tt`, `u`.

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

De nos jours, les différents langages pouvant être mis en jeu pour la composition d'une page web (ou d'une application web) sont très intimement liés. Ainsi, l'on voit souvent regroupés HTML 5, JavaScript et CSS (dans sa version 3 en cours d'élaboration) sous le terme générique HTML 5 lui-même.

Il s'agit bien là d'un abus de langage, mais un abus justifié : le contenu (HTML) étant bien dissocié de la forme (CSS), mais peu exploitable pour les visiteurs sous une forme brute sans mise en pages, et peu dynamique sans langage de script pour des interactions avec le contenu de la page lui-même. Il est donc difficile de se servir de l'un sans l'autre pour la création de sites complets. **HTML 5 est la pierre angulaire de l'édifice.**

On pourra aussi noter la *coïncidence* – ou plutôt la concomitance – de la période de développement de CSS 3, qui est somme toute logique dans le processus d'évolution des langages web, mais qui associe bien souvent les deux dans les démonstrations technologiques.

De même, HTML 5 est *livré* avec plusieurs API évoluées qui se manipulent avec JavaScript. Les moteurs les plus récents embarquent des avancées majeures pour JavaScript :

- la nouvelle version du langage ECMAScript 5, incluant l'API JSON et le mode strict *(strict mode)* ;
- les tableaux typés natifs pouvant représenter un gain de performance ;
- XMLHttpRequest 2 et les objets FormData ;
- l'API Selectors et l'attribut `classList` ;
- les attributs `async` et `defer` ;
- et toutes les autres améliorations mineures au niveau atomique qui soulagent le développeur de façon majeure.

Les implications du point de vue du développement et de l'intégration sont dès lors plus complexes que par le passé. La synergie de ces trois langages paraît désormais essentielle pour concevoir un site web attractif ou une application web multi-plate-forme.

Page web ou application web ?

La frontière entre une application et une page est floue. On peut considérer qu'une page reste – d'un point de vue technique – cantonnée à un rôle d'affichage de contenu, structuré et mis en forme, comme on sait le faire depuis de nombreuses années, avec pour interaction maximale la saisie d'un formulaire envoyant des données vers un serveur. Une application, elle, fera l'usage d'API évoluées pour mener à bien sa mission : stockage, génération de données, affichage interactif, communications intensives avec un serveur. Néanmoins, personne n'est obligé de se servir de

l'ensemble des fonctionnalités proposées par HTML5 pour déclarer son œuvre« application HTML5 ».

Figure 1–8
Couteau suisse du Web

HTML 5 n'est pas un tout monolithique. C'est un ensemble de fonctionnalités individuelles, bâties autour d'un langage rétrocompatible. Cette approche va permettre d'implémenter certaines d'entre elles progressivement dans les navigateurs, avec l'inconvénient de devoir se préoccuper du support de ces fonctionnalités les unes après les autres, voire de proposer des alternatives de remplacement pour ne pas altérer le confort de l'utilisateur.

Le concept clé de la rétrocompatibilité mérite que l'on s'attarde encore un peu sur lui. Dans son acception générale, il signifie que les navigateurs interprétant HTML 5 doivent aussi continuer à interpréter les pages conçues en HTML 4.0, 4.01 et XHTML 1.0. La spécification définit bien quels éléments sont obsolètes (indiqués en détail dans les annexes), mais aussi comment continuer à les prendre en charge, ce qui place les précédentes versions de HTML comme un « sous-ensemble » de cette nouvelle évolution. Si vous concevez des pages HTML, elles sont déjà en HTML 5.

Pourquoi des standards pour le Web ?

Les standards web sont gages d'interopérabilité et de pérennité. Contrairement aux technologies propriétaires, personne ne peut réclamer de droits pour les utiliser. Ils

permettent de définir une manière universelle de créer les pages web sans les écrire à destination d'un logiciel en particulier – comme on a pu le voir au temps de la « guerre des navigateurs » et des pages « optimisées » pour certains d'entre eux.

L'adhésion à des normes reconnues mondialement facilite la structuration et le balisage des documents, réduit les coûts associés à la production de sites. Elle permet aussi de garder un cuir chevelu sain, en évitant de s'arracher les cheveux avec la conception de pages interprétées de façon commune sur la plupart des navigateurs.

Conclusion et perspectives

C'est déjà fini ? Faisons un point sur l'avenir du HTML et la concurrence des autres technologies web.

HTML est plus que jamais une technologie vivante autour de laquelle s'articulera le Web de demain. La quantité de fonctions proposées, autour du cœur de ce qui a fait les premières pages du réseau, ne cesse de s'agrandir. Les inventeurs des nouvelles spécifications ont su transposer les principaux souhaits des développeurs, et sont prêts à aller encore plus loin. La frontière entre systèmes d'exploitation et Web sera de plus en plus ténue comme le prouvent les API disponibles dès HTML 5. Les applications web, de plus en plus nombreuses, envahissent notre quotidien, accessibles à moindre coût, évolutives en permanence. Il vous appartient de participer à cette révolution !

Quid des frameworks JavaScript et de Flash ?

JavaScript a grandi avec le Web et gagné ses lettres de noblesse. Des frameworks solides (jQuery, Dojo, Mootools et tant d'autres) ont vu le jour pour accélérer la réalisation d'applications complètes et combler quelques lacunes. HTML 5 lui-même fait la part belle aux API et reste ouvert à tous les compléments facilitant la vie du développeur. Outre l'apport de la rétrocompatibilité, les librairies vont continuer à évoluer pour aller encore plus loin en profitant des briques offertes par HTML. Elles se spécialiseront, puis l'on verra émerger un microcosme de noyaux modulaires et complémentaires, qui auront chacun leur domaine de prédilection, en orbite autour des API de stockage, de communication, de graphisme et d'interopérabilité.

Quant à Flash, Silverlight, Adobe AIR, Flex ou Java, qui ont encore beaucoup à proposer pour le Web, il leur faudra choisir l'ouverture ou la spécialisation. Ces technologies ont tiré vers le haut la démocratisation d'Internet auprès des particuliers tant que des professionnels, et cohabiteront encore avec HTML. Les débats sont nombreux, mystiques, tandis que la situation actuelle ne permet pas de prédire si les géants du logiciel (Adobe, Microsoft, Apple, Google) tenteront encore d'imposer leur vision du Web ou de la mobilité pour dominer le monde des communications. Comme c'est déjà le cas avec le marché des applications pour mobiles développées avec des SDK *(Software Development Kit)* spécifiques, il faudra se montrer prudent. HTML 5 a tout pour exceller dans ce domaine, cependant il reste dans certains cas lié aux contraintes imposées par la plate-forme matérielle et logicielle.

Perspectives d'avenir

Selon le rédacteur Ian Hickson, il y aura probablement une spécification « HTML 6 » et des suivantes, jusqu'à ce que le Web tel que nous le connaissons aujourd'hui trouve une autre voie d'évolution. La réflexion a déjà débuté, avant même l'achèvement de HTML 5 dont la suite de tests (inexistante pour HTML 4) doit être constituée depuis zéro pour parvenir après validation à la recommandation W3C. Le groupe de travail pourra ainsi partir sur ces nouvelles bases et probablement passer sur un modèle évoluant d'une façon plus réactive et modulaire par rapport aux besoins des internautes et aux progrès rapides des navigateurs.

Existera-t-il vraiment une telle numérotation des versions à long terme ? Le W3C souhaite poursuivre dans cette voie tandis que le WhatWG veut mettre en avant l'innovation réactive et le « standard vivant », tout en étoffant progressivement HTML. Le groupe de travail HTML fait référence aux futures évolutions avec le terme non officiel « HTML.next ». Que ces prochaines évolutions du langage soient baptisées « HTML 6 » ou non n'est finalement pas très important. L'essentiel reposera sur les épaules des éditeurs de navigateurs, et des passionnés qui tous les jours feront avancer l'étendue du langage.

EN SAVOIR PLUS **L'avenir de HTML**

HTML.next

- http://www.w3.org/wiki/HTML/next

La vague suivante regroupera sûrement les spécifications et les nouveautés qui n'ont pas fait partie de la vague « 5 ». Que pourra-t-on y retrouver ? Probablement une

gestion des fenêtres modales et boîtes de dialogue. Actuellement, celles-ci sont simulées par des scripts complexes et de nombreuses lignes de code HTML/CSS, qui ne sont pas toujours gérées de manière efficace, ni accessibles.

Les connexions *peer-to-peer* sont aussi dans la ligne de mire. On pourra y découvrir une intégration plus poussée des applications web aux systèmes d'exploitation avec les notifications, l'amélioration du *Drag & Drop* depuis de multiples sources. Des améliorations pour la gestion des médias et périphériques sont également sur la feuille de route. Avec *Media Capture API* et la capture de flux audio (micro) et vidéo (webcam, photo) dans le navigateur nous pourrons tous découvrir de nouveaux usages à HTML.

Quelques avancées en ce sens sont aussi destinées à l'établissement d'interactions fortes entre deux services web, par exemple un webmail et une application web d'édition d'images, afin que cette dernière puisse mettre ses compétences à disposition du premier pour l'édition des pièces jointes reçues par e-mail. Les possibilités sont infinies, et ce n'est qu'un début !

À n'en pas douter, HTML nous réserve encore de belles surprises pour améliorer notre quotidien de surfeurs et concepteurs web.

EN LIGNE **Poursuivez votre lecture en ligne avec lune annexesur CSS**

Rendez-vous sur le site d'accompagnement du livre pour téléchargerau format PDF

- l'**annexe C**, qui donne un précieux rappel sur les « **Feuilles de style CSS** ».

Retrouvez également l'auteur sur le site d'accompagnement du livre.

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

Fonctionnalités modifiées et obsolètes

A

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 MIME `text/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` sura 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`, `margintop`, `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.

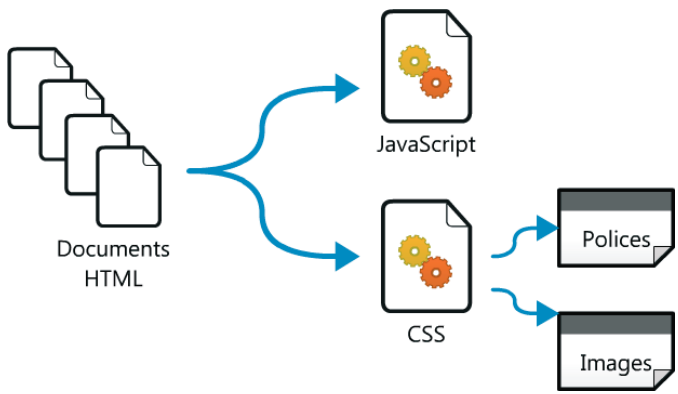


Figure 2–2
Une feuille CSS pour de multiples pages HTML

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	`nav { }`
	`p { }`
	`ul { }`
		`<nav>`
		`<p>`
		`<ul>`

Sélecteur de classe	`.remarque { }` `div.remarque { }`	Tout élément portant l'attribut `class="remarque"` `<div class="remarque">`.
Sélecteur d'id	`#intro { }` `header#intro { }`	Tout élément portant l'attribut `id="intro"` `<header id="intro">`.

Sélecteur d'attribut	`[alt] { }`
	`input[type=submit] { }`
	`[rel=nofollow] { }`
		Tout élément possédant un attribut `alt`
		`<input type="submit">`.
		Tout élément portant l'attribut `rel="nofollow"`.

Sélecteur d'enfant direct	`ul>li { }`	Tout élément `<li>` enfant direct d'un `<ul>`.
Sélecteur d'élément adjacent	`h1+p { }`	Tout élément `<p>` immédiatement précédé par un élément `<h1>`.
Sélecteur d'éléments frères	`h2~p { }`	Un ou plusieurs éléments `<p>` précédés par un élément `<h2>`.

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
`px`	Pixels	Relative
`%`	Pourcentage (en général par rapport à l'élément parent)	Relative
`em`	Cadratin (relatif à la taille de la police)	Relative
`ex`	Hauteur de la lettre x	Relative
`in`	Pouces (= 2,54 cm)	Absolue
`cm`	Centimètres (= 10 mm)	Absolue
`mm`	Millimètre	Absolue
`pt`	Points (= 1/72 in)	Absolue
`pc`	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
`color`	Couleur du texte
`font`	(Déclaration groupée)
`font-weight`	Graisse
`font-size`	Taille de police
`font-family`	Famille de police
`font-variant`	Variante
`font-stretch`	Étirement du texte
`text-decoration`	Décoration de texte

Tableau 2–3 Texte (suite)

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

Exemple

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

Tableau 2–4 Fonds

Propriété	Rôle
`background`	(Déclaration groupée)
`background-color`	Couleur de fond
`background-image`	Image de fond
`background-repeat`	Répétition de l'image de fond
`background-position`	Position de l'image de fond
`background-attachment`	Attache de l'image de fond par rapport à la page
`background-origin`	Position du fond par rapport à la boîte de l'élément
`background-size`	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
text-shadow	Ombrage du texte
text-outline	Contour du texte
box-shadow	Ombrage d'un élément boîte
opacity	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
border	(Déclaration groupée)
border-color	Couleur du bord
border-spacing	Espacement du bord
border-collapse	Fusion du bord
border-style	Style du bord
border-width	Largeur de la bordure
border-radius	Rayon du bord arrondi
border-image	Style de bord avec image
outline	Contour
outline-style	Style du contour
outline-color	Couleur du contour
outline-width	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
`display`	Mode d'affichage
`float, clear`	Mode flottant
`position`	Positionnement
`z-index`	Ordre de recouvrement « vertical »
`overflow`	Mode de dépassement de bloc
`overflow-y`	Mode de dépassement de bloc vertical
`overflow-x`	Mode de dépassement de bloc horizontal
`width`	Largeur
`height`	Hauteur
`top`	Position par rapport au haut
`left`	Position par rapport à la gauche
`right`	Position par rapport à la droite
`bottom`	Position par rapport au bas
`padding`	Marge interne
`margin`	Marge externe
`min-height`	Hauteur minimale
`max-height`	Hauteur maximale
`min-width`	Largeur minimale
`max-width`	Largeur maximale
`vertical-align`	Alignement vertical
`visibility`	Visibilité
`rotation`	Rotation
`rotation-point`	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
`list-style`	Style de liste
`list-style-type`	Type de puce pour les éléments de la liste
`list-style-image`	Image de puce pour les éléments de la liste
`list-style-position`	Position de la puce

Exemple

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

Tableau 2–9 Autres

Propriété	Rôle
`cursor`	Apparence du curseur
`direction`	Direction d'écriture

Exemple

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

Tableau 2–10 Animations

Propriété	Rôle
`animation`	(Déclaration groupée)
`animation-delay`	Délai d'animation
`animation-direction`	Sens d'animation
`animation-duration`	Durée d'animation
`animation-iteration-count`	Nombre d'itérations
`animation-name`	Nom d'animation
`animation-timing-function`	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
transform	(Déclaration groupée)
transform-origin	Point d'origine de la transformation
transform-style	Style de transformation
perspective	Effet de perspective
perspective-origin	Point d'origine de la perspective
backface-visibility	Visibilité de l'arrière de l'élément transformé

Exemple

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

Tableau 2–12 Transitions

Propriété	Rôle
transition	(Déclaration groupée)
transition-delay	Délai de transition
transition-duration	Durée de transition
transition-property	Propriété à laquelle appliquer la transition
transition-timing-function	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
`:link`	Lien
`:visited`	Lien visité
`:hover`	Élément survolé
`:active`	Élément actif
`:focus`	Élément ayant le focus
`:first-child`	Premier enfant
`:last-child`	Dernier enfant
`:nth-child(n)`	n-ième enfant
`:nth-last-of-child(n)`	n-ième dernier enfant
`:nth-of-type(n)`	n-ième type d'élément
`:nth-last-of-type(n)`	n-ième dernier type d'élément
`:first-of-type`	Premier type d'élément

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

Pseudo-*	Rôle
`:last-of-type`	Dernier type d'élément
`:only-child`	Enfant unique
`:only-of-type`	Élément de type unique
`:checked`	État coché
`:enabled`	État activé
`:indeterminate`	État indéterminé
`:not(expr)`	Négation de l'expression
`::first-letter`	Première lettre
`::first-line`	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
`@import`	Importer une autre feuille de style CSS.
`@charset`	Déclaration de l'encodage des caractères.
`@page`	Définir des règles générales pour les médias paginés.
`@font-face`	Importer un fichier de police (fonte) externe.
`@media`	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

C

Un sage d'Ikea disait : « Un bon design doit toujours rester accessible ». La future recommandation WAI-RAIA 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 propreté 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.

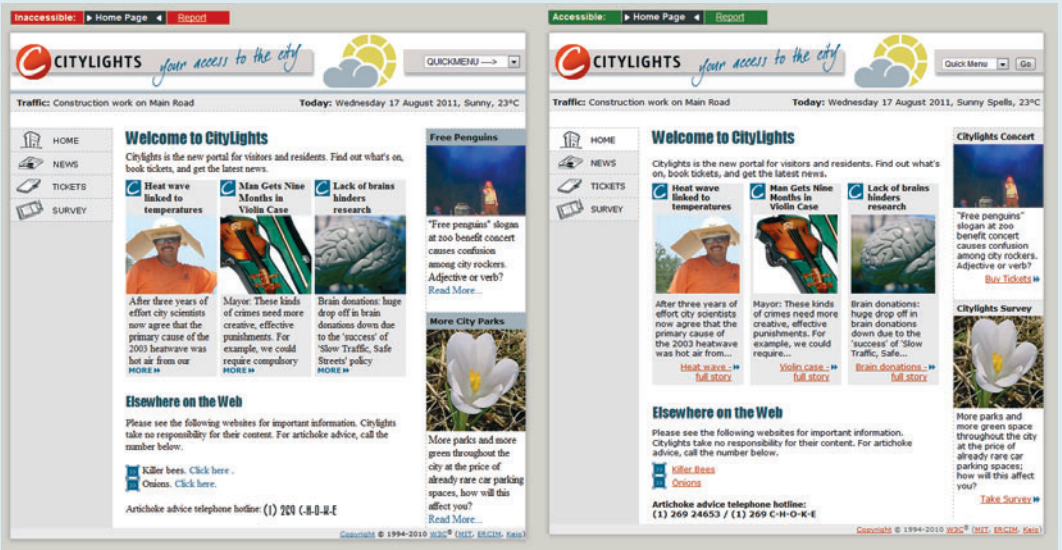


Figure 3–3 Démonstration avant (inaccessible) et après (accessible) http://www.w3.org/WAI/demos/bad/

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 XXIe siècle, beaucoup d'intégrateurs (ainsi qu'Alsacreations.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.

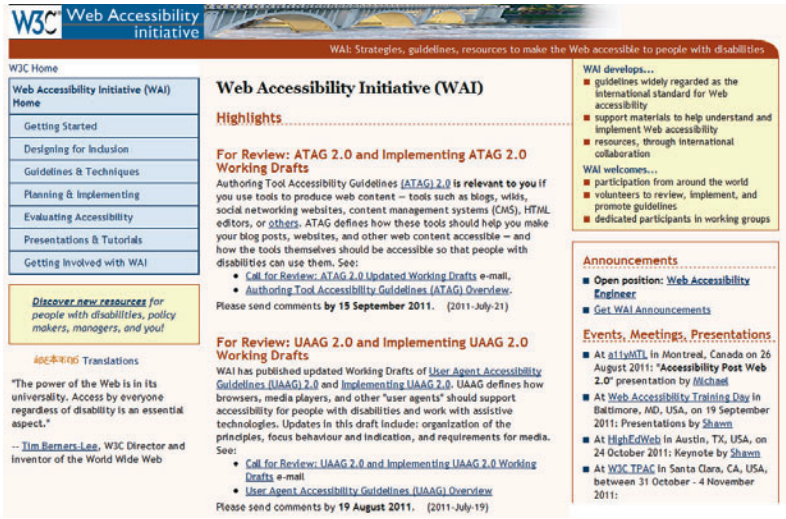


Figure 3–4
Web Accessibility Initiative (WAI)

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

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
`application`	Région correspondant à une application web (à l'inverse d'un document statique).
`banner`	Typiquement, l'en-tête comprenant un logo et éventuellement un outil de recherche.
`complementary`	Section complémentaire à la section principale (main), de même niveau, qui peut rester pertinente lorsqu'elle en est détachée.
`contentinfo`	Région contenant des informations relatives au document.
`form`	Région contenant des objets qui dans leur ensemble constituent un formulaire.
`main`	Contenu principal.
`navigation`	Ensemble de liens de navigation.
`search`	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
`article`	Un article formant une partie indépendante du document.
`columnheader`	Un en-tête de colonne.
`definition`	Une définition.
`directory`	Une liste de références aux membres d'un groupe (une table des matières).
`document`	Une région correspondant à un document (à l'inverse d'une application).
`group`	Un ensemble d'objets qui ne sont pas destinés à être inclus dans un résumé du document ou dans sa table des matières.
`heading`	Un en-tête de section.
`img`	Un conteneur pour un ensemble d'éléments qui forment une image.
`list`	Une liste d'éléments non interactifs.
`listitem`	Un élément dans une liste ou une table des matières *(directory)*.
`math`	Une expression mathématique.
`note`	Une note annexe au contenu principal.
`presentation`	Un élément de présentation dont les rôles implicites ne seront pas relayés à l'API d'accessibilité.
`region`	Une grande partie du document assez importante pour être incluse dans le résumé ou la table des matières.
`row`	Une ligne de cellules dans une grille.
`rowheader`	Un en-tête de ligne de cellules dans une grille.
`separator`	Un séparateur de contenu ou de groupes d'éléments de menus.
`toolbar`	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
`alert`	Un message d'alerte.
`alertdialog`	Un dialogue contenant un message d'alerte.
`button`	Un contrôle d'entrée (un bouton) destiné à déclencher des actions lorsqu'il est cliqué ou pressé.
`checkbox`	Une case à cocher possédant trois états : vrai, faux ou mixte.
`combobox`	Une liste de choix (conteneur).
`dialog`	Une fenêtre de dialogue, usuellement modale, demandant un choix ou une information à l'utilisateur.
`grid`	Une grille (un tableau) pouvant contenir des cellules organisées en colonnes et en lignes (conteneur).
`gridcell`	Une cellule appartenant à une grille.
`link`	Une référence à une ressource externe qui provoque la navigation vers cette ressource lorsqu'elle est activée.
`listbox`	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
`log`	Une région contenant des informations ajoutées au fur et à mesure dont les plus anciennes peuvent disparaître.
`marquee`	Une région dynamique contenant des informations non essentielles qui changent fréquemment.
`menu`	Un contrôle offrant une liste de choix à l'utilisateur (conteneur).
`menubar`	Un menu qui est destiné à rester visible et qui est usuellement présenté à l'horizontale (conteneur).
`menuitem`	Une option parmi un groupe contenu dans menu ou menubar.
`menuitemcheckbox`	Un élément de menu pouvant être coché et possédant trois états : vrai, faux, mixte.
`menuitemradio`	Un élément de menu parmi un groupe d'autres congénères, dont un seul peut être coché à la fois.
`option`	Un élément sélectionnable dans une liste.
`progressbar`	Une barre de progression pour les tâches qui peuvent nécessiter un certain temps.
`radiogroup`	Un groupe de boutons radio (conteneur).
`radio`	Un élément de bouton radio parmi un groupe d'autres congénères, dont un seul peut être coché à la fois.
`scrollbar`	Un contrôle graphique permettant de faire défiler le contenu dans la zone visible
`slider`	Un contrôle permettant à l'utilisateur de sélectionner une valeur parmi un intervalle donné.
`spinbutton`	Un contrôle permettant à l'utilisateur de sélectionner une valeur à intervalles réguliers, parmi un intervalle donné.
`status`	Un conteneur recueillant une information d'état dont l'importance ne justifie pas l'emploi d'une alerte.
`tab`	Un onglet parmi un ensemble.
`tablist`	Une liste d'onglets (`tab`) qui sont des références respectives à des conteneurs (`tabpanel`).
`tabpanel`	Un conteneur pour les ressources associées à un onglet.
`textbox`	Un champ d'entrée texte libre.
`timer`	Un conteneur doté d'un compteur numérique indiquant la quantité de temps écoulé depuis une date, ou restant jusqu'à une date fixée.
`tooltip`	Un pop-up contextuel affichant la description d'un autre élément.
`tree`	Une liste contenant une arborescence de groupes pouvant être réduits et redéployés (conteneur).
`treegrid`	Une grille dont les lignes peuvent être réduites et redéployées (conteneur).
`treeitem`	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">
  <img src="couper.png" role="button" id="btn1" alt="Couper">
  <img src="copier.png" role="button" id="btn2" alt="Copier">
  <img src="coller.png" role="button" id="btn3" alt="Coller">
</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">
<img id="slider" src="curseur.png" role="slider" alt="Coefficient"
aria-valuemin="0" aria-valuemax="100">
```

```
</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
`aria-atomic`	Voir "live"
`aria-busy`	Voir "live"
`aria-controls`	Voir "relations"
`aria-describedby`	Voir "relations"

Tableau 3–4 Attributs ARIA globaux (suite)

État ou propriété	Description
`aria-disabled`	Voir "contrôles d'interface"
`aria-dropeffect`	Voir "drag & drop"
`aria-flowto`	Voir "relations"
`aria-grabbed`	Voir "drag & drop"
`aria-haspopup`	Voir "contrôles d'interface"
`aria-hidden`	Voir "contrôles d'interface"
`aria-invalid`	Voir "contrôles d'interface"
`aria-label`	Voir "contrôles d'interface"
`aria-labelledby`	Voir "relations"
`aria-live`	Voir "live"
`aria-owns`	Voir "relations"
`aria-relevant`	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
`aria-autocomplete`	Indique que des suggestions d'autocomplétion sont fournies à l'utilisateur (en ligne dans le champ, en liste, les deux, ou aucune).	`inline`, `list`, `both`, `none`	`combobox` `textbox`
`aria-checked`	Statut coché ou décoché (état).	`true`, `false`, `mixed`, `undefined`	`option`
`aria-disabled`	Indique que l'élément est perceptible mais désactivé (état).	`true`, `false`	Tous

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

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

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

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

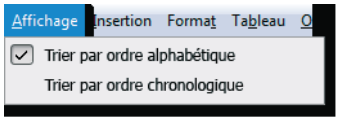
Exemple d'application de propriétés et rôles ARIA

```
<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">
        <img src="checked.png" role="presentation" alt="">
        Trier par ordre alphabétique
      </li>
     <li role="menuitemcheckbox"aria-checked="false">
       Trier par ordre chronologique
     </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
`aria-atomic`	Indique la portée de l'information retournée à l'utilisateur lorsque le contenu d'un élément est modifié.	`true, false`
`aria-busy`	Indique qu'un élément et ses descendants sont en cours de mise à jour (état).	`true, false`
`aria-live`	Indique qu'un élément pourra être mis à jour et décrit le type de mises à jour que les agents utilisateurs peuvent attendre.	`off, polite, asservitive`
`aria-relevant`	Indique quelles modifications pourront être opérées.	Une combinaison parmi les termes clés :`additions`, `removals`, `text`, `all`

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

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

`aria-activedescendant`	Identifie le descendant actif d'un composant.	Identifiant	`composite`
			`group`
			`textbox`

`aria-controls`	Identifie les autres éléments dont la présence ou le contenu sont contrôlés par cet élément.	Liste d'identifiants	
`aria-describedby`	Identifie le ou les éléments qui décrivent l'objet (complémentaire à `aria-labelledby`).	Liste d'identifiants	

Tableau 3–8 Attributs ARIA de relations (suite)

État ou propriété	Description	Valeurs	Rôles
`aria-flowto`	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	
`aria-labelledby`	Identifie le ou les éléments qui confèrent une étiquette à l'objet (similaire à `aria-label`).	Liste d'identifiants	
`aria-owns`	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	
`aria-posinset`	Définit la position d'un élément dans un ensemble d'autres éléments de type `listitem` ou `treeitem`.	Nombre entier	`listitem` `option`
`aria-setsize`	Définit le nombre total d'éléments dans l'ensemble courant de type `listitem` ou `treeitem`.	Nombre entier	`listitem` `option`

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="applicaton" 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>
  <img src="firebug.png" alt="Capture d'écran de Firebug"
aria-describedby="legende1">
```

```
  <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 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.

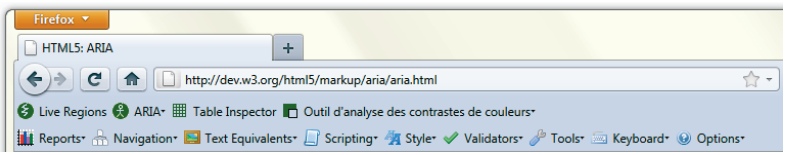


Figure 3–6
Extensions pour Firefox

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
• role="navigation"	DIV	• HTML • BODY	`<div role="navigation">`
• role="banner"	H1	• HTML • BODY • DIV • DIV#header • DIV#header-inside	`<h1 role="banner">`

• role="search"	FORM	• HTML
		• BODY
		• DIV
		• DIV#sous-menu
			`<form method="post"`
			`action="/search/"`
			`role="search"`
			`id="recherche">`

• role="main"	DIV	• HTML • BODY • DIV#global • DIV#page	`<div role="main"` `id="content">`
• role="complementary"	DIV	• HTML • BODY • DIV#global • DIV#page	`<div role="complementary"` `id="sidebar">`
• role="contentinfo"	DIV	• HTML • BODY • DIV#global • DIV#page	`<div role="contentinfo">`

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

Index

C

D

E