Les technologiesdu Web

Des standards pour échanger de l'information

Toulibre
Jean-Sébastien Kroll-Rabotin & Emmanuelle Helly

Tim Berners-Lee in *TechnologyReview*, July 1996

> Anyone who slaps a 'this page is best viewed with Browser X' label on a Web page appears to be yearning for the bad old days, before the Web, when you had very little chance of reading a document written on another computer, another word processor, or another network.
>
> Toute personne qui colle une étiquette « cette page est optimisée pour un navigateur X » sur une page Web semble aspirer à un retour aux vieux jours d'avant le Web, quand on avait peu de chance de lire un document écrit sur un autre ordinateur, un autre éditeur de texte, ou un autre réseau.

Plan de la présentation

- Histoire du Web
- Les langages et formats du Web
  - La sémantique du web
  - Mise en page et design Web
  - Interactivité et contenu dynamique
- Outils complémentaires
  - Compléments sur l'accessibilité
  - Mettre son site internet en ligne
  - Système de gestion de contenu
- Exercices pratiques
- Liens utiles

La présentation commence par une partie d'introduction du contexte historique de l'apparition du Web qui donne une définition du terme « Web », puis présente les trois langages majeurs du Web historique (que l'on appellerait aujourd'hui Web 1.0), à savoir (x)HTML, CSS et ECMAscript. Quelques exercices pratiques viennent apporter des exemples concrets pour mettre en œuvre ces langages avant de présenter d'autres aspects courants du Web et de son avenir qui ne seront pas abordés en détail ici. Une dernière partie référence un ensemble de liens très utiles pour se documenter sur le Web et les bonnes pratiques à suivre quand on développe pour le Web.

Un chapître est intitulé la « sémantique du web ». Il traite de l'intégration du sens dans la structuration des documents web et renvoie à l'usage linguistique du terme « sémantique » et non à l'acception usuelle de « web sémantique » dont le sens a dérivé pour ne définir que la portion du web destinée à être interprété par des traitements automatisés.

Naissancedu Web

- Avant le Web : « Internet » !
  - Un réseau mondial
  - Des protocoles d'échange d'information
    - Mail
    - Usenet
    - FTP
- **1990** : Apparition du **lien hypertexte**
  - Invention de l'URI, de HTTP et de HTML par Tim Berners-Lee (au CERN)

  - URI
    Uniform Resource Identifier : Adresse web
    HTML
    HyperText Markup Language : Format de page Web
    HTTP
    HyperText Transfer Protocol

Internet est parfois appelé « le réseau des réseaux ». C'est un réseau d'ampleur planétaire qui ne se distingue du réseau local entre joueurs d'une LAN que par sa portée beaucoup plus large. Bien sûr, il est techniquement impensable de relier autant de machines avec l'infrastructure d'une LAN, c'est pourquoi Internet prend en fait plutôt la forme d'un réseau de réseaux. Cependant, le principe est bien le même : relier de multiples machines entre elles et leur permettre de communiquer via ces connections.

Sur Internet, comme dans tout autre réseau, toute machine qui propose des services à celles qui s'y connectent est appelée « serveur ». Toute machine du réseau peut à la fois être serveur et client pourvu qu'elle sache communiquer avec les autres pour fournir ou utiliser des services. Pour communiquer, les machines échangent suivant des protocoles standardisés. Avant le Web, d'autres protocoles pré-existaient, par exemple pour l'échange de courriers électroniques, la diffusions de « news » ou le transfert de fichiers.

Le Web est né de l'idée (géniale) de Tim Berners-Lee de transposer le lien hypertexte à Internet. Il a donc inventé un format permettant d'associer des méta-données à des éléments de la page, ainsi qu'une méthode d'identification des ressources. Avec ces deux concepts essentiels, il devenait alors possible d'associer l'identifiant d'une ressource à un élément d'une page web, créant de ce fait un lien de l'élément vers la ressource. Quand cette ressource est une autre page comprenant des liens, elle lie alors à son tour de nouvelles ressources et devient un nœud du réseau de ressources. Ce réseau extensible en sautant d'un nœud à l'autre ressemble alors à une toile d'araignée, d'où l'appellation de « Web ».

Le format permettant d'associer des méta-données au contenu est le HTML. L'identifiant d'une ressource est appelé URI et prend différentes formes selon la méthode d'accès à la ressource.

Exemples :

- ftp://ftp.is.co.za/rfc/rfc1808.txt
- http://www.ietf.org/rfc/rfc2396.txt
- ldap://[2001:db8::7]/c=GB?objectClass?one
- mailto:John.Doe@example.com
- news:comp.infosystems.www.servers.unix
- tel:+1-816-555-1212
- telnet://192.0.2.16:80/
- urn:oasis:names:specification:docbook:dtd:xml:4.1.2

Les URI de ressources Web (HTTP), sont appelées URL et prennent la forme http://[nom DNS de la machine hôte]/[chemin vers la ressource]

Évolutionvers le « graphique»

- Les premiers navigateurs en mode texte sont développés par le CERN.
- **1992** : la balise **<IMG>**, insertion d'images
- **1993** : **MOSAIC**, le premier navigateur graphique « grand public »
- **1994-1995** : MOSAIC est repris dans **Netscape Navigator** et **Microsoft Internet Explorer**
- La guerre des navigateurs commence et de nombreuses nouvelles balises apparaissent, notamment pour la mise en page :
  - <CENTER>, <TABLE>, <BACKGROUND>, <APPLET>, <FRAME>, <EMBED>, <SCRIPT>, <STYLE>, <LAYER>, <DOCTYPE>

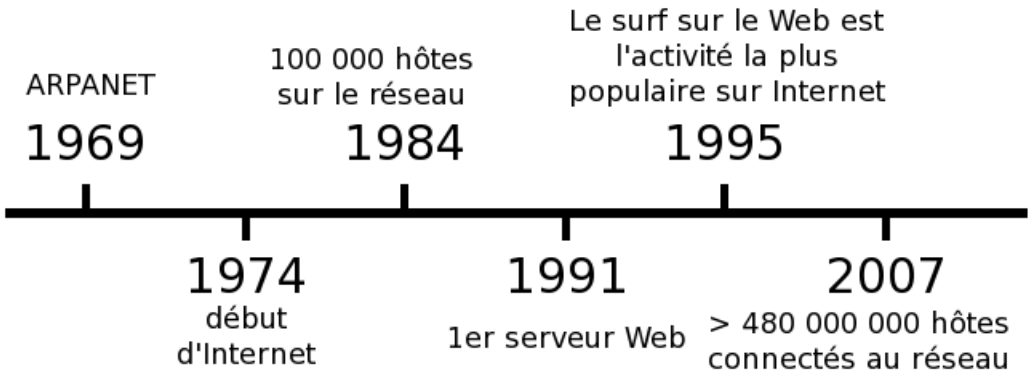
Parmi les méta-données que le langage HTML permet s'associer au texte, il y a notamment le découpage en paragraphes et la hiérarchie des titres.

Avec l'évolution du matériel informatique et des débits accessibles sur Internet, le Web se met à transporter des images en plus du texte, et l'inclusion d'images à l'intérieur des pages Web arrive très vite. Le confort qu'apporte le rendu graphique des pages Web par rapport au mode textuel fait que l'accès au Web ne se fait quasiment plus que par des navigateurs graphiques. Dès lors, le sens des balises du langage HTML dérive de la description de méta-données structurelles associées au texte, vers l'aspect graphique qu'elles produisent.

Différents navigateurs concurrents arrivent sur le marché et cherchent à se distinguer les uns des autres en se lançant dans une course à la richesse graphique de leurs rendus. Ils implémentent de nouvelles balises qui n'associent plus de méta-données structurelles mais uniquement des informations sur comment les éléments de la page doivent être affichés. Des balises de mise en page (comme <CENTER> et <FRAME>) servent à positionner les éléments, d'autres permettent d'associer des styles et des images aux éléments (comme <BACKGROUND>, <STYLE> puis plus tard <FONT>), d'autres encore sont détournées de leur sens structurel pour utiliser leur capacité de positionnement dans la page (comme <TABLE>).

Après avoir permis l'inclusion d'images dans les pages, le Web s'enrichit de nouvelles inclusions dans ses pages grâce à des balises (comme <APPLET> et <EMBED>) qui permettent d'utiliser des applications extérieures au navigateur pour afficher des éléments que le HTML et le navigateur ne savent pas décrire seuls. C'est notamment utilisé pour créer des pages interactives, si bien que les navigateurs eux même vont ensuite embarquer des interpréteurs pour des langages de script afin d'être capable d'afficher des pages interactives sans application externe. Ces scripts s'insèrent directement dans la page Web à l'aide de balises spéciales (comme <SCRIPT> quelle surprise !).

Internet et le Web



- Avec le développement d'Internet, le Web devient accessible à plein de monde et représente un marché croissant.
- Le « surf » devient l'activité favorite sur Internet.
- Au fur et à mesure qu'Internet devient accessible au public, « Web » et « Internet » sont de plus en plus confondus.

« Balkanisation» du Web

- **1994-1996**, le règne de **Netscape**
  Netscape domine le marché
  Développement du Web « graphique ».
  Autres navigateurs : Opera, Internet Explorer
- **1996-1998**, la grande guerre
  IE menace **NS**
  Chacun implémente ses propres balises
  □ HTML incompatible
- **1998-2002**, IE a gagné
  Peu d'évolution
- Conséquence de cette guerre : HTML devenu une « soupe de balises » (ou « tag soup »)

De 1994 à 1996, un navigateur issu de MOSAIC, Netscape, domine le marché des navigateurs Internet et développe le Web « graphique » (par opposition à « sémantique »). D'autres navigateurs apparaissent et coexistent avec lui (Opera, Internet Explorer).

De 1996 à 1998, la grande guerre commence entre NS et IE qui est vendu avec le système Windows. Pour se démarquer, chacun implémente ses propres balises (éventuellement brevetées) permettant de plus en plus d'effets visuels et d'interactivité. Les balises reconnues par les deux navigateurs diffèrent de plus en plus et maintenir une page en HTML pour les deux navigateurs devient un travail fastidieux, à tel point que beaucoup de mainteneurs de sites Web font le choix de ne plus écrire leurs pages que pour l'un des deux qu'ils choisissent en fonction de celui qui leur semble le plus utilisé ou le plus apte à rendre les effets qu'ils souhaitent ... et plein d'autres critères très subjectifs. Il reste très peu de place aux navigateurs concurrents : Opera survit, mais difficilement.

Autour de 1998, la politique de vente liée de Microsoft pour son navigateur IE a eu raison de Netscape, et IE s'impose comme le navigateur quasi-unique du paysage. Comme la concurrence a presque disparue, Microsoft ne propose que très peu d'évolutions à son navigateur pendant plusieurs années, et comme le Web qui subsiste est écrit pour IE, le Web aussi stagne.

Le Web a beaucoup été transformé par la course aux effets visuels entre IE et NS : le HTML qui décrit les pages n'est plus un format d'échange mais juste une « soupe de balises » (ou « tag soup ») destinée à un navigateur unique pour un rendu graphique. Les méta-données sémantiques du HTML d'origine ont été complètement détournées de leur usage.

Et pourtant...

- Dès **1994**, normalisation du langage HTML confiée à un consortium : le **W3C**.
- **HTML** : format de données structurées associées à des méta-données qui complètent la signification
- Séparation du contenu et de la forme
  - En **1996** : CSS, des feuilles de style pour les documents HTML
- En **2002** : apparition du navigateur **Firefox**
  - **Firefox** et **Opera** implémentent les recommandations du W3C.

Dès 1994, la normalisation du langage HTML avait été confiée au W3C qui cherche à préserver le sens des balises et choisit de séparer l'information contenue au format HTML de sa présentation à l'écran décrite dans des fichiers annexes, appelés « feuilles de style ».

Un autre format que le HTML sert à décrire les feuilles de style et donne lieu à un nouveau standard, les feuilles des style en cascade (ou « Cascading Style Sheets » abrégé en « CSS »).

Depuis 2002, Firefox et Opera profitent de la stagnation d'IE et se démarquent de celui-ci en implémentant les recommandations du W3C. D'autres navigateurs sont apparus, comme Safari (pour le système MacOS depuis que IE pour Mac n'est plus développé) et une multitude de navigateurs libres qui foisonnent sous GNU/Linux. Les recommandations du W3C servent de véritables standards portés par tous ces navigateurs.

État actuel du Web

- IE toujours le plus utilisé
- Mais **Firefox**, **Opera** et les autres font concurrence
- Multiplication des terminaux d'accès au Web
  - téléphones, Palm™, platines de salon, frigo, ...
  - **nécessité** des standards !
- IE est toujours en position dominante **mais** menacé **et** sous pression pour respecter les standards
- Maintenir un site standard **et** compatible IE
  → travail accessible à un humain moyen

IE est encore le navigateur le plus utilisé dans le monde (il est toujours vendu avec le système Windows, lui même vendu avec presque tous les matériels informatiques neufs pour les particuliers) mais Firefox, Opera et les autres occupent désormais une part du marché non négligeable face à IE.

Les terminaux reliés à Internet sont de plus en plus divers (et tendent même souvent vers le n'importe quoi) et n'ont pas les mêmes capacités d'affichage (résolutions, jeux de couleurs, ...) voire ne sont pas destinés à l'affichage (lecteurs braille, synthétiseurs vocaux, moteurs de recherche, etc ...). Il est donc nécessaire de séparer le contenu de la présentation, et d'utiliser un standard compatible avec les contraintes de chaque terminal. Des nouveaux appareils sont mis sur le marché tous les jours, il est impossible d'offrir un document adapté à chacun d'eux individuellement, par contre il est possible de les faire tous parler le même langage.

Microsoft cherche toujours à assoir la position dominante de son navigateur IE en adaptant les recommandations du W3C à sa sauce et de manière incompatible avec les standards, cependant, malgré ce retard en partie volontaire, la pression de la concurrence l'a forcé à améliorer énormément son support des standards, notamment dans les versions les plus récentes de IE.

Définitiondu Web sémantique

- Se concentrer sur le contenu et la **signification** des balises :
  - **méta-données** : informations que le texte pur ne peut pas contenir
    liens, structure (titres, paragraphes, listes, ...), langue, ...
  - **alternatives** aux données non-textuelles
- Un document **identifiable** de manière **unique** contient **toute** l'information et **rien que** l'information
  - une URL
  - de l'information « facile » à interpréter, complète et accessible ...
  - limiter le « bruit »
- Utilisation des **standards** du W3C
  - HTML
  - XHTML

Pour que l'information soit accessible par tous les utilisateurs et tous les programmes indépendamment de leur capacité d'affichage, elle doit être séparée et stockée séparément des règles de présentation de la page. Par contre un maximum (toute ?) d'information utile à la bonne compréhension/ interprétation du contenu doit être présent.

Les données considérées les plus accessibles sont celles exprimées sous forme de texte. Le texte est lisible par des programmes, et son affichage peut être adpaté au matériel et à l'utilisateur (les exemples les plus parlants sont les cas de la synthèse vocale et des afficheurs en Braille).

Les formats du Web

- **HTML**, sous-langage de **SGML**
  - hérite du balisage entre « < » et « > »
  - SGML très complexe aucun navigateur HTML n'est un bon interpréteur SGML
- **XHTML**, sous-langage de **XML**
  - XML, simplification de SGML
  - balisage plus simple et strict, possibilité d'écrire des navigateurs capables d'interpréter du XML strictement
  - XHTML : standardisation des balises utiles du HTML dans un langage XML
  - bien adapté à l'élaboration d'un standard « strict »

Standard Generalized Markup Language (langage normalisé de balisage généralisé, SGML) est un langage de description à balises (norme ISO 8879:1986) qui sert à structurer des données indépendamment de leur présentation. Il était très utilisé au CERN pour stocker leurs documents et Tim Berners-Lee s'en est donc inspiré pour formaliser son langage HTML pour le Web.

SGML est, comme son nom l'indique, un langage très généraliste et complexe dont les capacités excèdent largement les besoins du Web. De ce fait, seule une partie des possibilités de SGML sont exploitées et comprises par les navigateurs. L'interprétation laxiste du HTML et la course aux balises de IE et NS ont rendu l'usage du HTML très lointain de son formalisme initial et pour éviter cet écueil, le W3C a décidé de reposer ses standards sur un autre langage à balises, le XML. Le XML est un SGML simplifié dont les formats de description de données du W3C sont des sous-langages, c'est pourquoi le W3C a proposé un autre format que le HTML, le XHTML qui couvre la même fonction mais avec un formalisme XML.

Cependant XML et SGML se ressemblent beaucoup, et XHTML et HTML se ressemblent tout autant, ainsi traduire une page HTML standard en XHTML est une tâche très aisée, souvent même réalisable automatiquement par un programme informatique. La véritable difficulté qui est apparue après la période de « balkanisation » du Web pour traduire le HTML d'alors en XHTML standard a été que le HTML largement utilisé n'était pas du tout compatible avec le HTML formalisé par le W3C. Les habitudes des mainteneurs de sites Web s'étaient elles aussi très éloignées des concepts de SGML comme la séparation du contenu et de la présentation.

L'approche exclusive du Web par son aspect rendu dans un navigateur graphique a notamment conduit les concepteurs de sites à utiliser des programmes qui faisaient le travail de produire les balises pour la mise en page (Frontpage, Dreamweaver, etc ...) mais comme ces programmes sont incapables de comprendre le sens du contenu de la page, ils produisent un balisage dénué de sens : un exemple courant est l'utilisation de tableaux pour la mise en page sur différentes colonnes. Cette approche n'est pas encore complètement oubliée et la transition est lente (et ralentie par l'attitude d'IE et des éditeurs de logiciels historiques pour le Web), cependant même si la compatibilité stricte avec les standards est rarement atteinte sur les sites Web, il y a eu beaucoup d'évolution sur la séparation contenu/présentation.

La relative simplicité du XML et du XHTML ne profite pas qu'aux navigateurs qui sont capable de mieux interpréter les pages Web mais aussi aux concepteurs qui peuvent désormais assez aisément écrire eux-mêmes leur propre code HTML « à la main » et maîtrisent donc le sens des balises qu'ils utilisent. Ainsi, la richesse du XHTML n'est pas limitée par la connaissance qu'un programme a des balises, mais par celle du concepteur qui peut s'étendre bien plus aisément au fur et à mesure qu'il apprend le langage.

Structure d'un document

- **DOM** : éléments constitutifs de la page organisés en **arborescence** d'objets
- Objet ⇔ Élément de [**structure** | **données** | **méta-données**]
- DOM partagé par tous les langages (HTML, CSS, scripts, etc ...)
- DOM normalisé : relations parent/enfants entre éléments définis dans une DTD
- (X)HTML : éléments ⇔ balises <...>

Le Document Object Model (DOM) est une représentation des objets indépendante du format ou du langage de programmation, il défini la structure d'un document et prend la forme d'une arborescence d'objets. À la base de cette arborescence il y a un objet racine unique qui contient le document dans son ensemble, en (X)HTML c'est l'objet <html>. Tous les autres objets sont affiliés à un parent et un seul et contiennent différents types d'information :

- des informations sur la structure du document (titres, paragraphes, etc ...)
- du contenu (texte, images, etc ...)
- des informations sur le type de contenu, sa signification ou toute information complémentaire du contenu

En fait, le DOM est une représentation du contenu du document, donc la frontière entre ces types d'information est ténue : les informations de structure définissent l'organisation du contenu lui-même, et les méta-informations sur le contenu sont structurantes. Le DOM définit aussi les relations possibles entre objets, c'est à dire que chaque objet ne peut avoir pour enfant que des objets de certains types et les règles en la matière respectent la cohérence de leurs significations. Par exemple un objet « titre » ne peut pas contenir un objet « paragraphe », par contre tous deux peuvent contenir un objet « abréviation ». Ces règles sont définies dans une Définition de Type de Document (DTD). Il existe une DTD pour chaque version ou variante de (X)HTML. Ce n'est pas à l'auteur du document de rédiger sa DTD, celle-ci est imposée par le langage qu'il utilise, il doit par contre s'y conformer.

Chaque élément est représenté dans le fichier (X)HTML par une balise entre chevrons et réciproquement. Il n'y qu'un élément qui n'est pas écrit entre chevrons, c'est l'élément de texte qui de toute façon ne peut pas être parent d'un autre élément. Il est forcément en fin de branche dans l'arborescence du DOM.

Exemple d'un document XHTML simple

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml">
4  <head>
5  <title>Titre du document</title>
6  <meta http-equiv="Content-Type" content="application/xhtml+xml;
       charset=UTF-8" />
7  </head>
8  <body>
9  <h1>Gros titre</h1>
10 <p>Lorem ipsum <a href="dolor.html">dolor</a> sic amet ...</p>
11 </body>
12 </html>
```

Dans l'exemple de code XHTML précédent, on retrouve :

1. déclaration XML (XHTML, est un sous-langage de XML)
2. déclaration de la DTD utilisée, ici XHTML 1.0 Strict
3. élément racine <html>
4. début de l'élément en-tête
5. titre du document
6. objet <meta> pour associer des méta-données au document. Ici ces méta-données indiquent comment les agents utilisateur doivent interpréter le document.
7. fin d'élément en-tête
8. début de corps de document
9. élément titre de premier niveau
10. élément paragraphe contenant un élément lien
11. fin de corps de document
12. fin d'élément racine <html>

Quelques règles de base sur le XHTML :

- Les espaces et les retour à la ligne sont seulement des séparateurs de mots. Leur nombre n'importe pas, deux mots séparés par plusieurs espaces et retours à la ligne dans le code correspondent à deux mots (distincts) à la suite l'un de l'autre dans le flux d'information à traiter. Le texte brut du code n'est donc pas contraint à une mise en page, celle-ci étant du domaine de la forme, elle est spécifiée ailleurs.
- Les éléments sont représentés par deux balises, l'un ouvrante <...> et l'autre fermante </...>. Les éléments enfants sont déclarés à l'intérieur de ces deux balises. Toutefois, pour les éléments qui n'ont pas d'élément enfant, les deux balises peuvent être regroupées en une seule de la forme <... /> qui est à la fois ouvrante et fermante (cf. l'élément « meta dans l'exemple de code).
- Certains éléments ont des propriétés qui sont définies à l'intérieur de la balise ouvrante, ces propriétés sont donc attribuées à l'élément en question et ne sont donc pas des éléments enfants. Ces propriétés n'ont rien à voir avec des éléments, elles sont propres au type d'élément auxquelles elles sont attribuées. On les appelle aussi attributs, certains attributs sont facultatifs et alimentent l'information qu'apporte l'élément, d'autres sont obligatoires. Par exemple, l'attribut « href » de l'élément « a » contient l'adresse pointée par le lien.
- Deux éléments fondamentaux sont des enfants directs de <html> : ce sont <head> et <body>. Ils séparent le contenu du document des méta-données associées à celui-ci. En effet, le (X)HTML permet d'associer des méta-données à des éléments de la page et à des portions de texte, mais aussi d'en associer au document lui-même. Tout ce qui concerne le contenu de page est donc enfant de <body>, alors que ce qui se réfère au document (X)HTML dans son ensemble et en tant que document complet est enfant de <head>. C'est notamment dans les « headers » (en-têtes) qu'il sera possible d'associer des styles de présentation au document.

Quelques éléments importants

<h1>, **<h2>**, **<h3>**, **<h4>**, **<h5>**, **<h6>**
: Titres : 6 niveaux de titres dans la DTD XHTML

<p>
: Paragraphe

<a>
: Ancre : association de méta-données de la forme URL
Identification : `<a id="toto">` (URL : http://mon.site.web/page.html#toto)
Lien : `<a href="http://...">`. URL absolue ou relative.

<img>
: Image : uniquement depuis un navigateur graphique texte alternatif obligatoire !
`<img src="swirl.jpeg" alt="Logo de Debian" longdesc="swirl.html" />`

Hormis les quelques éléments essentiels à la définition du document et ses en-têtes, une petite poignée d'éléments suffit à structurer le contenu d'une page Web. Six niveaux de titres sont disponibles pour hiérarchiser les différentes sections du document. À noter : il n'y a pas d'élément "section" dans l'arborescence du DOM qui permettrait de lier des sections par des relations parent/enfants, ce sont seulement les titres qui renseignent sur l'imbrication des sections entre elles. Entre les titres, le texte peut être organisé en paragraphes, ceux-ci ne s'imbriquent pas non plus et ne définissent donc bien qu'un seul paragraphe. L'absence d'élément "section" se comprend aisément, car si l'arborescence du DOM représente l'ensemble du contenu d'un document, il ne faut pas oublier que le (X)HTML est tout d'abord fait pour transcrire de l'information par et pour des humains, hors les textes écrits usuels se lisent à la suite et se parcourent comme un flux linéaire d'information, par contre le saut direct à un titre ou d'un paragraphe à l'autre est une manière courante de se déplacer à des endroits précis dans ce flux. En somme, un document Web ne doit pas perturber la façon usuelle d'interpréter l'information que l'utilisateur reçoit, et l'arborescence DOM n'est que la représentation technique de la structure des éléments.

À la base du Web, il y a l'hyperlien. Il permet d'associer à un groupe d'éléments une URL, soit pour pointer vers celle-ci, soit pour être identifié avec celle-ci. Les hyperliens associés à des ancres (élément <a>) sont ce que les navigateurs graphiques traduisent par le chargement de la page pointée suite à un clic sur un élément de l'ancre. L'appellation "ancre" vient du fait qu'elle permet littéralement d'accrocher une URL à un élément de la page. L'URL de cet élément est alors celle de la page suivie du nom de l'ancre, séparés par le caractère dièse « # ».

Le Web étant couramment parcouru à l'aide de navigateurs graphiques, il est courant de l'utiliser pour diffuser des images, il y a donc un élément dont le contenu affiché est défini par une URL pointant vers une image. En raison de son aspect graphique, cet élément <img> doit toutefois être utilisé de manière réfléchie pour respecter la séparation contenu/présentation du Web.

- L'image insérée dans la page y est en fait seulement liée, l'information est contenue dans un fichier séparé qui possède sa propre URL. C'est l'URL qui est décrite par l'élément <img> et non le contenu réel de l'image, il faut donc veiller à ce que l'image reste toujours disponible à la même URL sinon elle ne sera pas accessible dans la page Web.
- Le document (X)HTML est réservé au contenu, pas à la mise en forme ni à la présentation. Les images qui sont insérées dedans ne doivent donc être que celles volontairement diffusées comme du contenu. Par exemple, sur le site Web d'une société, il peut être normal d'insérer le logo dans les informations contenues dans la page au même titre que le nom et l'adresse de la société. Une autre utilisation courante de cet élément est l'insertion de vignettes dans une page qui permettent de télécharger des images. Ces vignettes sont donc des liens (ancres) parents d'un élément <img> qui contient une miniature de l'image pointée par le lien. La miniature représente alors réellement l'image pointée et donne bien une information sur du contenu.
- D'une part le contenu image n'est pas réellement inséré dans la page, d'autre part une image n'est pas une forme d'information facile d'accès. Même si l'usage de navigateurs graphiques est courant, ils ne sont pas le seul moyen de parcourir le Web et beaucoup d'autres de ces moyens sont incapables de retranscrire l'information graphique. Pour que tout le contenu soit dans le document XHTML, il faut donc impérativement spécifier des alternatives pour ceux qui n'ont pas accès à l'information contenue dans les images. L'attribut « alt » de l'élément <img> est donc obligatoire et doit contenir une information textuelle de remplacement pour l'image au cas ou celle-ci ne serait pas accessible (mauvais fichier à l'URL indiquée ou information inutilisable par le navigateur). Si l'image est très complexe ou si elle contient des informations essentielles à la compréhension du document, il est même recommandé d'ajouter un attribut « longdesc » qui contient l'URL d'un document texte descriptif de l'image qui rend cette information accessible même sans approche graphique.

Des éléments génériqueset spécialisés

- Éléments génériques :

  ** **
  : élément « bloc » générique

  ****
  : élément « en ligne » générique

- Éléménts attachés à un sens particulier :
  - **<abbr>**, **<acronym>** : sigles et abréviations
  - **<q>**, **<blockquote>** : citations
  - **<ul>**, **<ol>**, **<li>**, **<dl>**, **<dt>**, **<dd>** : listes
  - **<table>**, **<tr>**, **<td>** : tableaux
  - ****, **** : emphase
  - etc ...

Les balises sémantiques découpent le contenu en paragraphes et permettent de hiérarchiser les titres, cependant elles n'imbriquent pas les sections les unes dans les autres. Pourtant il peut parfois être utile de décrire une arborescence de sections notamment si l'on veut faire hériter aux enfants des propriétés des parents, ou si l'on veut identifier une section en fonction de son niveau d'imbrication. Pour cela, il existe un élément (X)HTML générique qui n'apporte pas de sens supplémentaire aux enfants qu'il englobe mais permet de grouper tous types d'éléments dans un même parent : c'est l'élément . De même pour englober des portions des texte ou spécifier des méta-données génériques à un élément texte, il existe l'élément .

La distinction technique entre et implique de comprendre les notions d'élément bloc et d'élément en-ligne qui ne sont pas abordées ici en détail. Cependant il est important de savoir que tous deux n'admettent pas les mêmes types d'éléments parents et enfants. La compréhension de leur sens suffit néanmoins à déterminer leur usage. Par exemple, il est incohérent de vouloir faire d'une , un enfant de <abbr> par contre pourquoi pas un .

Les éléments <abbr> et <acronym> servent à définir des sigles et des abréviations. <acronym> devrait être réservé aux sigles prononçables mais comme cette définition varie d'une langue à une autre et qu'elle a plus trait à la manière de présenter le contenu qu'au type de contenu lui-même, cet élément n'a pas un rôle très clair et peut être abandonné au profit d'un usage plus large de <abbr>. D'ailleurs <acronym> a disparu dans les propositions de nouvelles versions pour XHTML.

Des éléments sont réservés à l'identification de citations, et comme et possèdent une variante en-ligne et une variante bloc. L'élément en-ligne <q> ne doit pas être redondant avec l'usage de guillemets. Normalement les navigateurs devraient pouvoir automatiquement ajouter les guillemets en fonction de la langue du document et de la citation. Comme la balise <blockquote> historique n'ajoutait pas de guillemets, celle définie dans le standard XHTML reste compatible avec l'ancienne, et l'ajout de guillemets autour des blocs de citation est toléré. En pratique, les navigateurs supportant correctement les standards permettent de spécifier l'ajout de guillemets grâce aux styles (CSS) et cette méthode est a priori mieux adaptée pour séparer contenu et mise en forme.

Deux types de listes coexistent en (X)HTML, les listes ordonnées (ordered <ol>) et sans ordre (unordered <ul>). Leur aspect peut être déterminé par des styles et il est possible d'utiliser des chiffres ou des symboles comme puces pour les deux types, donc le choix de l'une ou l'autre ne doit reposer que sur l'importance de l'ordre des éléments de la liste. Par exemple, les étapes d'un manuel peuvent être structurées dans une liste <ol>, alors qu'une liste de courses correspond bien à une liste <ul>. Les items de ces listes sont définis par l'élément <li>. D'autres éléments permettent de définir des listes spécifiques adaptées aux définitions.

Des éléments sont aussi destinés à la structuration de données tabulaires, certains encore permettent de mettre en relief les portions de texte important ou d'indiquer une emphase (très importante en anglais notamment). Il existe un grand nombre de balises qui essaient de parcourir l'ensemble des besoins spécifiques pour identifier les différents types de contenus. La liste et l'usage de chacune est soumis à des règles spécifiées dans les standards (X)HTML et se trouve parmi les documents produits par le W3C. Pour les types de contenu très spécifiques qui n'ont pas d'élément adapté, l'usage d'éléments génériques peut pallier à ce manque, toutefois un certain nombre d'usages sur le Web définissent déjà des tolérances sur le sens des éléments de (X)HMTL pour permettre de structurer au mieux les documents particuliers.

Mise en page et syntaxe CSS

- **CSS**, langage dédié à la présentation :
  - Accès aux éléments du **DOM** par des **sélecteurs**
  - Exemple :

```
body {
  color: #000;
  background-color: rgb(255, 255, 255);
  background-image: url("mon_image.jpg");
  background-repeat: no-repeat;
}
li {
  font-size: 1.1em;
}
ul > li {
  /* les éléments <li> enfants de <ol> ne sont pas affectés */
  font-size: 1.2em;
}
```

Les feuilles de styles peuvent être utilisées pour décrire des rendus très divers destinés à des supports multiples. Il est par exemple possible de spécifier des styles à destination de synthétiseurs vocaux, mais la suite n'abordera dans les exemples que des styles graphiques, et de fait le vocabulaire utilisé est celui de la conception graphique (mise en page). De même, CSS peut être utilisé pour mettre en forme d'autres documents que du (X)HTML, mais ici seul (X)HTML sera abordé.

CSS est le langage de description de la présentation d'un page Web. Il permet de sélectionner des éléments du DOM et d'y associer un ensemble de consignes d'affichage qui constituent le style de ces éléments. Un sélecteur CSS peut identifier plusieurs éléments, et un élément peut correspondre à plusieurs sélecteurs. Le style d'un élément peut donc être défini à plusieurs endroits donc certaines des consignes d'affichage peuvent être redondantes, voire contradictoire. C'est la dernière consigne définie qui sera retenue pour afficher l'élément, ce qui permet de définir des styles génériques et de les écraser pour certains éléments pour leur donner un aspect particulier.

La syntaxe CSS associe un sélecteur à un bloc de consignes d'affichage. Les propriétés et les constructions invalides syntaxiquement doivent être ignorées au moment de l'interprétation de la feuille de style. Une construction valide est constituée d'un sélecteur suivi d'un bloc contenant une ou plusieur règles CSS séparées par des points-virgules « ; ». Chaque règle contient le nom d'une propriété CSS séparé de la valeur qui y est associée par deux points « : ».

Les sélecteurs

`A B`
: Sélecteur descendant, tout élément B descendant de A

`A > B`
: Sélecteur d'enfant, tout élément B enfant de A

`A + B`
: Sélecteur adjacent, tout élément B suivant un élément A

`A[att]`, `A[att="val"]`, `A[att~="val"]`, `A[att|="val"]`
: Sélecteur d'attribut : existence, égalité, correspondance, premier mot

`*`
: Sélecteur d'élément quelconque, peut être sous-entendu

`A, B`
: Liste de sélecteurs (ou inclusif)

Différents opérateurs permettent de définir des sélecteurs plus ou moins complexes pour parcourir et sélectionner des éléments du DOM. La descendance s'exprime en séparant les sélecteurs par des blancs (espaces). La descendance directe (rapport parent enfant) est s'exprime avec l'opérateur « > », et l'adjacence de deux éléments enfants d'un même parent avec « + ».

Il est possible de sélectionner des éléments en fonction de leurs attributs. L'opérateur crochet « [ ] » désigne un attribut, si celui-ci est défini pour l'élément testé, alors le sélecteur est appliqué, sinon l'élément n'est pas sélectionné. Des opérateurs internes aux crochets permettent des tester la valeur de l'attribut :

- = teste l'égalité stricte. Par exemple `img[alt="logo"]` sélectionne tout élément <img> dont l'attribut « alt » vaut "logo" exactement.
- ~= cherche des occurrences d'un terme dans la valeur de l'attribut et est vrai s'il en trouve. Ainsi `img[alt~="logo"]` sélectionne toutes les images pour lesquelles l'attribut « alt » contient "logo", donc `<img alt="logo">` est sélectionné et `<img alt="le logo de ma société">` l'est aussi.
- |= extrait le premier mot d'un liste séparée par des tirets. Si cet opérateur est très spécifique, c'est parce qu'il répond à un besoin courant du Web. En effet, les codes de représentation des langues sont des identifiants de deux lettres, suivis éventuellement d'une autre séquence de lettres séparée par un tiret qui définit le sous-langage (par exemple `fr-FR` pour le français de France ou `fr-CH` pour le français parlé en suisse). Pour testé l'attribut « lang » d'un élément cet opérateur permet d'extraire la langue et de la comparer à une valeur, indépendamment du sous-langage (par exemple `h1[lang|="fr"] { color: blue; }`).

Il existe aussi un sélecteur générique qui sélectionne tous les éléments, il sert par exemple à sélectionner tous les enfants d'un autre élément comme dans `div > *`.

L'opérateur « , » permet d'attribuer un bloc de propriétés CSS à plusieurs sélecteurs.

Identificationd'éléments

- Possibilité de :
  - grouper des éléments par classes avec l'attribut « class »;
    Exemple : `<p class="summary">`
  - identifier (nommer) des éléments avec l'attribut « id »
    Exemple : `<img alt="logo de ma société" id="logo">`
- Sélecteurs CSS correspondants :

A. NomDeClasse
: Sélecteur de classe ↔ `A[class~="NomDeClasse"]`

A#NomDElement
: Sélecteur par identifiant ↔ `A[id="NomDeElement"]`

Les sélecteurs CSS permettent de parcourir le DOM de manière assez efficace, mais pas d'identifier des éléments particuliers. Pour cela, deux attributs peuvent être ajoutés aux éléments dans le document (X)HTML afin de grouper par catégorie ou d'identifier de manière unique certains éléments. Ces attributs sont génériques et n'apportent pas de connotation aux éléments auxquels ils sont attribués, de ce fait ils ne perturbent pas la structure logique du document (X)HTML. Par contre, en les réutilisant dans les CSS, ils permettent d'appliquer les styles à des éléments choisis.

L'attribut « class » permet de créer des catégories appelées « classes ». Il suffit de donner la même valeur de « class » à plusieurs éléments pour qu'ils soient tous accessibles à travers cette catégorie dans la feuille de style. Un élément peut être associé à plusieurs catégories, il suffit de séparer leur nom par des espaces dans le contenu du champ « class ».

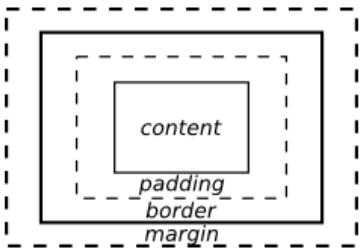
L'attribut « id » est un identifiant unique. Il ne doit apparaître qu'une seule fois dans une page Web (il y a déjà été fait référence dans cette présentation pour spécifier des URL internes à un document), ainsi normalement un sélecteur CSS par identifiant ne sélectionne qu'un seul élément du document.

Les sélecteurs de classe et d'identifiant sont respectivement « . » et « # ».

Propriétésusuellesutiles(en CSS visuelles)

- color
- background
  - -color
  - -image
  - -repeat
  - -position
- text-decoration

- margin, padding, border



- font
  - -style
  - -weight
  - -size
  - -family
- text-align
- text-indent

Exemples :

- `color : [ #fff | #ffffff | rgb(255, 255, 255) | white ] ;`
- `background : #fff url("cloud.png") no-repeat top left ;`
- `text-align : center ; text-decoration : underline ;`

`color` définit la couleur du texte. Les valeurs de couleur en CSS peuvent être exprimées sous forme d'un nombre héxadécimal précédé d'un dièse « # », à trois ou six chiffres. Une notation décimale peut être employée avec la fonction `rgb()` qui prend en paramètre la valeur de chaque couleur primaire (dans l'ordre rouge-vert-bleu) exprimée entre 0 et 255. De plus, certaines couleurs sont pré-enregistrées sous la forme de mots-clés, mais l'usage de ces mots-clés est déconseillé car l'agent utilisateur risque de ne pas en connaître certains, alors que pour les notations numériques, à partir du moment où il la comprend il peut comprendre toutes les couleurs.

`background` est une propriété qui regroupe plusieurs :

- `background-color` définit la couleur de fond.
- `background-image` contient l'URL d'une image pour le fond ou `none`.
- `background-repeat` peut prendre une valeur parmi `repeat`, `repeat-x`, `repeat-y`, , et `no-repeat` selon la façon dont le motif de fond doit être reproduit ou non pour remplir tout l'arrière plan de l'élément.
- `background-position` permet de positionner le motif de fond par rapport à la zone occupée par l'élément. Il peut être une combinaison de `top` ou `bottom` et de `left` ou `right`. Il peut aussi être exprimé de manière numérique à l'aide des unités CSS standards.

Les unités CSS comprennent notamment : `px` (pixel), `pt` (point), `em` (`1em` = taille de base) et `%` `em` et `%`sont des unités relatives à valeur héritée pour l'élément sélectionné.

`font` et ses dérivés permettent de changer la taille, le style, la graisse et la police utilisés pour le texte.

Le nombre de propriétés CSS n'est pas infini, mais s'en approche, il est inutile de vouloir toutes les apprendre. Des documents de références et des programmes spécialisés permettent de s'y retrouver assez aisément.

Associationd'une feuille de style à un document

- L'élément <style> : peu intéressant, présentation dans le document, mauvaise distinction fond/forme
- Lier le document (dans l'en-tête) à un fichier CSS séparé :
  ```
  <link rel="stylesheet" href="style.css" type="text/css" media="screen" id="MonStyle" />
  ```
  - `<link>` : méta-donnée, liaison d'un fichier externe au document
  - « rel » : nature de la relation liant le fichier externe au document
  - « href » : URL du fichier externe
  - « type » : type du fichier lié
  - « media » : média pour lesquels ce fichier a du sens
  - « id » : identifiant de l'élément <link>

Un élément <style> existe et peut s'insérer dans les en-têtes du document (X)HTML. Son utilisation n'est pas détaillée ici car elle a peu d'intérêt, en effet elle annule complètement l'intérêt d'avoir séparé le fond et la forme du document, puisqu'elle réinjecte les informations sur la forme à l'intérieur du document contenant le fond. L'approche privilégiée est donc de lier le document à une (ou plusieurs) feuille de style externe à travers les méta-données du document.

L'élément <link> sert à établir un lien entre le document et des fichiers externes identifiés par leur URL. Son attribut « rel » permet de définir la nature de la relation qui lie le document au fichier externe. Dans le cas présent, ce fichier est une feuille de style pour notre document, donc la valeur de « rel » est le mot-clé « stylesheet ». L'attribut « href » contient l'URL du fichier lié et « type » son type, ici ce fichier est au format CSS ce qui s'écrit avec la norme MIME : « text/css ». Il n'est pas opportun de lier le fichier externe au document pour tous les médias utilisés, ici la feuille de style est à destination des médias graphiques comme les écrans. Il existe d'autres médias graphiques comme les présentations, les appareils mobiles et d'autres, il est possible d'associer le fichier à plusieurs média en les listant dans l'attribut « media ». Dans l'exemple ci-dessus, la feuille de style n'est utilisée que pour le médium « screen » qui correspond à un affichage à l'écran, le support le plus courant pour visualiser un document Web.

Règles spéciales

- Règles : mots-clés « @ »

@charset
: Définition du jeu de caractères
`@charset "ISO-8859-1";`

@import
: Import de fichier CSS
`@import "style.css";` ou `@import url("style.css") print, screen;`

@media
: Spécification de règles destinées à un medium donné
`@media print, screen { h1 > span { color: #000; } }`

@page
: Propriétés de rendu des pages
`@page { size: landscape; margin: 1cm; }`

Une règle spéciale commence par un mot clé précédé de « @ » et se termine avec :

- un point-virgule « ; »;
- ou la fin du premier bloc qui la suit (« { } »).

`@charset` permet de préciser le jeu de caractères utilisé dans la feuille de style. Il ne peut être utilisé que dans les feuilles de style externes (sinon c'est le jeu de caractère du document où est défini le style qui remplace cette déclaration). En pratique, seules des feuilles de style très complexes nécessitent un telle déclaration, car les propriétés CSS usuelles et le contenu qu'elles acceptent n'utilisent que des caractères communs aux différents encodages.

`@import` permet d'insérer un fichier CSS dans une feuille de style. C'est une manière pratique d'organiser ses styles. Cette règle doit forcément être suivie de définitions CSS pour être valide (elle ne peut pas terminer un fichier CSS par exemple), et il est possible de préciser pour quels médias l'appliquer.

`@media` permet de ne spécifier des règles CSS que pour un médium précis.

`@page` sert à associer des propriétés CSS aux pages pour les terminaux concernés (comme l'impression par exemple). Comme les pages ne font pas partie de la structure du document mais uniquement de sa présentation, il n'est pas possible de les sélectionner avec les sélecteurs CSS simples, d'où l'utilité d'une règle spéciale à cet effet.

Média CSS 2

Types de médias	Groupes de médias			
	continuous/paged	visual/aural/tactile	grid/bitmap	interactive/static
speech	continuous	aural	N/A	*les deux*
braille	continuous	tactile	grid	*les deux*
spanboss	paged	tactile	grid	*les deux*
handheld	*les deux*	visual	*les deux*	*les deux*
print	paged	visual	bitmap	static
projection	paged	visual	bitmap	static
screen	continuous	visual	bitmap	*les deux*
tty	continuous	visual	grid	*les deux*
tv	*les deux*	visual, aural	bitmap	*les deux*

Le standard CSS 2 définit un certain nombre de médias qui pourra être étendu dans les standards suivants. Ces médias sont censés être représentatifs des divers terminaux d'accès au Web pour permettre d'adapter la mise en forme des contenus du Web aux terminaux utilisés.

Ces médias sont classés par groupes et à chacun de ces groupes sont associées des propriétés CSS qui ont du sens. Ainsi les règles qui n'ont pas de sens pour le terminal utilisé sont tout simplement ignorées, ce qui permet d'écrire des feuilles de style génériques pour tous les médias, ensuite seuls les médias concernés par les règles de la CSS les interpréteront et les appliqueront. L'usage des médias dans les CSS ne sert donc pas isolé du code spécifique à chaque média, mais bien à définir des mises en forme spécifiques, il n'y pas besoin de séparer les codes.

En fait, CSS va même plus loin puisque toute règle qui n'a pas de sens, y compris pour une raison de syntaxe non conforme, doit être ignorée. Ce procédé rend l'usage des CSS très tolérant aux erreurs et surtout rend les CSS d'une certaine version interprétables par un agent utilisateur qui ne comprend que la version précédente. L'agent utilisateur n'applique que les règles qu'il comprend. En pratique les agents utilisateurs n'ont qu'une implémentation des CSS (sauf IE pour des raisons historiques) qui n'est pas calée sur une version donnée, ils essaient d'anticiper la version suivante et peuvent donc implémenter seulement une partie de la spécification d'une version donnée : ce n'est pas gênant, ils n'appliquent toujours que ce qu'ils comprennent.

L'ensemble des règles et propriétés CSS défini dans le standard CSS 2 est énorme et devrait encore s'étendre dans les versions suivantes, il est donc inutile de vouloir toutes les passer en revue. L'essentiel est d'en connaître quelques-unes et de pouvoir en chercher d'autres quand l'utilité apparaît, et de garder à l'esprit que tout est dépendant du médium utilisé pour accéder au document Web.

Complémentssur les CSS

- Définition de styles alternatifs :
  ```
  <link rel="alternate stylesheet" href="style.css" type="text/css" media="visual"
  id="AutreStyle" />
  ```
- Sélecteurs avancés :
  - `a:link,a:active,a:visited,`
  - `*:hover`
  - `*:before,*:after`

Si vous avez tout retenu jusqu'ici sur les CSS, vous en savez probablement déjà trop !

Une manière de vérifer que la séparation contenu/présentation a bien été réalisée est de visualiser le document sans appliquer de feuille de style. Si l'ensemble de l'information utile structure le document, alors la mise en forme par défaut doit donner un résultat clair et compréhensible. Grâce à cette séparation il est même possible de proposer plusieurs présentations, pas uniquement en fonction du médium d'accès, mais aussi par volonté de répondre aux goûts variés des utilisateurs. Ainsi, certains agents utilisateurs permettent de choisir la feuille de style à utiliser pourvu que le document en propose plusieurs alternatives les unes aux autres. La feuille de style par défaut se déclare normalement et ses alternatives de la même manière mais en remplaçant la relation « stylesheet » par « alternate stylesheet ».

Des sélecteurs avancés étendent encore le potentiel des CSS. Les « :link », « :active » et « :visited » qui s'appliquent à l'élément <a> permettent de tenir compte de l'historique et des actions de navigation de la personne qui consulte le document Web et permettent de lui fournir des repères souvent indispendables (distinction des liens parcourus, de ceux encore inexplorés). Comme expliqué précédemment, les liens sont des ancres associées à une URL (celle vers laquelle pointe le lien). Donc le même élément ancre <a> peut être un lien (si l'attribut « href » est défini) ou non. Le sélecteur « :link » permet donc de distinguer les liens des ancres simples (comme `a[href]` mais uniquement pour les liens non-visités). « :active » et « :visited » sélectionnent respectivement le lien actif (sélectionné dans la page, utile notamment lors de la navigation au clavier) et les liens dont les cibles ont déjà été visitées.

« :hover » permet de ne sélectionner un élément que s'il est sous le curseur et apporte permet un premier pas vers l'interactivité sans utiliser d'autre langage que CSS. Enfin d'autres sélecteurs comme « :before » et « :after » permettent de fournir du contenu à destination exclusive de la présentation. Leur usage courant est d'insérer des caractères dans le flux de texte d'un document, comme les guillemets autour des citations. En effet, l'élément <q> contient déjà l'information que le texte est une citation, mais pour le mettre en forme les règles de typographies veulent que le texte soit encadré de guillemets. L'idéal est de le faire avec les CSS, et « :before » et « :after » servent notamment à cela.

Les agents utilisateurs pouvant n'implémenter qu'une partie du standard CSS, lors de l'utilisation de sélecteurs avancés, il est impératif de vérifier que la mise en forme reste acceptable même lorsque ces sélecteurs ne sont pas interprétés par l'agent utilisateur.