

Introduction : du HTML au XHTML

Vers une normalisation

Concepts et Technologies XML

Master Pro ILI

Année 2008-09



Les langages de balises : *SGML (Standard Generalized Markup Language)* date du début des années 70

Le web : création au CERN en 1989 par Tim Berners-Lee.

Objectif : mettre à disposition facilement des documents.
À l'origine, des travaux de recherche en physique qui intéressaient une communauté internationalement dispersée.

En utilisant un langage de description de documents :

HTML (*HyperText Markup Language*)

- ▶ langage de balisage hérité de SGML
- ▶ liens entre des documents situés n'importe où sur le réseau
- ▶ diffusion de documents contenant du texte, des images, du son
- ▶ la mise en forme est interprétée par le navigateur

Basé sur un protocole : **HTTP**

- ▶ un serveur web : ou serveur **HTTP-D** processus qui attend des requêtes d'un client web ;
- ▶ un client web : ou navigateur, processus qui effectue les requêtes *HTTP* auprès d'un serveur.
- ▶ exécution sur le serveurs de traitements (scripts CGI, ...)
- ▶ ou sur le client
- ▶ généralisation : les web services

Localisation d'une ressource sur Internet :

- ▶ Uniform Resource Locator (URL)

`protocole://nomHote[:port]/chemin/nomFichier[#part]}`

- ▶ Uniform Resource Name (URN)

tous les autres moyens d'identifier une ressource sans préciser son emplacement

- ▶ [file](#) accès à un fichier du disque local
- ▶ [ftp](#) accès à un serveur ftp
- ▶ [http](#) accès à un serveur http
- ▶ [https](#) accès à un serveur http avec une liaison sécurisée
- ▶ [mailto](#) envoi d'un message
- ▶ [news](#) accès à un forum Usenet
- ▶ [telnet](#) connexion vers un service telnet
- ▶ ...

- ▶ page web (.html, .htm, .php, .asp,...)
- ▶ XML (.xml, .xsl, .svg,...)
- ▶ image (.gif, .jpg, .jpeg,...)
- ▶ vidéo (.mpeg,...)
- ▶ fichier pour imprimante (.ps, .pdf...)
- ▶ audio (.mp3, .ogg, .wav, .au,...)
- ▶ fichier compressé (.gz, .z, .zip,...)
- ▶ javascript (.js,...)
- ▶ fichier texte (.text, .txt...)
- ▶ données propriétaires (Flash, ...)

Le **W3C (World Wide Web Consortium)** www.w3.org a été fondé en 1994 par Tim Berners-Lee.

Son objectif est de rédiger des recommandations pour la spécification des langages, des services, des protocoles liés au Web. Dans les années 90, guerre entre les navigateurs : ajouts de spécificités au langage HTML.

Site optimisé pour . . .

Avoir du code normalisé :

être indépendant de la plateforme du client !

L'équipe : une soixantaine de chercheurs et d'ingénieurs, pour la plupart dans l'un de ces trois centres

- ▶ le MIT (Massachusset's Institute of Technology)
- ▶ l'ERCIM (Centre Européen de Recherche en Informatique et Automatique)
- ▶ Keio University (Japon)

Les membres : plus de 358 actuellement

- ▶ des universités
- ▶ des centres de recherche
- ▶ des sociétés privées (AT&T, Google, IBM, ILOG, MicroSoft, Mozilla, Nokia, SUN, etc. . .)



Le W3C édite des recommandations, qui sont des spécifications, i.e. des normes pour tous les protocoles et technologies du Web afin d'assurer leur interopérabilité et de guider l'évolution du web. Lorsqu'une recommandation est éditée, après un long processus, c'est qu'un consensus a été trouvé entre tous les membres du W3C. Outre les recommandations, le w3c propose de la documentation, des tutoriaux, des validateurs. . .

<http://www.w3.org>

- ▶ XHTML et les CSS
- ▶ Javascript
- ▶ XML
- ▶ DTD
- ▶ XPath
- ▶ XSLT
- ▶ schémas XML
- ▶ les API de programmation XML (SAX, DOM)
- ▶ XQuery

- ▶ à l'origine, HTML décrit la structure du document
- ▶ mais dérive vers un langage d'apparence (codage de la mise-en-forme *en dur*)
- ▶ permissivité du HTML
- ▶ interprétation différente selon les navigateurs
- ▶ pas de possibilité de sorties spécialisées (impression, braille, etc)

XML : eXtensible Markup Language

ensemble de règles qui permettent la création de langages de balisage personnalisés ou l'utilisation de langages de balisage existants.

Nombreuses technologies autour de XML.

Pour l'affichage de documents : les feuilles de style [CSS Cascading Style Sheets](#)



XHTML : version de HTML qui hérite de XML

- ▶ traduction de HTML 4.0 compatible avec XML
- ▶ cadre strict du XML : meilleur comportement
- ▶ permet d'accéder aux extensions de XML.

XHTML 1.0 existe en trois versions :

- ▶ **Strict XHTML** : élimine de nombreux éléments de mise en forme du HTML. Utilisation d'une CSS pour obtenir la mise en forme voulue ;
- ▶ **Transitional XHTML** : conserve les éléments et attributs de HTML. Garantit la compatibilité avec les anciens navigateurs.
- ▶ **Frameset XHTML** : idem que strict avec la possibilité d'utiliser des cadres.



Services offerts par le W3C :

- ▶ validateur XHTML
- ▶ validateur de CSS

Différences syntaxiques :

- ▶ balises en minuscules ;
- ▶ toutes les balises doivent comporter une balise de fin `</p>`,
``
- ▶ les balises vides doivent se finir par un marqueur `
`
- ▶ toutes les valeurs d'attributs doivent être quotées
``

- ▶ Pour créer un document XHTML 1.0

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0  
Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

- ▶ Pour créer un document XHTML 1.1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">
```

- ▶ Structure : `html`, `head`, `body`
- ▶ Éléments de l'entête
 - ▶ Metadata : `meta`, `title`
 - ▶ Liaisons : `base`, `link`
 - ▶ Scripts côté client : `script`
 - ▶ Styles : `style`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xml:lang="fr" lang="fr">
  <head>
    <title>
Openweb.eu.org - Les feuilles de style en cascade
    </title>
    <meta name="keywords"
        content="css, cascading style sheets">
    <link rel="stylesheet" type="text/css"
        href="/style.css" />
  </head>
```

...



- ▶ Éléments de type blocs
 - ▶ de structure : `h1-h6`, `p`, `div`
 - ▶ d'expression : `address`, `blockquote`, `pre`
 - ▶ de présentation : `center`, `hr`
- ▶ Éléments de type en ligne
 - ▶ de structure : `em`, `q`, `strong`, `span`
 - ▶ d'expression : `abbr`, `acronym`, `cite`, `code`, `dfn`, `del`,
`ins`, `kbd`, `samp`, `var`, `img`
 - ▶ de présentation : `big`, `font`, `small`, `sub`, `sup`, `tt`, `br`,
`bdo`

`<p>A la différence des méthodes employées dans les années 90, les <acronym>CSS</acronym> permettent une stricte séparation du contenu <acronym title="HyperText Markup Language" lang="en">HTML</acronym> et des informations de mise en page. Le gain réalisé est considérable.</p>`

`<p>`
Les feuilles de style `<code class="token">print</code>` permettent une impression immédiate d'une page depuis le navigateur, et dispensent d'avoir à créer une version imprimable du document
`<code class="token">HTML</code> </p>`



- ▶ Liens : `a`
- ▶ Listes : `dl`, `dt`, `dd`, `ol`, `ul`, `li`
- ▶ Tables : `table`, `td`, `th`, `tr`, ...
- ▶ Formulaires : `form`, `input`, `select`, `option`, `textarea`, ...
- ▶ Images : `img`, `area`, `map`
- ▶ Cadres : `frame`, `noframes`, ...
- ▶ Scripts côté client : `noscript`, `script`
- ▶ Événements : `onclick`, `ondblclick`, `onmousedown`, `onmouseup`, `onmouseover`, `onmousemove`, `onmouseout`, `onkeypress`, `onkeydown`, `onkeyup`
- ▶ ...

Pour accéder à une feuille de style

- ▶ avec un fichier XML

```
<?xml-stylesheet type="text/css"  
                href="maFeuilleDeStyle.css" ?>
```

- ▶ avec un fichier XHTML

```
<link rel="stylesheet"  
      href="maFeuilleDeStyle.css" type="text/css" />
```

On peut ajouter un attribut `media` pour choisir la feuille de style selon le média d'accès

```
<link rel="stylesheet" href="maFeuilleDeStyleImpres.css"
      media="print" type="text/css"/>
```

```
<link rel="stylesheet" href="maFeuilleDeStyleEcran.css"
      media="screen" type="text/css" />
```

Valeurs possibles pour l'attribut media :

screen : par défaut ; tty ; tv ; projection ; handheld ; print ;
braille ; aural ; all

Le choix du média peut être également effectué à l'intérieur de la feuille de style :

```
@media print {  
  body { font-size: 10pt }  
}
```

```
@media screen {  
  body { font-size: 12pt }  
}
```

Le choix de la mise en forme peut être laissé à l'utilisateur par la proposition des feuilles de styles alternatives.

```
<link rel="stylesheet" title="normale" href="normaleScreen.css"
      media="screen" type="text/css"/>
<link rel="stylesheet" title="normale" href="normaleImpression.css"
      media="print" type="text/css"/>
<link rel="alternate stylesheet" title="Une css originale"
      href="originaleImpression.css" media="print" type="text/css"/>
<link rel="alternate stylesheet" title="Une css originale"
      href="originaleScreen.css" media="screen" type="text/css"/>
<link rel="alternate stylesheet" title="Une autre css"
      href="differente.css" type="text/css"/>
```

Une feuille de style CSS est

- ▶ une collection de règles
- ▶ une règle :

selecteur { propriétés }

- ▶ une propriété :

nom_propriété : valeur ;

```
h4 {  
  text-align : left ;  
  font-weight : bold ;  
  font-variant : small-caps ;  
  font-size : 100% ;  
  margin-top : 1em ;  
  margin-bottom : 0.5em ;  
}  
  
h5 {  
  text-align : left ;  
  text-decoration : underline ;  
}
```

- ▶ plusieurs règles peuvent s'appliquer sur un même élément :
algorithme de résolution de conflits
- ▶ héritage de propriétés : *Cascading* style sheets !

- ▶ pas de modification de l'ordre des éléments
- ▶ pas de génération de table des matières . . .
- ▶ pas de tests logiques
- ▶ sélecteurs limités aux noms d'éléments, attributs et contexte

`nom-element [attr = "valeur"]`

Le plus simple : le sélecteur `*`

```
* {  
    color: blue;  
}
```

`monelt` sélectionne tous les éléments de nom `monelt`

`monelt[monattr]` sélectionne tous les éléments de nom `monelt` qui possèdent un attribut `monattr`.

`monelt[monattr="mavaleur"]` sélectionne tous les éléments de nom `monelt` qui possèdent un attribut `monattr` dont la valeur a été fixée à "mavaleur".

`monelt[monattr1="mavaleur1"] [monattr2="mavaleur2"]` sélectionne tous les éléments de nom `monelt` qui vérifient tous les critères précisés.

`monelt[monattr~"mavaleur"]` sélectionne tous les éléments de nom `monelt` qui possèdent un attribut `monattr` dont la valeur contient "mavaleur".

Le plus classique : `p[class="uneClasse"]`

équivalent à `p.uneClasse`

et pour sélectionner tous les éléments d'une certaine classe :

`.uneClasse`

document XHTML

```
<h3 class="exemple">Un titre</h3>
```

```
<p class="exemple">blablabla</p>
```

et feuille de style

```
p.exemple { color : red;}
```

```
.exemple { background-color : #555;}
```



Pour une mise en forme d'un élément unique

`#moneltunique`

s'appliquera à l'élément qui possède un attribut `id` dont la valeur est "moneltunique"

Pour appliquer une même règle à des éléments différents

`elt1, elt2, elt3`

Exemple :

```
h1, h2, h3, h4, h5, h6 { font-weight : bold;}
```

On peut exploiter des informations contextuelles pour appliquer une règle :

- ▶ ascendance de l'élément
- ▶ voisins de même niveau

`td > p` : un paragraphe qui est un descendant direct d'une cellule de tableau

```
ul {indent : 3em ;}
```

```
ul > ul {indent : 6em ;}
```

```
ul > ul > ul {indent : 9em ;}
```

`table p` : un paragraphe qui est un descendant d'un tableau, quelque soit la profondeur

`li :first-child` : sélectionne le premier élément d'une liste

`p :first-line` : sélectionne la première ligne d'un paragraphe
(attention ! zone variable)

`body > p :first-child :first-letter` : pour la première lettre du premier paragraphe...

```
body > p :first-child :first-letter {  
    font-size : 300%;  
    font-color : red;  
}
```

`h1 + p` : sélectionne le premier paragraphe frère d'un titre de niveau 1

`:before` `:after` : sélectionne un point juste avant ou juste après l'élément

```
body > * :first-child :before {  
    content : "Il était une fois ";  
    font-weight : bold;  
}
```

mais aussi

`a :link`, `a :hover`, `a :active`, `a :visited`

Principe général : toutes les règles s'appliquent. En cas de conflit, les sélecteurs les plus spécifiques ont la priorité. Les conflits sont réglés pour chaque valeur de propriété.

1. les sélecteurs de type `id` ont la priorité la plus haute
2. le nombre de sélecteurs d'attributs et de pseudo-classes est important
`*[class="uneclasse"][attr="unevaleur"]` est plus spécifique que `p :first-child`
3. on privilégie les descriptions généalogiques les plus précises (hors pseudo-éléments)
4. s'il reste un conflit, c'est la dernière règle spécifiée qui l'emporte

- ▶ les éléments héritent les propriétés de leurs ancêtres
- ▶ `body` est l'endroit idéal pour placer les valeurs par défaut
- ▶ certaines propriétés restent cependant ignorées (`background-image`)

Plus de 120 propriétés dans CSS2 !

Les unités de mesure

- ▶ *les mesures absolues* : `mm`, `cm`, `in` et les mesures typographiques `pt` et `pc`
- ▶ *les mesures relatives* :
 - ▶ `em` : taille de la police courante
 - ▶ `ex` : hauteur de `x` dans la police courante
 - ▶ les pourcentages

Propriétés de texte

- ▶ *famille de police* : `font-family`. Valeur : `serif`, `sans-serif`, `monospace`, `cursive`, `fantasy` pour les classes génériques, ou noms de police (à indiquer du plus spécifique au moins spécifique)
- ▶ *taille de police* : `font-size`. Valeur : mesure ou mots-clés `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large` mais aussi `smaller` et `larger`
- ▶ *style et graisse* : `font-style`. Valeur : `normal`, `italic`, `oblique`, `inherit`; mais aussi `font-weight`. Valeur : `light`, `normal`, `bold`, `lighter`, `bolder`
- ▶ *couleur* : attributs `color` et `background-color`. Valeur prédéfinie ou hexadécimale.



Propriétés de texte : alignement et indentation

- ▶ *alignement* : `text-align`. Valeur : `left`, `right`, `center`, `justify`
- ▶ *indentation* : `text-indent` pour la première ligne d'un bloc. Peut avoir une valeur négative.

Les propriétés des blocs

- ▶ *les marges* : `margin-left`, `margin-right`, `margin-top`, `margin-bottom`. Valeur : une longueur ou un pourcentage de la largeur de l'élément conteneur ;
- ▶ *les bordures* : (à l'intérieur des marges) `border`
 - ▶ largeur `thin`, `medium`, `thick`
 - ▶ style `solid`, `dashed`, `dotted`, `double`, ...
 - ▶ couleur `blue`, `green`, `#12bc3f`, ...
- ▶ *les espacements* : pour séparer le texte de la bordure à l'intérieur `padding`
- ▶ *la largeur* : `width`. Elle correspond au contenant **plus les marges, bordures et espacements gauches et droits.**

- ▶ *flux normal* : les blocs sont positionnés dans un flux vertical. Les éléments en-ligne sont positionnés dans un flux de gauche à droite et de haut en bas.
- ▶ *sortir du flux* : `float` peut prendre les valeurs `left`, `right`, `inherit` ou `none` (valeur par défaut).

- ▶ le placement vertical n'obéit pas aux mêmes règles.
- ▶ pour éviter des débordements : `clear` empêche une cohabitation horizontale avec un bloc flottant. Valeurs possibles : `left`, `right`, `both`, `none`, `inherit` (par défaut, `none`)

- ▶ modifier la position d'un bloc : `position` peut prendre les valeurs `static`, `relative`, `absolute`, `fixed` (par défaut, `static`)
- ▶ visibilité d'un élément : `visibility` peut prendre les valeurs `visible` (par défaut) ou `hidden`, ou alors propriété `display` qui peut prendre la valeur `none`.

et d'autres...

- ▶ pour gérer les tables
- ▶ pour gérer les listes
- ▶ pour obtenir un compteur (de titre, de liste, ...)
- ▶ pour insérer du texte
- ▶ ...