Programmation WEB

(X)HTML et CSS

Programmation licence

IUT de Fontainebleau

4 octobre 2014

Langage à balises

- HTML (hyper text markup language) est un langage à balises (tag).
- Décrit les pages WEB.
- Version 4.01 depuis 1999

`http://www.w3.org/TR/html401/`

- Version 5 spécification en cours

`http://www.w3.org/TR/html51/`

  - intégration de la vidéo et son, support pour le dessin.
  - ajout d'attributs, de contrôles et de balises structurantes.
  - adaptation aux périphériques.

Balise

- Balise : mot clé entre "<" et ">" (case insensitive).
  - Généralement, une balise de début (ouvrante) et de fin (fermante).
  - Certaines balises sont seules : `<br>` (elles n'ont pas de contenus)
- Attributs : les balises peuvent contenir des attributs
  - une paire `nom="valeur"` (minuscule, souvent case sensitive)
  - Il peut y en avoir plusieurs : `id, class, onclick, ...`

Balise et attributs

```
<a href="http://www.iut-fbleau.fr"> iut </a>
```

- Balise ouvrante <a>
- Attribut href de la balise
- contenu texte de la balise
- Balise fermante

Balise

- Balise : mot clé entre "<" et ">" (case insensitive).
  - Généralement, une balise de début (ouvrante) et de fin (fermante).
  - Certaines balises sont seules : `<br>` (elles n'ont pas de contenus)
- Attributs : les balises peuvent contenir des attributs
  - une paire `nom="valeur"` (minuscule, souvent case sensitive)
  - Il peut y en avoir plusieurs : `id, class, onclick, ...`

Balise et attributs

```
<a href="http://www.iut-fbleau.fr"> iut </a>
```

- Balise ouvrante <a>
- Attribut href de la balise
- contenu texte de la balise
- Balise fermante

Balise

- Balise : mot clé entre "<" et ">" (case insensitive).
  - Généralement, une balise de début (ouvrante) et de fin (fermante).
  - Certaines balises sont seules : `<br>` (elles n'ont pas de contenus)
- Attributs : les balises peuvent contenir des attributs
  - une paire `nom="valeur"` (minuscule, souvent case sensitive)
  - Il peut y en avoir plusieurs : `id, class, onclick, ...`

Balise et attributs

```
<a href="http://www.iut-fbleau.fr"> iut </a>
```

- Balise ouvrante <a>
- Attribut `href` de la balise
- contenu texte de la balise
- Balise fermante

Balise

- Balise : mot clé entre "<" et ">" (case insensitive).
  - Généralement, une balise de début (ouvrante) et de fin (fermante).
  - Certaines balises sont seules : `<br>` (elles n'ont pas de contenus)
- Attributs : les balises peuvent contenir des attributs
  - une paire `nom="valeur"` (minuscule, souvent case sensitive)
  - Il peut y en avoir plusieurs : `id, class, onclick, ...`

Balise et attributs

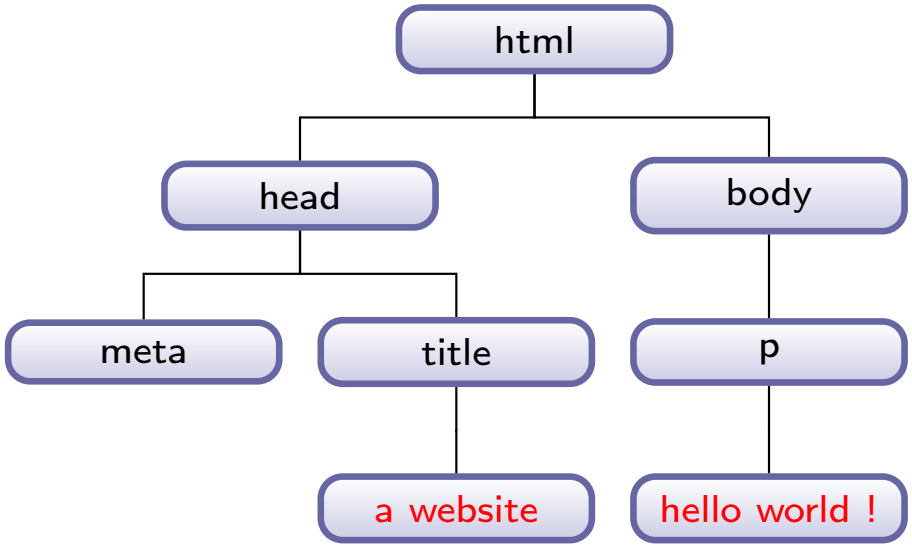
```
<a href="http://www.iut-fbleau.fr"> iut </a>
```

- Balise ouvrante <a>
- Attribut `href` de la balise
- contenu texte de la balise
- Balise fermante

Imbrication des balises

L'écriture d'un document html syntaxiquement correct obéit à des règles précises :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>a website</title>
  </head>
  <body>
    <p>hello world !</p>
  </body>
</html>
```



Représentation arborescente

XHTML

XHTML = HTML compatible XML

- insérer une déclaration

```
<!DOCTYPE html PUBLIC ...>
```

- Balises et attributs en miniscules.
- Chaque balise nécessite une balise de fin

```
<img src="..." />
<frame src="..." name="..." />
<br />
<meta name="..." content="..." />
```

- Pas d'attribut vide, valeur entre quote simple.

Structure minimale d'un document html

Structure du document html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>a website</title>
  </head>
  <body>
    <p>hello world !</p>
  </body>
</html>
```

Structure minimale d'un document html

Structure du document html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>a website</title>
  </head>
  <body>
    <p>hello world !</p>
  </body>
</html>
```

Un préambule DOCTYPE qui indique la syntaxe qui est utilisé dans le document.

Structure minimale d'un document html

Structure du document html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>a website</title>
  </head>
  <body>
    <p>hello world !</p>
  </body>
</html>
```

La balise html est la racine du document. Elle contient :

- la balise head, qui contient des métadonnées.
- la balise body, qui contient le contenu.

Structure minimale d'un document html

Structure du document html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>a website</title>
  </head>
  <body>
    <p>hello world !</p>
  </body>
</html>
```

La balise head, qui contient ici :

- la balise `meta` qui permet de rajouter des mots clés, le type du contenu, une description. Utilisée par les navigateurs et robots.
- un titre avec la balise `title`

On peut rajouter :

- Des ressources utilisées par la page avec la balise `link`.
- Des références aux fichiers javascript avec la balise `script`.

Structure minimale d'un document html

Structure du document html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>a website</title>
  </head>
  <body>
    <p>hello world !</p>
  </body>
</html>
```

La balise body contient les balises affichées dans le navigateur.

Validation de la syntaxe

http://validator.w3.org

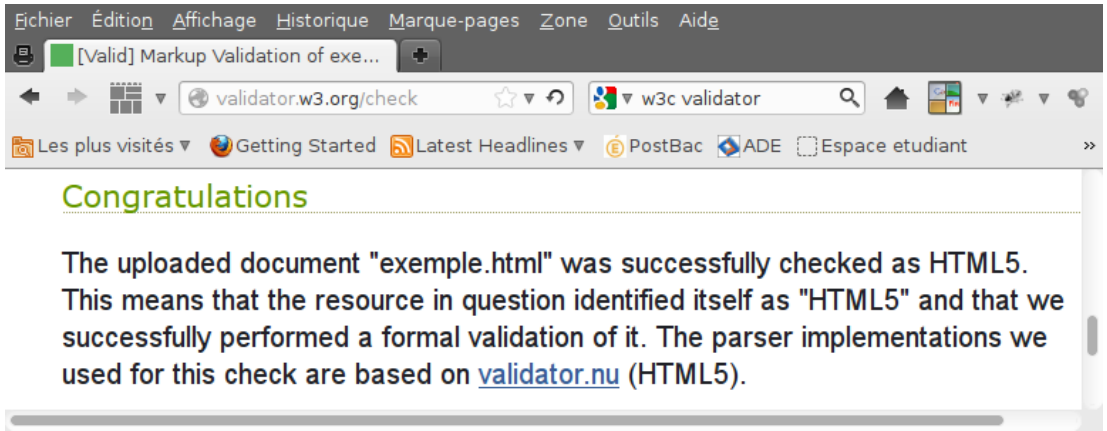
Correction de la syntaxe

source html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>a website</title>
  </head>
  <body>
    <p>hello world !</p>
  </body>
</html>
```

Validation

http://validator.w3.org



class et id

les attributs class et id permettent d'identifier un élément utilisable avec les `css` et `javascript`.

- `class` est un attribut réutilisable qui peut être utiliser sur tout élément de la page.
- le nom d'une classe est une chaîne qui ne peut pas commencer par un chiffre.
- la valeur d'un `id` ne peut être utlier qu'une seule fois sur la page.

```
<div id="header">
  <h1 class="pageTitle">ANNONCES</h1>
  <div class="logo">
```

Normalisation

- W3C : World Wide Web Consortium.

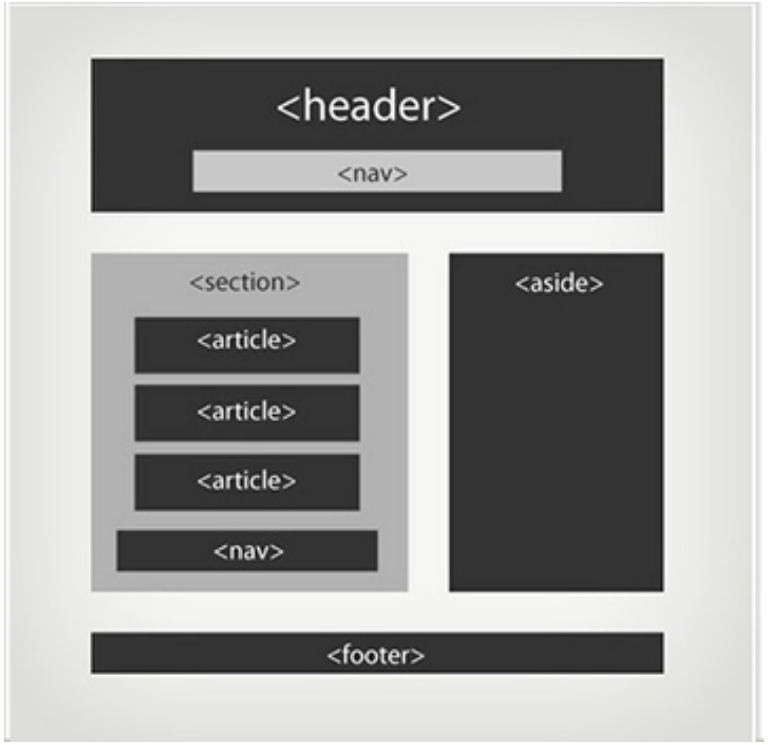
http://www.w3.org

- Consortium internationale composé d'organisations commerciales, éducatives, gourvernementales, ou individuelles.
- Pour valider une page :

http://validator.w3.org

Les éléments de section, introduits avec html5 (`section`, `article`, `nav`, `aside`, `header`, `footer`) définissent des portions du document avec une valeur sémantique particulière.

section	Section générique regroupant un même sujet, généralement avec un titre
article	Section de contenu indépendante
nav	Section possédant des liens de navigation principaux (au sein du document ou vers d'autres pages)
aside	Section dont le contenu est un complément par rapport à ce qui l'entoure, qui n'est pas forcément en lien direct avec le contenu mais qui peut apporter des informations supplémentaires.
header	Section d'introduction d'un article, d'une autre section ou du document entier (en-tête de page).
footer	Section de conclusion d'une section ou d'un article, voire du document entier (pied de page).



Textes - sémantique

Il s'agit de balises qui ont un contenu texte avec une sémantique précise :

- paragraphe
- <h1> ... <h6> : titre
- emphase
- renforcement
- <pre>affiche le contenu tel quel</pre>
- <blockquote>mise en valeur d'un paragraphe</blockquote>
- <acronym></acronym> et <abbr></abbr>.
- etc ...

Textes - mise en forme

- texte
-
 passage à la ligne
- italique
- gras
- <u>souligné</u>
- <strike>texte barré</strike>
- <u>souligné</u>
- <big>, <small>
- <sub>, <sup>
- etc

Listes

- Liste ordonnée <ol>, liste à puces
- Elément de chaque liste

```
<ul>
  <li>item 1</li>
  <li>item 2</li>
</ul>
```

- Liste de définition <dl> (définition list)
- rajouter un terme <dt> et sa définition <dd>

```
<dl>
  <dt>html</dt>
  <dd>langage utilisé pour affiché du contenu sur le web</dd>
  <dt>balise</dt>
  <dd>élément présentant certaines fonctionnalités</dd>
  <dt>attribut</dt>
  <dd>élément spécifique à une balise pouvant prendre certaines valeurs</dd>
</dl>
```

Liens

```
<a href="url ou #nom d'une ancre" target="ou">texte ou image</a>
```

- `target` représente où le lien doit être ouvert :
  - s'il manque, la même page
  - `_blank` un nouvel onglet ou nouvelle page
  - autres valeurs : `_self`, `_parent`, `_top`
- `url` est l'url du lien. il peut être :
  - absolu : `http://www.iut-fbleau.fr`
  - relatif par rapport
    - à l'adresse de base si document contient un élément `base`,
    - à l'adresse de la page en cours sinon.

Pour créer une ancre :

ancre

```
<a name="ancre1">mon ancre</a>
```

Expliquez les différences

```
<h1>Sports</h1>
<p>
  <a href="sport.html#foot">Le football </a>
  <a href="#arc">Le tir à l'arc</a>
  <a href="http://www.ff-handball.org/">handball</a>
</p>
```

Tableaux

Exemple

```
<table cellspacing="2px" cellpadding="2px;" rules="all"
  style="border:solid 1px black;">
  <caption>Titre du tableau</caption>
  <thead>
    <tr>
      <th>Titre Col 1</th>
      <th>Titre Col 2</th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td>Valeur Footer 1</td>
      <td>Valeur Footer 2</td>
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <td>Valeur 1</td>
      <td>Valeur 2</td>
    </tr>
  </tbody>
</table>
```

Formulaire

Un formulaire permet la saisie et l'envoi de données au serveur :

```
<form action="url" method="post ou get">
  <input type="submit" name="soumission" value="envoyer" />
</form>
```

- `url` : définit l'url du script à qui sont envoyées les données sur le serveur.
- `method` : les données sont envoyées soit en get (dans l'url), soit en post (dans le corps de la requête http).

Un formulaire contient un ou plusieurs éléments de type `input`, `select`, `textarea`, ... qui permettent à l'utilisateur de saisir des données.

- Image :

```
<img src="url de l'image" alt="texte alternatif">
```

- Video : balise <video>.

```
<video controls="controls" poster="exemples/exemplechatbg.jpg">
  <source src="exemples/exemplechat.webm" type="video/webm" />
  <source src="exemples/exemplechat.mp4" type="video/mp4" />
  <source src="exemples/exemplechat.ogv" type="video/ogg" />
  Votre navigateur ne supporte pas le tag <video>.
</video>
```

- Son : balise <audio>.

Structures

<!-...->
<!DOCTYPE>
<html>
<head>
<title>
<meta />
<body>

Sections

<section>
<header>
<hgroup>
<nav>
<article>
<details>
<summary>
<figure>

<figcaption>
<aside>
<footer>

Références

<base />
<link />
<style>
<script>
<noscript>

Cadres

<frameset>
<frame />
<noframes>
<iframe>

Listes

<dir>

<dd>
<dl>
<dt>

Liens

<a>
<map>
<area>

Multimédia

<img />
<video>
<track>
<audio>

<source>
<embed>
<applet>
<object>
<param />
<canvas>
<svg>

Tableaux

<table>
<caption>
<colgroup>
<col />
<thead>
<tbody>
<tfoot>
<tr>
<th>
<td>

Formulaires

<form>
<fieldset>
<legend>
<label>
<button>
<input />
<textarea>
<select>
<optgroup>
<option>
<isindex>
<menu>5
<command>5
<datalist>5
<output>5
<keygen>5

Rendus visuels

<hr />

<wbr />5
<meter>5
<progress>5

Textes - mise en forme

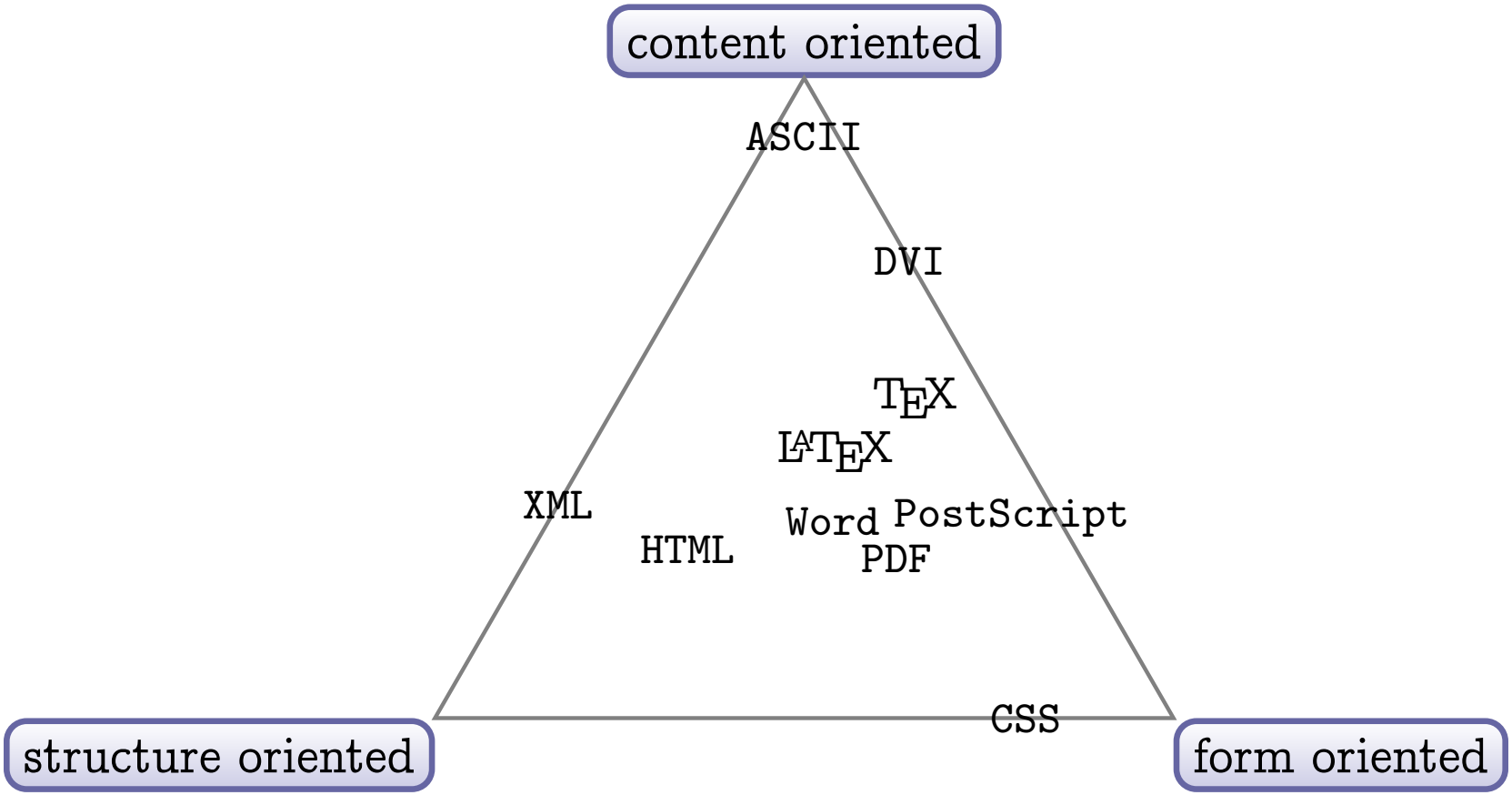
<b>
<basefont />
<bdi>5
<bdo>
<big>
<font>
<i>
<mark>5
<strike>
<sub>
<tt>
<u>
<s>
<small>

Texte - sémantique

<abbr>
<acronym>
<address>
<blockquote>
<cite>
<code>
<del>
<dfn>

<h1> ...<h6>
<ins>
<kbd>
<p>
<pre>
<q>
<rp>5
<rt>5<ruby>5
<samp>

<time>5
<var>


content oriented
ASCII
DVI
TEX
LATEX
XML
HTML
Word
PostScript
PDF
CSS
structure oriented
form oriented

Séparer le fond de la forme

- Structuration qui contient des éléments de présentation.

1

```
<br/><br/>
<b><font size="+2">
    Notre famille
</font></b>
<br/><br/>
<font size="+1">
  Michele et Jean-Luc
</font>
```

```
<div id="ma_famille">
  <h3>Notre famille</h3>
  <p>Michele et Jean-Luc</p>
</div>
```

2

- Structuration logique. (sémantique)

Principes généraux

Boîtes et flux

- Chaque élément du document html génére une zone (boîte) munie de **propriétés** d'affichage : marges, bordure, arrière-plan, largeur, etc.
- Ces boîtes peuvent être déplacées par rapport à leur position par défaut dans le flux.

Principes généraux

Boîtes et flux

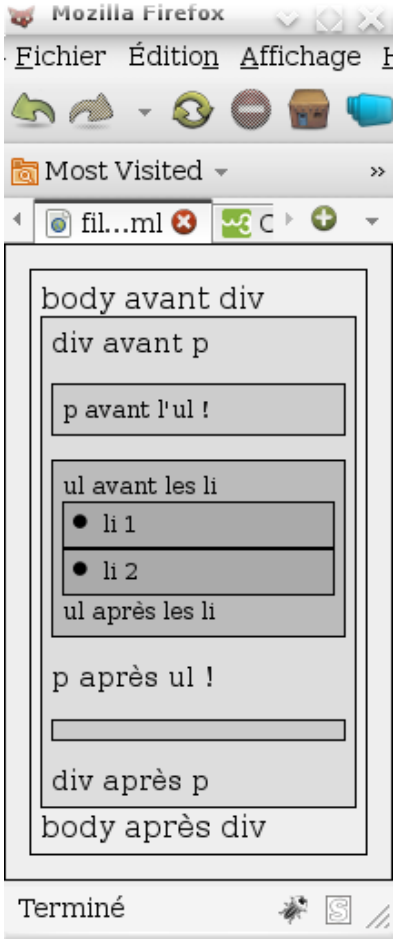
- Chaque élément du document html génére une zone (boîte) munie de **propriétés** d'affichage : marges, bordure, arrière-plan, largeur, etc.
- Ces boîtes peuvent être déplacées par rapport à leur position par défaut dans le flux.

```
<body>
  boîte body
  <div class="c1">
    boîte div
    <p>boîte p<span>un texte
        dans sa
        boîte</span>
    </p>
    <ul>
      <li>boîte li 1</li>
      <li>boîte li 2</li>
    </ul>
  </div>
</body>
```

Principes généraux

Boîtes et flux

- Chaque élément du document html génére une zone (boîte) munie de **propriétés** d'affichage : marges, bordure, arrière-plan, largeur, etc.
- Ces boîtes peuvent être déplacées par rapport à leur position par défaut dans le flux.



Principes généraux

Boîtes et flux

- Chaque élément du document html génére une zone (boîte) munie de **propriétés** d'affichage : marges, bordure, arrière-plan, largeur, etc.
- Ces boîtes peuvent être déplacées par rapport à leur position par défaut dans le flux.

Règle de sélection et propriétés : le langage CSS permet de :

- sélectionner un/des éléments,
- modifier les valeurs de certaines propiétés pour les éléments selectionnés.

```
selecteur
{propriete : valeur ;}
```

```
<body>
  <p>un paragraphe</p>
  <p class="exemple">un autre paragraphe
    selectionné par la règle
  </p>
</body>
```

Principes généraux

Boîtes et flux

- Chaque élément du document html génére une zone (boîte) munie de **propriétés** d'affichage : marges, bordure, arrière-plan, largeur, etc.
- Ces boîtes peuvent être déplacées par rapport à leur position par défaut dans le flux.

Règle de sélection et propriétés : le langage CSS permet de :

- sélectionner un/des éléments,
- modifier les valeurs de certaines propiétés pour les éléments selectionnés.

Mise en cascade et héritage

Mise en cascade des styles.

- Imbrication de plusieurs styles avec des priorités différentes.
- Les propriétés non définies pour la balise courante **sont héritées** des balises parentes (quand cela est possible).
- un style défini directement dans la page est prioritaire sur un style défini dans un fichier.

Principe des feuilles de style

- Principes des Cascading Style Sheets.
  - Distinction entre structuration et mise en forme.
  - Appliquer des règles de styles (règles de mise en forme) aux balises html.
- Pourquoi ?
  - Réutilisation et partage.
  - Homogénéisation.
  - Amélioration de l'accéssibilité.
  - Adaptation aux média.
  - Modification du contenu et/ou de la présentation indépendante.
- Les règles CSS ne transforment pas le contenu.
- Plusieurs niveaux de recommandations pour CSS : 1,2 et 3.

`http://www.w3.org/TR/CSS/`

- Le niveau 3 est en discussion. La suite concerne surtout CSS2.
- Réferences sur le web : `http://www.w3schools.com/`, `http://fr.selfhtml.org/`

Dans l'html

avec l'attribut style des balises

```
<h1 style="color:green">...</h1>
<p style="color:red">...</p>
```

avec la balise style

```
<head>
  <style type="text/css">
    p{
      font-size:16pt;
      color:blue;
    }
  </style>
</head>
```

A l'extérieur

Stockage dans un/des fichiers à part, chargé avec la page.

dans un fichier

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

Il est possible de choisir un fichier particulier suivant le médium.

règle CSS @import

```
<style type="text/css">
  @import "generales.css";
  @import url("avancees.css");
  @import url("impression.css") print, embossed;
  @import url("portable.css") handheld;
  @import url("normal.css") screen;
</style>
```

Notion de règle de sélection

Une déclaration (**Règle**) en css regroupe 3 parties :

- un selecteur,
- une propriété
- et une valeur.

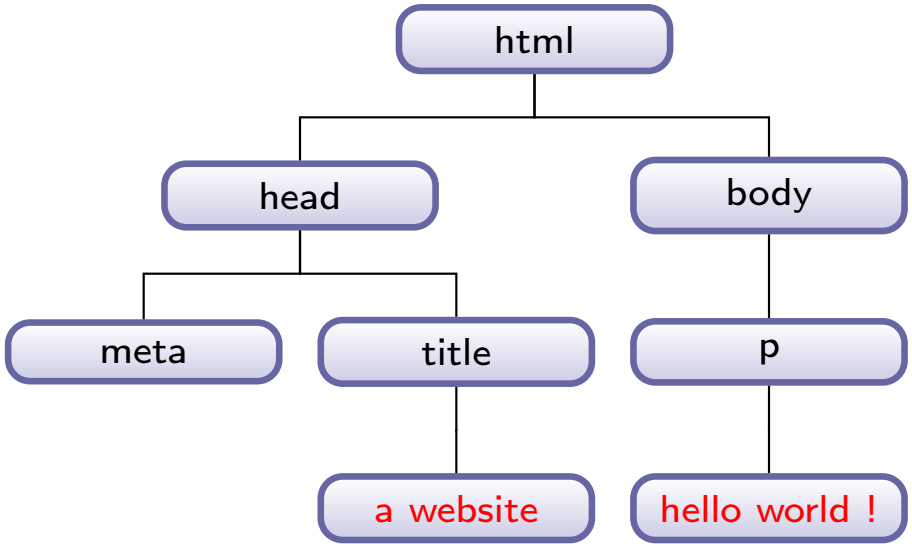
Règle

```
selecteur
{propriete : valeur ;}
```

Exemple

```
body {color:black;}
p {font-family : "helvetica";}
p {text-align :center;color:red;}
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>a website</title>
  </head>
  <body>
    <p>hello world !</p>
  </body>
</html>
```



Les selections utilisent la représentation arborescente d'un document html avec les notions de descendants, de fils, d'élèments adjacents.

Selection

- Appliquer un style à toutes les balises : **selecteur universel** * :

Exemple

```
*{
  text-align : center;
}
```

- Appliquer un style à une balise particulière : **selecteur de type** :
  - Regroupement :

```
h1,h2,h3{
  width:100%;
}
```

  - Imbrication

```
li p{
  text-align:center;
}
```

Selection : classes de style

- Classe particulière pour un élément particulier :

```
p.right{text-align:right}
p.left{text-align:left}

<p class="right">droite</p>
<p class="left">gauche</p>
```

- Classe générale pour un élément particulier.

```
p{font-size:16pt;}
```

- Classe particulière pour n'importe quel élément.

```
.blue{color:blue;}
```

-  On peut cumuler les classes

Selection id

On peut selectionner un élément unique grâce à son identifiant dans une règle.
Des mécanismes plus avancées permettent d'affiner la sélection ...

```
<style>
#vert{color:green;}
p#para1{color:red;}
</style>

<h1 id="vert">
  titre en vert
</h1>

<p id="para1">
  paragraphe en rouge
</p>
```

Selecteur de descendants

- Descendant (imbrication) :

```
li p{
  text-align:center;
}
```

```
div * p {text-align:center}
```

Expliquez ?

Selecteur d'enfants

- Descendants directs (les fils) :

```
#menu > p {color : green;}
```

S'applique à tous les p fils de l'élement #menu.

- Enfants adjacents : de la forme E1 + E2. E2 est le sujet du selecteur ; vérifié quand E1 et E2 sont adjacents dans l'arbre.

```
h1 + h2 { margin-top: -5mm }
h1.opener + h2 { margin-top: -5mm }
```

Selecteur d'attributs

Avec CSS2, on peut selectionner des éléments par leurs attributs. Les sélecteurs d'attributs peuvent trouver une correspondance de quatre façons :

- att : quand un élément a un attribut "att", quelle que soit sa valeur ;
- att=val : quand un élément a un attribut "att" dont la valeur est exactement "val" ;
- att~=val : quand un élément avec un attribut "att" qui contient val
- att|=val : quand un élément avec un attribut "att" qui commence par val .

```
h1[title] { color: blue; }
```

```
a[rel~="copyright"]
a[href="http://www.w3.org/"]
```

```
*[lang|="en"] { color : red }
```

Remarque : pour la selection de l'attribut class, pour les documents html, on utilise aussi la notation pointée.

Pseudo-éléments et pseudo-classes

CSS introduit les concepts de pseudo-éléments et de pseudo-classes qui permettent une mise en forme à partir d'informations absentes de l'arbre du document.

- Les pseudo-éléments créent des abstractions dans l'arbre en plus des éléments déjà spécifiés par le langage du document.
- Les pseudo-classes classent les éléments selon des caractéristiques autres que leur nom, attribut ou contenu, celles-ci ne pouvant pas en principe être déduites de l'arbre du document. Les pseudo-classes peuvent être **dynamiques**.

Pseudo-éléments et pseudo-classes

CSS introduit les concepts de pseudo-éléments et de pseudo-classes qui permettent une mise en forme à partir d'informations absentes de l'arbre du document.

- Les pseudo-éléments créent des abstractions dans l'arbre en plus des éléments déjà spécifiés par le langage du document.
- Les pseudo-classes classent les éléments selon des caractéristiques autres que leur nom, attribut ou contenu, celles-ci ne pouvant pas en principe être déduites de l'arbre du document. Les pseudo-classes peuvent être **dynamiques**.

Pseudo-éléments et pseudo-classes

CSS introduit les concepts de pseudo-éléments et de pseudo-classes qui permettent une mise en forme à partir d'informations absentes de l'arbre du document.

- Les pseudo-éléments créent des abstractions dans l'arbre en plus des éléments déjà spécifiés par le langage du document.
- Les pseudo-classes classent les éléments selon des caractéristiques autres que leur nom, attribut ou contenu, celles-ci ne pouvant pas en principe être déduites de l'arbre du document. Les pseudo-classes peuvent être **dynamiques**.

Pseudo-classes

- `:first-child`,`:last-child`,`:nth-child(n)` : correspond au premier,dernier, nièeme élément enfant d'un autre élément :

```
div > p:first-child {text-indent : 0;}

* > a:first-child    /* a est le premier enfant pour
                        tout element */
a:first-child        /* Idem */
```

- pseudo-classes d'ancre : `:link` et `:visited`
  - La pseudo-classe :link s'applique aux liens qui n'ont pas été visités ;
  - La pseudo-classe :visited s'applique lorsque le lien a été visité par l'utilisateur.
  - Etats exclusifs !

```
a:link { color: red }
:link  { color: red }   /* idem */
```

- pseudo-classes dynamiques : `:hover`, `:active` et `:focus`
  - `:hover` : qui est appliquée quand l'utilisateur désigne un élément (au moyen d'un appareil de pointage) sans l'activer.
  - `:active`, qui est appliquée quand l'utilisateur active un élément. Par exemple, entre le moment où l'utilisateur presse le bouton de la souris et le relâche.
  - `:focus` , qui s'applique quand un élément reçoit l'attention (celui-ci acceptant les événements du clavier ou d'autres formes d'entrées de texte).
- Ces pseudo-classes ne s'excluent pas mutuellement
- Les éléments ne sont pas nécessairement des liens.

pseudo-éléments

- :first-letter, :first-line.

```
p:first-letter {
  font-size: 200%;
  font-style: italic;
  font-weight: bold;
  float: left
}
```

- :before, :after

```
h1:before {
  content: "Chapitre " counter(chapitre) ". ";
  counter-increment: chapitre;  /* Ajoute 1 au chapitre */
  counter-reset: section;       /* Remet la section a zero */
}
```

Calcul de la priorité d'un selecteur

c'est un nombre à 4 chiffres *abcd*.

- *a* : compte 1 si la déclaration provient d'un attribut style, 0 sinon.
- *b* : compte le nombre d'attributs `id` dans le selcteur.
- *c* : compte le nombre d'autres atributs et de pseudo-classes dans le selecteur.
- *d* : compte le nombre d'éléments et de pseudo-éléments dans le selecteur.

```
*               {}  /* a=0 b=0 c=0 d=0 -> specificity = 0,0,0,0 */
li              {}  /* a=0 b=0 c=0 d=1 -> specificity = 0,0,0,1 */
li:first-line {}  /* a=0 b=0 c=0 d=2 -> specificity = 0,0,0,2 */
ul li           {}  /* a=0 b=0 c=0 d=2 -> specificity = 0,0,0,2 */
ul ol+li        {}  /* a=0 b=0 c=0 d=3 -> specificity = 0,0,0,3 */
h1 + *[rel=up]{}  /* a=0 b=0 c=1 d=1 -> specificity = 0,0,1,1 */
ul ol li.red  {}  /* a=0 b=0 c=1 d=3 -> specificity = 0,0,1,3 */
li.red.level  {}  /* a=0 b=0 c=2 d=1 -> specificity = 0,0,2,1 */
#x34y           {}  /* a=0 b=1 c=0 d=0 -> specificity = 0,1,0,0 */
style=""            /* a=1 b=0 c=0 d=0 -> specificity = 1,0,0,0 */
```

Liste des propriétes CSS

- Formatage de la police
- Alignement et contrôle de paragraphe
- Marges et espace, Espaces intérieurs
- Bordures
- Couleurs et images d'arrière plan
- Formatage de listes
- Formatage de tableaux
- Positionnement et affichage d'éléments
- etc ...

CSS3

- 3 nouveaux selecteurs d'attributs :

att^ =val : quand un élément a un attribut "att" qui commence par val.
att$=val : quand un élément a un attribut "att" qui finit par val.
att*=val : quand un élément a un attribut "att" qui contient par val.

- Adjacence indirecte : `~`.
- Pseudo classes :
  `:nth-child()`,`:nth-last-child()`,`:last-child`,`:checked`
  `:empty`, `:not`
- Pseudo-éléments : `::selection` représente la selection de l'utilisateur.

CSS3

- Ombrages sur le texte et les boîtes : `[box|text]-shadow`.
- Coins arrondis : `border-radius`
- Transparence : `rgba(...)`.
- Positionnement `inline-block`.
- Transformation géométrique : `transform`
- Media Queries.

un lien `http://www.goetter.fr/`

Media Queries

Application de feuilles de styles en fonction des périphériques clients : **Responsive Web Design**.
En CSS2 :

css2/attribut media

```
<!doctype html>
<head>
  <meta charset="utf-8">
  <title>Media Queries !</title>
  <link rel="stylesheet" media="screen" href="screen.css"
    type="text/css" />
    <link rel="stylesheet" media="print" href="print.css"
    type="text/css" />
</head>
```

- On peut intégrer dans une feuille de style la selection du media avec la règle

```
@media print {
  #menu, h1 {
    display:none;
  }
}
```

- en CSS3, possibilité d'utiliser des critéres numériques précis portant sur : la taile, la couleur, l'aspect, etc .... pour selectionner un media.

media queries

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

Le modèle des boîtes

- Tous les éléments de la page (caractérisés par des balises) sont lus dans leur ordre d'apparition dans le flux HTML et sont positionnés dans des boîtes les unes à la suite des autres sur l'écran.
- Toutes ces boîtes sont par défaut placées relativement les unes par rapport aux autres, c'est à dire à la suite. En CSS, il est tout à fait possible de sortir une boîte du flux normal pour la placer n'importe où sur la page (position :fixed ; par exemple).
- En HTML, la plupart des balises peuvent se ranger dans l'une ou l'autre de deux catégories :
  - Les balises inline : c'est le cas par exemple des liens <a>.
  - Les balises block : c'est le cas par exemple des paragraphes .

block/inline

Elements `block`	Un élément de type block va prendre tout l'espace disponible horizontalement et l'élément suivant commencera obligatoirement à la ligne (un peu comme s'il y avait un retour chariot). Les balises block sont disposées les unes en dessous des autres, quelque soient leurs dimensions
Elements `inline`	Les éléments en ligne se placent au fil du texte. Ils servent à donner du sens à certaines parties du texte. La taille d'un élément en ligne est celle de son contenu, par exemple une phrase en italique au milieu d'un paragraphe est un élément en ligne

Remarque : on peut mixer les deux avec la valeur `inline-block`.

- La largeur et la hauteur ne peuvent être fixées que pour des éléments de type block (à l'exception des éléments en ligne ne contenant pas de texte <img /> <input />...) ;
- Une boîte block peut-être positionnée sur l'axe horizontal et vertical, alors qu'une boîte inline ne peut l'être que sur l'axe horizontal ;
- Les propriétés CSS position :absolute ; et position :fixed ; ne fonctionnement pas en inline ;
- Un élément block (sauf pour les paragraphes <p> et titres <h1>...<h6>) peut contenir d'autres éléments blocks ou en ligne ;
- Un élément en ligne ne peut contenir que d'autres éléments en ligne ;
- Les éléments de type block possèdent des valeurs de marges internes (padding) et externes (margin) non nulles, contrairement aux éléments en ligne.

Un exemple

```
<body>
  boîte body
  <div class="c1">
    boîte div
    <p>boîte p<span>un texte
        dans sa
        boîte</span>
    </p>
    <ul>
      <li>boîte li 1</li>
      <li>boîte li 2</li>
    </ul>
  </div>
</body>
```

Unités et valeurs

Unités de mesure

- unités absolues :

inch (in)	un pouce (2,54 centimètres) ;
millimétre (mm)	un millimètre (0,0394 pouce)
centimétre (cm)	un centimètre (0.394 pouce)
point (pt)	unité liée aux fontes (18pt=0.25in)
pica (pc)	unité liée aux fontes (12pc=18pt)

- unités relatives :

pixel (px)	un pixel (unité physique à l'écran)
em (em)	la valeur de font-size pour le container englobant
ex (ex)	la hauteur d'un x minuscule pour la fonte utilisée pour le container englobant

- autres unités :

pourcentage (%)	pourcentage des dimensions du container englobant
auto (auto)	calcul automatique par le navigateur (margesessentiellement)

Dimensions d'une boîte

Dimensions d'une boîte

Largeur et hauteur

`height`	hauteur ;
`min-height`	hauteur minimale ;
`max-height`	hauteur maximale ;
`width`	largeur ;
`min-width`	largeur minimale ;
`max-width`	largeur maximale ;
`margin`	espace entre la bordure et le container englobant ;
`padding`	espace de remplissage entre la bordure et le texte ;

Ces informations sont utilisées par le navigateur : en cas de redimensionnement, il affichera une barre d'ascenseur horizontale ou verticale dès que la fenêtre est trop petite.

Individualisation

- `margin-left: 4px;`
- `margin: 10px 20px 30 px 40px;`

Positionnement des containers

Types de positionnement : valeur de l'attribut `position`

`static`	positionnement par défaut par rapport au container englobant ;
`relative`	positionnement relativement à la position attendue ; l'espace normal reservé dans le flux pour l'élément est maintenu.
`absolute`	positionnement absolu, relatif (!!!) au premier container englobant non static, sera déplacé en cas de déroulement vertical ou horizontal (scrolling) ;
`fixed`	positionnement immuable par rapport à la zone de visualisation, ne sera pas déplacé en cas de déroulement vertical ou horizontal ;

A noter : `float` est une propriété à part entière, indiquant que le positionnement doit se faire en dehors du flux normal d'affichage, nous reverrons cela par la suite...

Positionner un bloc en CSS

Un bloc se positionne par rapport à son conteneur. Le conteneur initial est body.

- Mode `inline` ou `block` ou `inline-block`.
- En précisant des marges
- En utilisant la propriété `float` : permet de positionner un bloc à droite ou à gauche de son conteneur, le reste du conteneur occupera l'espace restant.
- en utilisant des positions absolues/relatives
  - `position:relative` : par rapport à l'élément lui-même !
  - `position:absolute` : par rapport au coin supérieur gauche du conteneur. Utiliser bluetop,`left`,`right`,`bottom` pour les coordonnées.
- Rendre le bloc visible ou pas : `visibility`.
- Superposer des blocs : `z-index`.
- On peut tronquer (clipper) un bloc avec la propriété `clip`

Postionnement flottant des containers

float: left|right|none

- le container ayant la propriété `float` est placé lorsqu'il est rencontré dans le document HTML ;
- il est placé relativement au container englobant ;
- la suite du document se place "autour" du container flottant.

```
<style type="text/css">
  .img{float:left;}
</style>
<body>
  <img class="img" src="linux.jpg"/>
  <p>Vive Linux</p>
</body>
```