

# Plan

- 1 Introduction - Généralités
  - Navigateurs Web
  - HTML & CSS
- 2 HTML
- 3 CSS

# Programmation Web

Jean-Christophe Deneuille

[jean-christophe.deneuille@xlim.fr](mailto:jean-christophe.deneuille@xlim.fr)



FACULTÉ  
DES SCIENCES  
ET TECHNIQUES



# Plan du cours

- 1 Introduction - Généralités
- 2 HTML
- 3 CSS

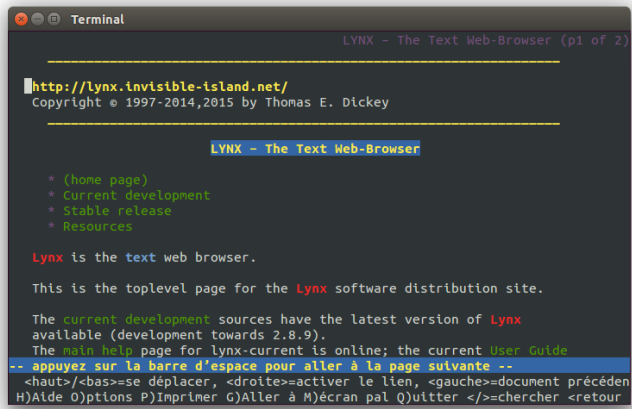
# Navigateurs

Il existe une multitude de navigateurs web.



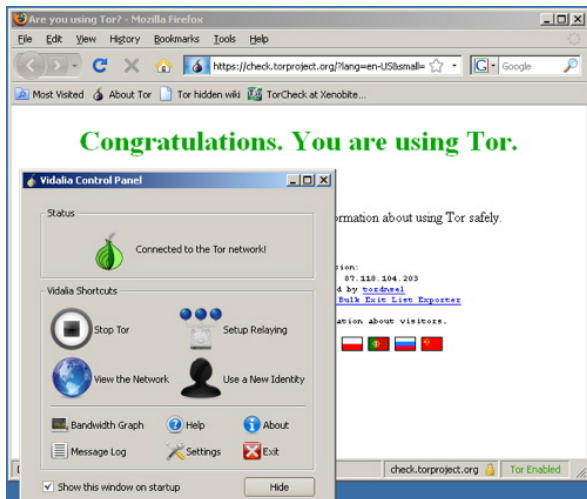
# Navigateurs

Il existe une multitude de navigateurs web, même en ligne de commande !



# Navigateurs

Certains même permettent de surfer “anonymement”.



# Navigateurs

Mais qu'est réellement un navigateur ?

# Navigateurs

Mais qu'est réellement un navigateur ?

## Navigateur

Un navigateur web est un logiciel (couche applicative) permettant de consulter le World Wide Web. Il s'agit donc a minima d'un client HTTP.

# Navigateurs

Mais qu'est réellement un navigateur ?

## Navigateur

Un navigateur web est un logiciel (couche applicative) permettant de consulter le World Wide Web. Il s'agit donc a minima d'un client HTTP.

Il s'agit donc d'un logiciel permettant d'interpréter le code source de la page, composé principalement d'**HTML**, et de **CSS**.

# Plan

- 1 Introduction - Généralités
  - Navigateurs Web
  - **HTML & CSS**
- 2 HTML
- 3 CSS

# HTML & CSS

## Hypertext Markup Language & Cascading Style Sheets

Le **HTML** et le **CSS** sont donc 2 langages qui, ensemble, permettent de créer des sites web.

**Tous les sites web** sont basés sur ces langages !

# HTML & CSS

## Hypertext Markup Language & Cascading Style Sheets

Le **HTML** et le **CSS** sont donc 2 langages qui, ensemble, permettent de créer des sites web.

**Tous les sites web** sont basés sur ces langages !

## HTML

Gérer et organiser le **contenu**.

# HTML & CSS

## Hypertext Markup Language & Cascading Style Sheets

Le **HTML** et le **CSS** sont donc 2 langages qui, ensemble, permettent de créer des sites web.

**Tous les sites web** sont basés sur ces langages !

### HTML

Gérer et organiser le **contenu**.

### CSS

Gérer l'**apparence** de la page web.

# Historique

(non exhaustif)

- 1991 : Début du HTML par Tim Berners-Lee
- 1994 : Création du W3C
- 1994-1995 : Début du CSS
- 1995 : v2.0 + draft v3.0 (tables, figures, ...)
- fin 1996 : standardisation CSS1 (relativement basique)
- début 1997 : v3.2 (standardisation)
- fin 1997 : v4.0 (support style + scripts, cadres, ...)
- 1998 : CSS2 (fonctionnalités étendues)
- 2000 : XHTML (basé sur XML → moins souple)
- 2001 : CSS2.1 (correction de la v2, prématurée)
- 2004 : WHATWG (relance de l'HTML)
- 2007 : Débuts d'HTML5, aboutissement CSS3 (modularité)
- fin 2009 : abandon d'XHTML2

(source : Wikipédia)

# Historique

(non exhaustif)

- 1991 : Début du HTML par Tim Berners-Lee
- 1994 : Création du W3C
- 1994-1995 : Début du CSS
- 1995 : v2.0 + draft v3.0 (tables, figures, ...)
- fin 1996 : standardisation CSS1 (relativement basique)
- début 1997 : v3.2 (standardisation)
- fin 1997 : v4.0 (support style + scripts, cadres, ...)
- 1998 : CSS2 (fonctionnalités étendues)
- 2000 : XHTML (basé sur XML → moins souple)
- 2001 : CSS2.1 (correction de la v2, prématurée)
- 2004 : WHATWG (relance de l'HTML)
- 2007 : Débuts d'HTML5, aboutissement CSS3 (modularité)
- fin 2009 : abandon d'XHTML2

(source : Wikipédia)

# Historique

(non exhaustif)

- 1991 : Début du HTML par Tim Berners-Lee
- 1994 : Création du W3C
- 1994-1995 : Début du CSS
- 1995 : v2.0 + draft v3.0 (tables, figures, ...)
- fin 1996 : standardisation CSS1 (relativement basique)
- début 1997 : v3.2 (standardisation)
- fin 1997 : v4.0 (support style + scripts, cadres, ...)
- 1998 : CSS2 (fonctionnalités étendues)
- 2000 : XHTML (basé sur XML → moins souple)
- 2001 : CSS2.1 (correction de la v2, prématurée)
- 2004 : WHATWG (relance de l'HTML)
- 2007 : Débuts d'HTML5, aboutissement CSS3 (modularité)
- fin 2009 : abandon d'XHTML2

(source : Wikipédia)

# Historique

(non exhaustif)

- 1991 : Début du HTML par Tim Berners-Lee
- 1994 : Création du W3C
- 1994-1995 : Début du CSS
- 1995 : v2.0 + draft v3.0 (tables, figures, ...)
- fin 1996 : standardisation CSS1 (relativement basique)
- début 1997 : v3.2 (standardisation)
- fin 1997 : v4.0 (support style + scripts, cadres, ...)
- 1998 : CSS2 (fonctionnalités étendues)
- 2000 : XHTML (basé sur XML → moins souple)
- 2001 : CSS2.1 (correction de la v2, prématurée)
- 2004 : WHATWG (relance de l'HTML)
- 2007 : Débuts d'HTML5, aboutissement CSS3 (modularité)
- fin 2009 : abandon d'XHTML2

(source : Wikipédia)

# Historique

(non exhaustif)

- 1991 : Début du HTML par Tim Berners-Lee
- 1994 : Création du W3C
- 1994-1995 : Début du CSS
- 1995 : v2.0 + draft v3.0 (tables, figures, ...)
- fin 1996 : standardisation CSS1 (relativement basique)
- début 1997 : v3.2 (standardisation)
- fin 1997 : v4.0 (support style + scripts, cadres, ...)
- 1998 : CSS2 (fonctionnalités étendues)
- 2000 : XHTML (basé sur XML → moins souple)
- 2001 : CSS2.1 (correction de la v2, prématurée)
- 2004 : WHATWG (relance de l'HTML)
- 2007 : Débuts d'HTML5, aboutissement CSS3 (modularité)
- fin 2009 : abandon d'XHTML2

(source : Wikipédia)

# Historique

(non exhaustif)

- 1991 : Début du HTML par Tim Berners-Lee
- 1994 : Création du W3C
- 1994-1995 : Début du CSS
- 1995 : v2.0 + draft v3.0 (tables, figures, ...)
- fin 1996 : standardisation CSS1 (relativement basique)
- début 1997 : v3.2 (standardisation)
- fin 1997 : v4.0 (support style + scripts, cadres, ...)
- 1998 : CSS2 (fonctionnalités étendues)
- 2000 : XHTML (basé sur XML → moins souple)
- 2001 : CSS2.1 (correction de la v2, prématurée)
- 2004 : WHATWG (relance de l'HTML)
- 2007 : Débuts d'HTML5, aboutissement CSS3 (modularité)
- fin 2009 : abandon d'XHTML2

(source : Wikipédia)

# Historique

(non exhaustif)

- 1991 : Début du HTML par Tim Berners-Lee
- 1994 : Création du W3C
- 1994-1995 : Début du CSS
- 1995 : v2.0 + draft v3.0 (tables, figures, ...)
- fin 1996 : standardisation CSS1 (relativement basique)
- début 1997 : v3.2 (standardisation)
- fin 1997 : v4.0 (support style + scripts, cadres, ...)
- 1998 : CSS2 (fonctionnalités étendues)
- 2000 : XHTML (basé sur XML → moins souple)
- 2001 : CSS2.1 (correction de la v2, prématurée)
- 2004 : WHATWG (relance de l'HTML)
- 2007 : Débuts d'HTML5, aboutissement CSS3 (modularité)
- fin 2009 : abandon d'XHTML2

(source : Wikipédia)

# Historique

(non exhaustif)

- 1991 : Début du HTML par Tim Berners-Lee
- 1994 : Création du W3C
- 1994-1995 : Début du CSS
- 1995 : v2.0 + draft v3.0 (tables, figures, ...)
- fin 1996 : standardisation CSS1 (relativement basique)
- début 1997 : v3.2 (standardisation)
- fin 1997 : v4.0 (support style + scripts, cadres, ...)
- 1998 : CSS2 (fonctionnalités étendues)
- 2000 : XHTML (basé sur XML → moins souple)
- 2001 : CSS2.1 (correction de la v2, prématurée)
- 2004 : WHATWG (relance de l'HTML)
- 2007 : Débuts d'HTML5, aboutissement CSS3 (modularité)
- fin 2009 : abandon d'XHTML2

(source : Wikipédia)

# Historique

(non exhaustif)

- 1991 : Début du HTML par Tim Berners-Lee
- 1994 : Création du W3C
- 1994-1995 : Début du CSS
- 1995 : v2.0 + draft v3.0 (tables, figures, ...)
- fin 1996 : standardisation CSS1 (relativement basique)
- début 1997 : v3.2 (standardisation)
- fin 1997 : v4.0 (support style + scripts, cadres, ...)
- 1998 : CSS2 (fonctionnalités étendues)
- 2000 : XHTML (basé sur XML → moins souple)
- 2001 : CSS2.1 (correction de la v2, prématurée)
- 2004 : WHATWG (relance de l'HTML)
- 2007 : Débuts d'HTML5, aboutissement CSS3 (modularité)
- fin 2009 : abandon d'XHTML2

(source : Wikipédia)

# Historique

(non exhaustif)

- 1991 : Début du HTML par Tim Berners-Lee
- 1994 : Création du W3C
- 1994-1995 : Début du CSS
- 1995 : v2.0 + draft v3.0 (tables, figures, ...)
- fin 1996 : standardisation CSS1 (relativement basique)
- début 1997 : v3.2 (standardisation)
- fin 1997 : v4.0 (support style + scripts, cadres, ...)
- 1998 : CSS2 (fonctionnalités étendues)
- 2000 : XHTML (basé sur XML → moins souple)
- 2001 : CSS2.1 (correction de la v2, prématurée)
- 2004 : WHATWG (relance de l'HTML)
- 2007 : Débuts d'HTML5, aboutissement CSS3 (modularité)
- fin 2009 : abandon d'XHTML2

(source : Wikipédia)

# Historique

(non exhaustif)

- 1991 : Début du HTML par Tim Berners-Lee
- 1994 : Création du W3C
- 1994-1995 : Début du CSS
- 1995 : v2.0 + draft v3.0 (tables, figures, ...)
- fin 1996 : standardisation CSS1 (relativement basique)
- début 1997 : v3.2 (standardisation)
- fin 1997 : v4.0 (support style + scripts, cadres, ...)
- 1998 : CSS2 (fonctionnalités étendues)
- 2000 : XHTML (basé sur XML → moins souple)
- 2001 : CSS2.1 (correction de la v2, prématurée)
- 2004 : WHATWG (relance de l'HTML)
- 2007 : Débuts d'HTML5, aboutissement CSS3 (modularité)
- fin 2009 : abandon d'XHTML2

(source : Wikipédia)

# Historique

(non exhaustif)

- 1991 : Début du HTML par Tim Berners-Lee
- 1994 : Création du W3C
- 1994-1995 : Début du CSS
- 1995 : v2.0 + draft v3.0 (tables, figures, ...)
- fin 1996 : standardisation CSS1 (relativement basique)
- début 1997 : v3.2 (standardisation)
- fin 1997 : v4.0 (support style + scripts, cadres, ...)
- 1998 : CSS2 (fonctionnalités étendues)
- 2000 : XHTML (basé sur XML → moins souple)
- 2001 : CSS2.1 (correction de la v2, prématurée)
- 2004 : WHATWG (relance de l'HTML)
- 2007 : Débuts d'HTML5, aboutissement CSS3 (modularité)
- fin 2009 : abandon d'XHTML2

(source : Wikipédia)

# Historique

(non exhaustif)

- 1991 : Début du HTML par Tim Berners-Lee
- 1994 : Création du W3C
- 1994-1995 : Début du CSS
- 1995 : v2.0 + draft v3.0 (tables, figures, ...)
- fin 1996 : standardisation CSS1 (relativement basique)
- début 1997 : v3.2 (standardisation)
- fin 1997 : v4.0 (support style + scripts, cadres, ...)
- 1998 : CSS2 (fonctionnalités étendues)
- 2000 : XHTML (basé sur XML → moins souple)
- 2001 : CSS2.1 (correction de la v2, prématurée)
- 2004 : WHATWG (relance de l'HTML)
- 2007 : Débuts d'HTML5, aboutissement CSS3 (modularité)
- fin 2009 : abandon d'XHTML2

(source : Wikipédia)

# Plan

## 1 Introduction - Généralités

## 2 HTML

- Présentation
- Syntaxe
- Formulaires
- Les entêtes
- Frames et XHTML

## 3 CSS

# Présentation

Il s'agit d'un langage de marquage/balilage. On spécifie donc "Ceci est mon titre, ceci est mon menu, voici le texte principal de la page, voici une image à afficher, etc..."

# Présentation

Il s'agit d'un langage de marquage/balisateur. On spécifie donc "Ceci est mon titre, ceci est mon menu, voici le texte principal de la page, voici une image à afficher, etc..."

## Editeurs de pages Web

- Mozilla Composer
- Microsoft Expression Web
- Dreamweaver ... Qualité du code généré médiocre

# Présentation

Il s'agit d'un langage de marquage/balisateur. On spécifie donc "Ceci est mon titre, ceci est mon menu, voici le texte principal de la page, voici une image à afficher, etc..."

## Editeurs de pages Web

- Mozilla Composer
- Microsoft Expression Web
- Dreamweaver ... Qualité du code généré médiocre

Je vous conseille donc d'utiliser votre éditeur de texte préféré pour la création de pages Web (surtout s'il s'agit de Sublime Text !)

# Encodage (1/2)

Un document HTML est de type text (au sens MIME). Cependant, le standard HTML 4.01 est Unicode.

# Encodage (1/2)

Un document HTML est de type text (au sens MIME). Cependant, le standard HTML 4.01 est Unicode. HTML définit donc des "entités de caractères" afin de gérer les différents accents.



## Encodage (1/2)

Un document HTML est de type text (au sens MIME).  
Cependant, le standard HTML 4.01 est Unicode.  
HTML définit donc des “entités de caractères” afin de gérer les différents accents.

	Lettre initiale		Modificateur		
&	a	A	acute	accent aigu	;
	e	E	grave	accent grave	
	i	I	circ	accent circonflexe	
	o	O	uml	accent trema	
	c	C	cedil	cédille	

## Encodage (1/2)

Un document HTML est de type text (au sens MIME).  
Cependant, le standard HTML 4.01 est Unicode.  
HTML définit donc des “entités de caractères” afin de gérer les différents accents.

	Lettre initiale		Modificateur		
&	a	A	acute	accent aigu	;
	e	E	grave	accent grave	
	i	I	circ	accent circonflexe	
	o	O	uml	accent trema	
	c	C	cedil	cédille	

Exemple : Réseaux → R&eacute;seau

## Encodage (2/2)

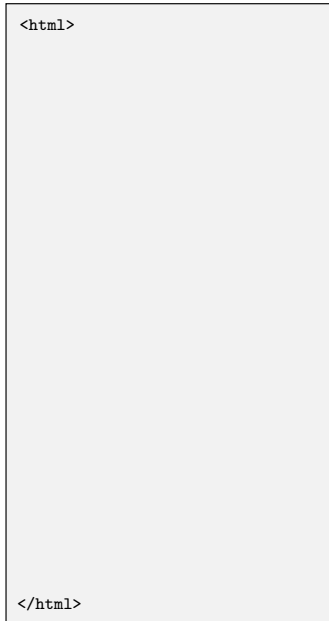
caractère	code texte	code numérique	commentaire
	&nbsp;	&#160;	espace insécable
	&shy;	&#173;	tiret de césure optionnelle (permet au navigateur de couper le mot au bon endroit si besoin de passer à la ligne)
	&lrn;	&#8206;	marque gauche-à-droite
	&rlm;	&#8207;	marque droite-à-gauche
"	&quot;	&#34;	guillemet anglais, guillemet droit (quote)
«	&laquo;	&#171;	guillemet français ouvrant
»	&raquo;	&#187;	guillemet français fermant
<	&lsaquo;	&#8249;	guillemet français simple ouvrant
>	&rsaquo;	&#8250;	guillemet français simple fermant
“	&ldquo;	&#8220;	guillemet double ouvrant, guillemet-apostrophe double culbuté
”	&rdquo;	&#8221;	guillemet double fermant, guillemet-apostrophe double
„	&bdquo;	&#8222;	guillemet double fermant bas, guillemet-virgule double inférieur
'	&apos;	&#39;	guillemet simple droit

## Plan

- 1 Introduction - Généralités
- 2 **HTML**
  - Présentation
  - **Syntaxe**
  - Formulaires
  - Les entêtes
  - Frames et XHTML
- 3 CSS

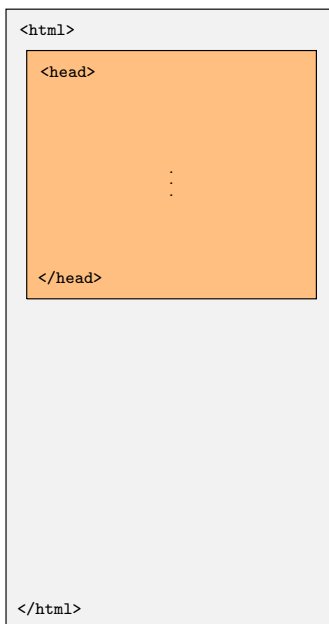
# Squelette HTML

- Balise `<html>`  
Délimite le document HTML



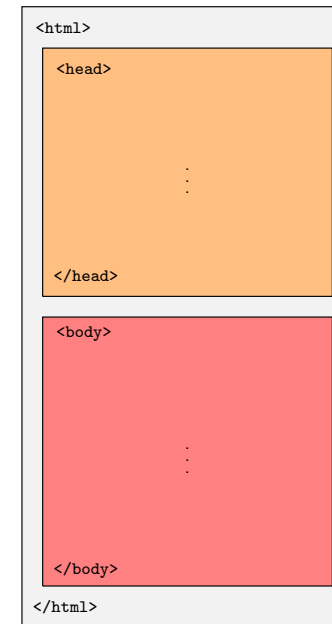
# Squelette HTML

- Balise `<html>`  
Délimite le document HTML
- Balise `<head>`  
Définit les méta-données associées au document HTML



# Squelette HTML

- Balise `<html>`  
Délimite le document HTML
- Balise `<head>`  
Définit les méta-données associées au document HTML
- Balise `<body>`  
Contient le corps de la page HTML



# Syntaxe

Il est possible d'insérer des commentaires dans une page HTML :

# Syntaxe

Il est possible d'insérer des commentaires dans une page HTML :

- Simple : `<!-- Un commentaire -->`

# Syntaxe

Il est possible d'insérer des commentaires dans une page HTML :

- Simple : `<!-- Un commentaire -->`

- Multilignes :

```
<!--
```

```
Un
```

```
commentaire
```

```
sur
```

```
plusieurs
```

```
lignes
```

```
-->
```

Navigateurs très tolérants :

Afficher ce qui est possible  
plutôt que

Produire des messages d'erreurs

# Syntaxe

Il est possible d'insérer des commentaires dans une page HTML :

- Simple : `<!-- Un commentaire -->`

- Multilignes :

```
<!--
```

```
Un
```

```
commentaire
```

```
sur
```

```
plusieurs
```

```
lignes
```

```
-->
```

## Éléments Blocs

- `<p>...</p>` : paragraphe
- `<blockquote>...</blockquote>` : citation importante
- `<address>...</address>` : information sur l'auteur
- `<div>...</div>` : générique bloc (voir CSS)
- `<pre>...</pre>` : texte préformaté

## Éléments Structure (1/2)

### • Base

- `<abbr>...</abbr>` : abréviation
- `<acronym>...</acronym>` : acronyme
- `<cite>...</cite>` : citation
- `<dfn>...</dfn>` : définition
- `<em>...</em>` : mise en valeur
- `<q>...</q>` : courte citation
- `<span>...</span>` : pour utilisation avec CSS
- `<strong>...</strong>` : mise en valeur importante
- `<br/>` : retour à la ligne

Les balises ont  
une signification  
Sémantique

Exemple :

```
<h1>Travaux de recherches</h1>
```

```
<h2>I. Introduction.</h2>
```

```
<p>
```

```
  Nous pensions faire <em>cela</em>.<br>
```

```
  Mais nous avons fait <em>ceci</em>.<br>
```

```
</p>
```

```
<h3>II. Conclusion.</h3>
```

```
<p>
```

```
  Comme disait Truc :
```

```
  <cite>On fera mieux la prochaine fois.</cite>
```

```
</p>
```

```
  Résultat obtenu :
```

## Travaux de recherches

### I. Introduction.

Nous pensions faire *cela*.

Mais nous avons fait *ceci*.

### II. Conclusion.

Comme disait Truc : *On fera mieux la prochaine fois.*

## Elements Structure (2/2)

### • Code

- `<code>...</code>` : code
- `<samp>...</samp>` : exemple de sortie
- `<kbd>...</kbd>` : texte entré par l'utilisateur
- `<var>...</var>` : variable dans un programme

### • Entêtes

- `<h1>...</h1>` : du plus important...
- `<h2>...</h2>`
- `<h3>...</h3>`
- ...
- `<h6>...</h6>` : ... au moins important

Les balises ont  
une signification  
Sémantique

Les balises permettant de positionner des attributs physiques (choix d'une fonte, couleur, etc.) pour une partie d'un document sont présentées au chapitre 11. Il est préférable de ne pas les mélanger avec les balises que nous venons de voir qui sont en rapport avec la structure logique du document. Pour les attributs physiques, il vaut mieux recourir aux feuilles de style (CSS) pour définir ceux-ci.

## Éléments Présentation

- **Usage déconseillé:** utiliser plutôt les balises structures avec un CSS

- `<b>...</b>` : texte en grad
- `<big>...</big>` : texte agrandi
- `<hr/>` : ligne horizontale
- `<i>...</i>` : texte en italique
- `<small>...</small>` : texte réduit
- `<sub>...</sub>` : texte en retrait (M<sub>lle</sub>)
- `<sup>...</sup>` : texte en puissance (x<sup>2</sup>)
- `<tt>...</tt>` : texte monospace

Les balises ont une signification de Présentation

## Listes (2/2)

- Liste de définition

```
<dl>
  <dt>item 1</dt>
  <dd>Definition 1</dd>
  <dt>item 2</dt>
  <dd>Definition 2</dd>
  <dd>More...</dd>
  <dt>item 3</dt>
  <dd>Definition 3</dd>
</dl>
```

item 1  
Definition 1  
item 2  
Definition 2  
More...  
item 3  
Definition 3

## Listes (1/2)

- Liste sans ordre

```
<ul>
  <li>item 1</li>
  <li>item 2</li>
  <li>item 3</li>
</ul>
```

- item 1
- item 2
- item 3

- Liste ordonnée

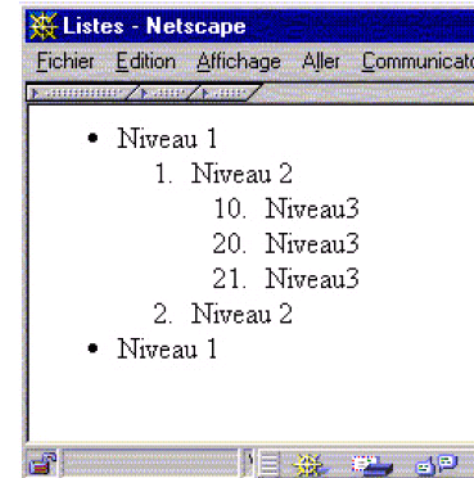
```
<ol>
  <li>item 1</li>
  <li>item 2</li>
  <li>item 3</li>
</ol>
```

- 1.item 1
- 2.item 2
- 3.item 3

## Les Listes

- Liste sans ordre <LI>, Liste Ordonnée <OL>

```
<UL>
  <LI>Niveau 1
  <OL>
    <LI>Niveau 2
    <OL>
      <LI VALUE="10">Niveau3
      <LI VALUE="20">Niveau3
    </OL>
  </OL>
  <LI>Niveau 2
</OL>
<LI>Niveau 1
</UL>
```



# Images (1/2)

- **Syntaxe:** `<img/>`
- **Attributs:**
  - `src` : adresse de l'image
  - `alt` : texte alternatif
  - `width` et `height` : dimension de l'image
- **Formats possibles:** GIF, JPEG, PNG...

```

```

# Hyperlien

Un lien hypertexte, dont l'affichage peut être un texte ou une image, est une référence à une autre ressource Web. En cliquant sur le lien, l'internaute est conduit à cette autre ressource. Les liens hypertextes constituent donc la base du concept de navigation sur le Web.

Exemple :

```
<a href="http://www.msi.unilim.fr/">
    Vers le site du LMSI
</a>
```

L'URI indiquée avec href peut aussi être relative à la page contenant le lien. Par exemple, depuis la page stockée à l'URL : `http://www.msi.unilim.fr/index.html`, le lien :  
`< a href="themes.html">`  
réfère la ressource associée à l'URL : `http://www.msi.unilim.fr/themes.html`

## Inclusion d'Images

### ■ Élément IMG

```
<IMG SRC="/icon/logo.gif" ALT="My Company Logo">
```

- image incluse dans le document
- Attribut optionnel
  - positionnement par rapport au texte
    - » ALIGN = TOP, MIDDLE, BOTTOM.
  - Forcer la taille de l'image
    - » width=180 height=60
    - » évite les reformatages intempestifs au chargement du document
- Remarque
  - Type d'image généralement supporté  
GIF, JPEG, GIF Animé, XBM
  - Plug-In pour les autres  
signifie le chargement du PlugIn et son installation (Fat-Browser)

# Hyperlien

La ressource désignée par href n'est pas nécessairement une ressource HTTP (puisque'il s'agit d'une URI).

Exemples :

```
<a href="ftp://ftp.lip6.fr/pub/python/2.4/Python-2.4.tgz">
    T&eacute;l&eacute;chargez la demi&egrave;re version
    de Python.
</a>

<a href="mailto:hegel@berlin-uni.de">
    Ecrivez moi pour obtenir des renseignements
    sur le Syst&egrave;me du Savoir Absolu.
</a>
```

# Liens locaux

Exemple :

```
<a name="intro">  
  <h1>Introduction</h1>  
</a>
```

Exemple :

```
<a href="#intro">Lien vers l'introduction</a>
```

Exemple :

```
<a href="cours.htm#intro">Lien vers l'introduction du cours</a>
```

L'attribut name peut être remplacé par l'attribut id. Cet attribut id peut de plus être utilisé pour définir une ancre avec d'autres balises. Ainsi :

```
<h2 id="section2">Section 2</h2>
```

crée une ancre dans le document, ancre qui peut ensuite être référencée par :

```
<a href="#section2">Retour &agrave; la section 2 du document</a>
```

## Tables (1/3)

```
<table border="1">  
  <tr>  
    <td>Row 1, cell 1</td>  
    <td>Row 1, cell 2</td>  
    <td>Row 1, cell 3</td>  
  </tr>  
  <tr>  
    <td>Row 2, cell 1</td>  
    <td>Row 2, cell 2</td>  
    <td>Row 2, cell 3</td>  
  </tr>  
</table>
```

Row 1, cell 1	Row 1, cell 2	Row 1, cell 2
Row 2, cell 1	Row 2, cell 2	Row 2, cell 3

## Tables (2/3)

```
<table border="1">  
  <tr>  
    <td>Row 1, cell 1</td>  
    <td colspan="2">Row 1, cell 2 and 3</td>  
  </tr>  
  <tr>  
    <td rowspan="2">Row 2 and 3, cell 1</td>  
    <td>Row 2, cell 2</td>  
    <td>Row 2, cell 3</td>  
  </tr>  
  <tr>  
    <td>Row 3, cell 2</td>  
    <td>Row 3, cell 3</td>  
  </tr>  
</table>
```

Row 1, cell 1 and 2	Row 1, cell 3	
Row 2 and 3, cell 1	Row 2, cell 2	Row 2, cell 3
	Row 3, cell 2	Row 3, cell 2

On peut procéder à une fusion de lignes ou de colonnes dans un tableau. On dispose pour cela des attributs rowspan et colspan pour les balises <td> et <th>.

```
<table border="1">  
  <tr><td rowspan="3">La</td> <td>Chartreuse</td> </tr>  
  <tr> <td>de</td> </tr>  
  <tr> <td>Parme</td> </tr>  
</table>
```

Affichage obtenu :

Chartreuse
La de
Parme

```
<table border="1">  
  <tr><td>Mort</td> <td>&agrave;</td> </tr>  
  <tr><td colspan="2">Cr&eacute;dit</td> </tr>  
</table>
```

Affichage obtenu :

Mort à
Crédit

# Plan

Exemple :

```
<table border="1">
  <caption>Années de naissance</caption>
  <tr> <th>Nom</th> <th>Année</th> </tr>
  <tr> <td>Dupont</td> <td>1964</td> </tr>
  <tr> <td>Durand</td> <td>1960</td> </tr>
</table>
```

Résultat obtenu :

Années de naissance	
Nom	Année
Dupont	1964
Durand	1960

Exemple :

```
<table>
  <tr>
    <td>
      
    </td>
    <td>
      
    </td>
  </tr>
</table>
```

Résultat obtenu :



## 1 Introduction - Généralités

## 2 HTML

- Présentation
- Syntaxe
- **Formulaires**
- Les entêtes
- Frames et XHTML

## 3 CSS

# FORMULAIRE

Un formulaire est délimité dans le code HTML par les balises : `<form>` et `</form>`. La balise `<form>` dispose des attributs suivants :

- **name** : pour une gestion du formulaire avec JavaScript
- **action** : indique l'URL ou l'adresse e-mail où sont envoyées les données
- **method** : choix entre les méthodes GET ou POST
- **enctype** : type MIME indiquant l'encodage des données du formulaire

Exemples :

```
<form name="donneesClient"
      action="cgi/verifClient.pl"
      method="GET">
  <!-- Contenu du formulaire -->
</form>
```

```
<form name="saisieDonnees"
      action="mailto:machin@domaine.org"
      method="POST"
      enctype="text/plain">
  <!-- Contenu du formulaire -->
</form>
```



# Les éléments d'un formulaire

## ■ FORM

- action, method (GET|POST), enctype, événements, ..

## ■ INPUT TYPE = (TEXT | PASSWORD | CHECKBOX | RADIO | SUBMIT | RESET | FILE | HIDDEN | IMAGE | BUTTON)

- Événements : *onfocus*, *onblur*, *onselect*, *onchange*, *accept*

## ■ TEXTAREA

## ■ BUTTON

## ■ SELECT, OPTGROUP, OPTION

## ■ FIELDSET *cadre de regroupement*

## ■ ISINDEX *obsolète*

Exemple :

```
<input type="radio"
  name="qualite" value="Mme" checked="checked">
  Madame<br>
<input type="radio"
  name="qualite" value="Mlle">
  Mademoiselle<br>
<input type="radio"
  name="qualite" value="M.">
  Monsieur<br>
```

Affichage obtenu :

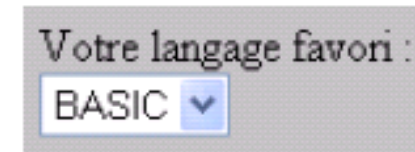


Madame  
 Mademoiselle  
 Monsieur

Exemple :

```
Votre langage favori :<br>
<select name="langage">
  <option value="BASIC">BASIC</option>
  <option value="Forth">Forth</option>
  <option value="Lisp">Lisp</option>
</select>
```

Affichage obtenu :





Votre langage favori :  
BASIC ▾

## Formulaires: exemple

- Entrée d'informations dans un formulaire

– les informations sont envoyés au serveur puis traités par un script



kartoch    
\*\*\*\*\*  
 Connexion automatique  
Connexion

```
<form method="post" action="/login.html" id="formulaire">
  <input type="hidden" name="url" value="http://linuxfr.org/pub/" />
  <input type="text" size="8" name="login" value="kartoch"/> <br />
  <input type="password" size="8" name="passwd" value="passwo"/>
  <br />
  <input type="checkbox" name="isauto" value="1" /> Connection Auto.
  <br />
  <input type="submit" name="Envoyer" value="Connexion"/>
</form>
```

# Les Formulaires HTML

## ■ Entrée d'informations dans un formulaire

- Evitez de rentrer votre numéro de carte de crédit !!!!!
- Le contenu du formulaire est envoyé à un serveur puis traité par un script du serveur

```
<HTML><HEAD><TITLE>Formulaire</TITLE></HEAD>
<BODY><H1 align=center>Formulaire</H1><hr>
<FORM ENCTYPE="multipart/form-data" METHOD="POST" ACTION="/servlet/formprocess">
Ligne de Texte : <INPUT TYPE=text VALUE=" A Remplir" NAME=lig>
Zone de Texte : <TEXTAREA NAME=zon COLS=40 ROWS=5>
A Remplir
</TEXTAREA>
<INPUT TYPE=checkbox VALUE="Opt1" NAME=opt> Option 1
<INPUT TYPE=checkbox VALUE="Opt2" NAME=opt> Option 2<BR>
<INPUT Type=submit Value="Envoi">
<INPUT Type=reset Value="Reinit">
</FORM>
</BODY></HTML>
```

# Plan

- 1 Introduction - Généralités
- 2 HTML
  - Présentation
  - Syntaxe
  - Formulaires
  - Les entêtes
  - Frames et XHTML
- 3 CSS

## Les Formulaires HTML

### ■ Assure l'interactivité avec l'utilisateur

- Entrée d'information dans un formulaire
  - Evitez de rentrer votre numéro de carte de crédit !!!!!
- Le contenu du formulaire est envoyé à un serveur puis traité par un programme CGI du serveur

```
<FORM action="http://somesite.com/prog/adduser" method="post">
<P>
First name: <INPUT type="text" name="firstname"><BR>
Last name: <INPUT type="text" name="lastname"><BR>
email: <INPUT type="text" name="email"><BR>
<INPUT type="radio" name="sex" value="Male"> Male<BR>
<INPUT type="radio" name="sex" value="Female"> Female<BR>
<INPUT type="submit" value="Send"> <INPUT type="reset">
</P>
</FORM>
```

First name:

Last name:

email:

Male

Female

## La partie <head> (2/2)

- <meta/>
  - spécifier des méta-informations sous la forme clé/valeur:
 

```
<meta name="author" content="F damien ">
<meta name="keywords" content="Flower, Kitty">
<meta name="description" content="My page
about Flower and Kitty (don't laugh)">
```
  - spécifier des entêtes HTTP:
 

```
<meta http-equiv="expires" content="Tue, 20
Aug 1996 14:25:27 GMT">
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-5">
```

Une balise <meta> a la forme suivante :

```
<meta name="identifiantInformation" content="information">
```

L'information sur le document HTML est identifiée par l'attribut name ; le contenu associé à cette information est fourni par l'attribut content.

Exemple :

```
<meta name="author" content="Karl Popper">
```

La norme HTML 4.01 ne définit aucun standard pour les noms et les contenus de la balise <meta>. Plusieurs balises de méta-données peuvent être présentes dans le document.

On peut aussi spécifier la langue utilisée par l'information de l'attribut content.

Exemple :

```
<meta name="keywords" lang="fr" content="Epistemologie, Falsifiabilité">
```

```
<meta name="keywords" lang="en" content="Epistemology, Falsifiability">
```

La balise <meta> a aussi un intérêt plus « pratique ». Elle permet d'insérer un en-tête HTTP dans la réponse du serveur. Pour cela, on substitue l'attribut http-equiv à l'attribut name.

Exemple :

```
<meta http-equiv="Expires" content="Tue, 20 Aug 1996 14:25:27 GMT">
```

a pour conséquence l'ajout de l'en-tête suivant à la réponse du serveur Web :

```
Expires: Tue, 20 Aug 1996 14:25:27 GMT
```

Remarque : l'en-tête HTTP ci-dessus peut être utile à des systèmes de cache (*proxy* par exemple) pour déterminer quand ils ont besoin de redemander un document.

Autre exemple :

```
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
```

# Plan

## 1 Introduction - Généralités

## 2 HTML

- Présentation
- Syntaxe
- Formulaires
- Les entêtes
- Frames et XHTML

## 3 CSS

## Le format HTML — les cadres

Les cadres, "frame", permettent de subdiviser la fenêtre du navigateur en plusieurs cadres indépendants, chacun de ces cadres pouvant afficher un document HTML différent.

Pour utiliser les cadres, il existe deux balises :

- la balise FRAMESET qui débute la définition d'un ensemble de cadres
- la balise FRAME qui définit chaque cadre

La définition de la taille de chacun de ces cadres peut être faite de manière relative en utilisant la notation en «%» par rapport à la taille de la fenêtre du navigateur, ou bien en pixel.

### Pourquoi ne pas les utilisés

L'utilisation des cadres **détruit l'expérience de navigation** de l'utilisateur :

- si l'utilisateur arrive sur une page qui doit faire partie d'un document comportant plusieurs cadres, il lui manque aux mieux des informations, au pire **il ne peut naviguer** sur le site.
- Exemple : Le document est décomposé en deux cadres :
- un cadre est employé pour donner une liste de liens (cadre menu), un autre cadre pour l'affichage du contenu de chaque lien (cadre information).
- Problème : l'utilisateur arrive directement sur une page «contenu» normalement affichée dans le cadre information, et ne voit pas le cadre menu. La navigation est alors impossible.
- l'utilisateur **ne peut mettre de signet** sur un document utilisant des cadres de telles manières qu'il retrouve le contenu de chaque cadre tel qu'il était à la pose du signet. Il ne mémorise que l'état initial des cadres obtenu lors du premier chargement.

Les logiciels de conception de pages HTML permettent de définir une **page type** et de réutiliser cette page type pour toutes les pages d'un site. Il suffit alors de construire le menu de navigation du site dans cette partie commune, ce qui annule le besoin de cadres.

## Le format HTML — les cadres

Les cadres sont définis en terme de ligne et de colonnes. Chacune des cases définies possède une dimension qui peut être fixée.

### Exemple de définition

```
<FRAMESET ROWS="60%,*" COLS="65%,20%,*">
```

Le «\*» permet d'utiliser la place restante.

Sur l'exemple, un ensemble de 6 cadres a été défini, sous forme de deux lignes de 3 colonnes.

La balise FRAMESET doit être employée avant le corps (balise BODY) du document.

Il doit y avoir pour chaque cadre défini, une balise FRAME qui permet d'associer au cadre :

- un document HTML;
  - un nom qui permet de désigner dans un lien le cadre où doit s'afficher le document désigné par le lien.
- Par défaut, c'est le cadre où se trouve le lien.

### Exemple

```
<HTML><HEAD> ... </HEAD>
<FRAMESET ROWS="30%,*">
  <FRAME SRC="frame1.html" NAME="a_droite">
  <FRAME SRC="frame2.html" NAME="a_gauche">
</FRAMESET>
<NOFRAMES>
```

Ce texte est affich&eacute; parceque votre navigateur ne g&eagrave;re pas les lien

```
<A HREF="...">version sans frame</A>
```

```
</NOFRAMES> </HTML>
```

L'attribut SCROLLING de la balise FRAME permet d'avoir des barres de défilement ou non pour le cadre.

### Utilisation de la notion de cible pour les liens

```
<A HREF="..." TARGET="a_gauche">
```

Exemple :

```
<frameset cols="20%,80%">
  <frame name="menu" src="menu.html">
  <frame name="article" src="introduction.html">
</frameset>
```

Dans l'exemple ci-dessus, la page menu.html contiendra probablement des liens vers des articles contenus sur le serveur. Si les liens ont la forme :

```
<a href="article1.html">Article 1</a>
```

la page de l'article s'affichera dans la frame du menu (ce qui n'est certainement pas l'effet souhaité). Pour que les pages HTML correspondant aux articles apparaissent dans l'autre frame, on va utiliser l'attribut target et le nom de la frame dans laquelle on désire que l'affichage ait lieu :

```
<a href="article1.html" target="article">Article 1</a>
```

On peut aussi avoir des cadres locaux à une page HTML avec l'élément iframe.

Exemple :

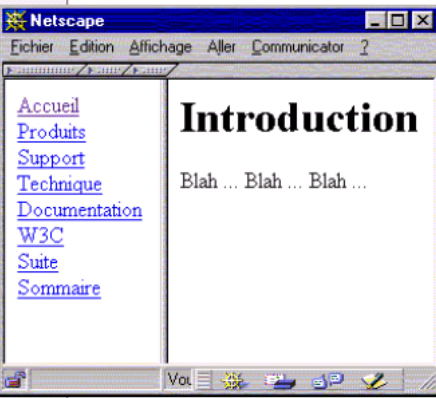
```
<table border="2">
<tr>
  <td>

  <iframe src="http://www.msi.unilim.fr/" width="250" height="250">
  </iframe>
  </td>
  <td>
  <iframe src="http://www.google.fr/" width="250" height="250">
  </iframe>
  </td>
</tr>
<tr>
  <td>
  <iframe src="http://www.alternatives87.eu.org/"
  width="250" height="250">
  </iframe>
  </td>
  <td>
  <iframe src="http://www.python.org/" width="250" height="250">
  </iframe>
  </td>
</tr>
</table>
```

## Les cibles dans les Frames HTML

```
indexframe.htm
<FRAMESET COLS="100, *">
<FRAME SRC="som1.htm" NAME="fr1">
<FRAME SRC="intro.htm" NAME="fr2">
</FRAMESET>
```

```
som1.htm
<HTML><HEAD><BASE TARGET="fr2"></HEAD>
<BODY>
<A HREF="intro.htm">
  Accueil</A><BR>
<A HREF="produit.htm" TARGET="fr2">
  Produits</A><BR>
<A HREF="sommaire.htm" TARGET="_top">
  Support Technique</A><BR>
<A HREF="http://www.w3c.org/TR" TARGET="_self">
  Documentation W3C</A><BR>
<A HREF="som2.htm" TARGET="fr1">
  Suite Sommaire</A><BR>
</BODY></HTML>
```



Résultat obtenu :



# XHTML

- Frames:
  - « *In general, frames do nothing but **add complexity** and **subtract usability**.* »
  - `frame`, `frameset`, `iframe`, `noframe`

## 12. XHTML.

XHTML (*eXtensible HyperText Markup Language*) est l'évolution du HTML préconisée par le W3C. De façon très simplifiée, on peut voir XHTML comme une reformulation en XML (*eXtensible Markup Language*) de HTML, XML étant une sorte de SGML allégé. Les buts principaux de XHTML sont :

- de séparer plus clairement que ne le fait HTML contenu et forme d'un document,
- d'utiliser les outils standards pour éditer et travailler sur des documents XML,
- de permettre un traitement automatisé plus simple des documents, notamment avec l'aide des outils XML (parsers, etc.),
- de rationaliser et faciliter les extensions du langage.

Il y a aujourd'hui deux normes concernant XHTML : XHTML 1.0 et XHTML 1.1.

XHTML 1.0 regroupe en fait trois DTDs qui correspondent à celles de HTML : Strict, Transitional et Frameset. XHTML 1.0 permet aux webmasters de migrer progressivement leurs sites en HTML vers XHTML.

XHTML 1.1 est conçu comme un ensemble de modules. XHTML Basic est un ensemble minimal de modules qui permettent un affichage même sur des appareils limités en capacités matérielles (comme les smartphones ou certains PDA). On peut ensuite adjoindre à XHTML Basic des modules étendant ses fonctionnalités ou en apportant de nouvelles. On trouve ainsi un module de script (pour le JavaScript), un module de style (pour les CSS), etc.

- Reprise de HTML avec la grammaire XML (plus stricte) mais considéré comme une branche différente: le développement de HTML n'est pas arrêté
  - XHTML 1:
    - reformulation XML de HTML 4
  - XHTML 1.1:
    - introduction de la modularité
    - disparition des 3 versions (strict, transitional et frameset)
  - XHTML 2 (en préparation):
    - Support Xform
    - XML event pour la gestion des événements
    - Nouveaux éléments, liste de navigation...



## Différences HTML 4.01 / XHTML 1

- Grammaire plus stricte
  - Les documents doivent respecter la hiérarchie

```
<p>here is an emphasized <em>paragraph</em>.</p>
```
  - Les éléments non vides doivent posséder une balise de fin

```
<p>here is a paragraph.</p><p>here is another paragraph.</p>
```
  - Les éléments vides (sans balise de fin) doivent se fermer avec `</>`

```
<dl compact="compact"/>
```
  - La valeur d'un attribut doit être présente et entre guillemets
  - L'attribut `name` est remplacé par `id`

## XML (2)

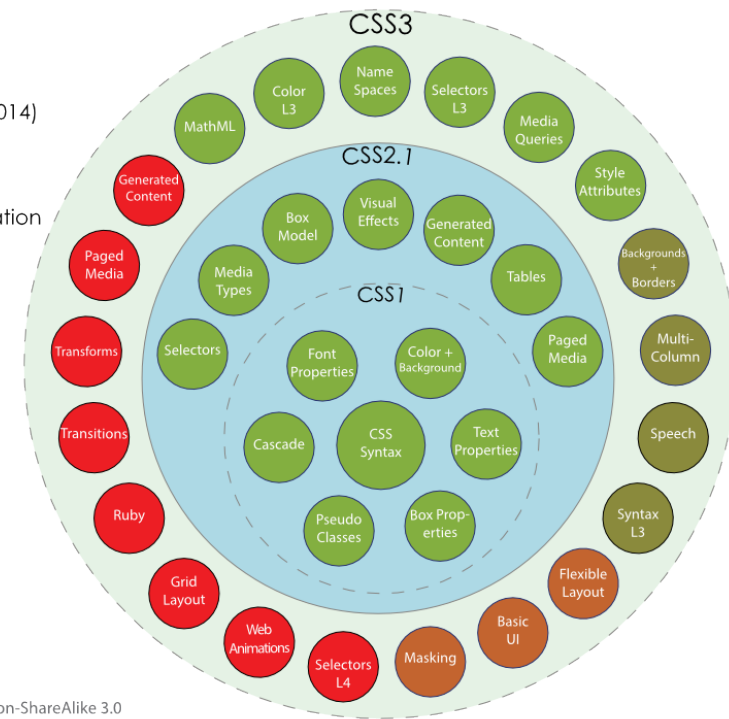
- Exemple de « technologies » web basés sur XML :
  - SVG (*Scalable Vector Graphic*)
  - XUL (*XML User Interface Language*)
  - XBL (*XML Binding Language*)
  - RDF (*Ressource Description Framework*)
  - RSS (*Really Simple Syndication*)
  - Xpath (*XML Path Language*)
  - XSLT (*Extensible Stylesheet Language Transformations*)
  - XML Web Services: SOAP, XML-RPC
- Ce sont « juste » des schémas XML garantissant l'interopérabilité



# CSS3

Taxonomy & Status (October 2014)

- W3C Recommendation
- Candidate Recommendation
- Last Call
- Working Draft
- Obsolete or inactive



By Sergey Mavrody 2011-14 | CC Attribution-ShareAlike 3.0

Introduction - Généralités  
○○

HTML  
○○○○○

CSS  
●○○○○

## Plan

1 Introduction - Généralités

2 HTML

3 CSS

- Bon croquis > long discours
- Exemple introductif
- Classes et Identificateurs
- Div et Span
- Attributs et Valeurs

Introduction - Généralités  
○○

HTML  
○○○○○

CSS  
●○○○○

## Plan

1 Introduction - Généralités

2 HTML

3 CSS

- Bon croquis > long discours
- Exemple introductif
- Classes et Identificateurs
- Div et Span
- Attributs et Valeurs

## Notre exemple

- Supposons que nous désirions qu'un titre de taille 1 ait une couleur bleue.
- Il y a trois façons d'obtenir cet effet avec le CSS.

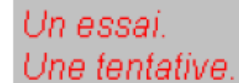
## Propriétés

Les noms des propriétés et les valeurs ne sont pas sensibles à la casse. On sépare les différentes propriétés d'un style par un point-virgule.

Exemple :

```
<p style="color: red; font-family: Arial; font-style: italic">  
    Un essai.<br />  
    Une tentative.<br />  
</p>
```

Affichage obtenu :



Un essai.  
Une tentative.

## Solution 1 : Utilisation de l'attribut de style

On spécifie ici directement le style qui ne sera valable que pour l'élément h1 courant

```
<h1 style="color: blue">  
    Introduction.  
</h1>
```

Résultat obtenu :



**Introduction.**

Un style est constitué de différentes propriétés (ici, la propriété « color ») auxquelles on affecte des valeurs (ici, le nom de couleur « blue »).

## Solution 2 : Feuille de style intégrée à la page HTML

On va définir un style par défaut pour tous les éléments h1 du document. On pourra toutefois utiliser un autre style en recourant à l'attribut style pour un élément h1 particulier (d'où le qualificatif de feuilles de style **en cascade**).

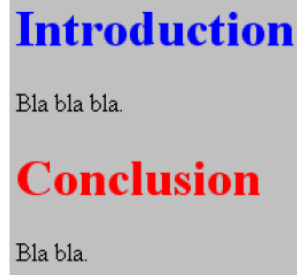
Pour insérer une feuille de style dans une page HTML, on ajoute à l'intérieur de l'élément head la structure suivante :

```
<style type="text/css">  
    <!-- définition du style -->  
</style>
```

## Solution 2

```
<!DOCTYPE ...>
<html>
  <head>
    <title>Un titre significatif</title>
    <style type="text/css">
      h1 { color: blue }
    </style>
  </head>
  <body>
    <h1>Introduction</h1>
    <p>Bla bla bla.</p>
    <h1 style="color: red">Conclusion</h1>
    <p>Bla bla.</p>
  </body>
</html>
```

Affichage obtenu :



## Remarques

- « h1 { color : blue } » est appelée une règle. « h1 » est le sélecteur de la règle, « color : blue » la déclaration de la règle.
- Il est possible de regrouper des balises pour leur donner un même style. On les sépare alors par des virgules. Exemple :  
h1, h2, h3 { font-style : italic }

## Solution 3 : Les feuilles de style externes

Il s'agit de la solution à privilégier puisqu'elle permet :

- de ne pas mélanger du contenu HTML avec la définition d'un style,
- de pouvoir réutiliser simplement un style pour différentes pages.

Pour employer cette méthode, on ajoute une balise <link> dans l'élément head de la page HTML.

Exemple :

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

L'attribut rel indique le type de lien (ici : une feuille de style).

L'attribut type sert à spécifier le type (MIME) de la feuille de style.

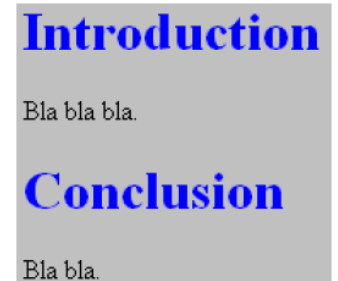
L'attribut href a pour valeur une URI référençant la feuille de style.

Remarque : Le fichier monStyle.css ne doit pas contenir de code HTML.

## Solution 3

```
<!DOCTYPE ...>
<html>
  <head>
    <title>Un titre significatif</title>
    <link rel="stylesheet" type="text/css" href="monStyle.css">
  </head>
  <body>
    <h1>Introduction</h1>
    <p>Bla bla bla.</p>
    <h1>Conclusion</h1>
    <p>Bla bla.</p>
  </body>
</html>
```

Résultat obtenu :



On a le fichier monStyle.css suivant :

```
/* Commentaire */
h1 { color: blue }
```



# Remarques

- Il est possible d'avoir plusieurs balises `<link>` pour un même document HTML. Ceci permet de combiner des définitions de style.
- Les commentaires CSS ont la même forme que les commentaires du C.

# Classes

Nous avons vu précédemment que l'on peut associer un style (constitué d'une ou plusieurs propriétés) à un élément.

Exemple :

```
p { font-size : 12pt; color : white }
```

Ce style s'appliquera alors par défaut à tous les paragraphes du document.



Pour distinguer différents types de paragraphes et leur assigner différents styles, on dispose de la notion de classe. On commence par définir dans la feuille de style différentes classes pour une balise. On positionne ensuite l'attribut `class` dans une balise du document HTML pour lui associer le style de la classe indiquée. Les balises n'ayant pas d'attribut `class` auront le style par défaut déterminé par le navigateur.

Introduction - Généralités  
oo

HTML  
ooooo

CSS  
oo●oo

## Plan

1 Introduction - Généralités

2 HTML

3 CSS

- Bon croquis > long discours
- Exemple introductif
- Classes et Identificateurs
- Div et Span
- Attributs et Valeurs

## Illustration classe 1/2

```
<!DOCTYPE ...>
<html>
  <head>
    <title>Un titre significatif</title>
    <style type="text/css">
      p.important { font-size: 1cm; font-weight: bold;
                    color: red }
      p.futile { font-size: 5mm; color: blue }
    </style>
  </head>
  <body>
    <p class="important">Lisez Login:</p>
    <p class="futile">Bla bla.</p>
    <p>Normal</p>
  </body>
</html>
```

Résultat obtenu :

**Lisez Login:**  
Bla bla.  
Normal

# Illustration classe 2/2

★ Si l'on souhaite affecter un style à tous les éléments d'une classe, on dispose de la notation :

```
.maClasse { color : green }
```

Résultat obtenu :

```
<!DOCTYPE ...>
<html>
  <head>
    <title>Un titre expressif</title>
    <style type="text/css">
      .important { font-size: 0.6cm; font-weight: bold;
                  color: red }
    </style>
  </head>
  <body>
    <h1 class="important">Un message important :</h1>
    <p class="important"><em>Lisez Linux Magazine</em></p>
  </body>
</html>
```

**Un message important :**  
*Lisez Linux Magazine*

## Identificateur

★ Pour limiter un effet de style à une seule balise de la page HTML, on a recours à un identifiant.

```
<!DOCTYPE ...>
<html>
  <head>
    <title>Un titre significatif</title>
    <style type="text/css">
      #beau { text-align: right; color: green }
    </style>
  </head>
  <body>
    <p id="beau">Lisez Linux Magazine</p>
    <p>Normal</p>
    <!--
      <p id="beau">Bla bla</p> → Interdit puisque un
                               identifiant doit être unique.
      <h2 id="beau">1</p>     → Interdit pour la même
                               raison.
    -->
  </body>
</html>
```

Affichage obtenu :

Lisez Linux Magazine  
Normal

Remarque : L'interdiction qui est mentionnée en commentaire est souvent ignorée par les navigateurs. Elle ne l'est pas par le validateur du W3C.

## Plan

1 Introduction - Généralités

2 HTML

3 CSS

- Bon croquis > long discours
- Exemple introductif
- Classes et Identificateurs
- Div et Span
- Attributs et Valeurs

## L'élément Div

La balise <div> sert à faire un regroupement dans un document HTML.

```
<!DOCTYPE ...>
<html>
  <head>
    <title>Th&egrave;se</title>
    <style type="text/css">
      div.resume { text-align : justify;
                  color: red }
    </style>
  </head>
  <body>
    <div class="resume">
      <h1>R&eacute;sum&eacute;</h1>
      <p>
        Ces travaux ont pour objectif essentiel de faire ceci, cela, et encore cela, choses fondamentales pour remplir ce magnifique rapport que vous lisez en moment.
      </p>
      <p>
        Bla bla bla.
      </p>
      <em>Mots-cl&eacute;s</em> : Bla.
    </div>
  </body>
</html>
```

Affichage obtenu :

### Résumé

Ces travaux ont pour objectif essentiel de faire ceci, cela, et encore cela, choses fondamentales pour remplir ce magnifique rapport que vous lisez en moment.

Bla bla bla.

*Mots-clés* : Bla.

# L'élément Div (suite)

Les blocs constitués par les éléments div peuvent être positionnés au pixel près.

```
<div id="bloc1" style="position: absolute; left: 10; top: 10;
    color : green; font-size: 0.7cm ">
  <p>
    Contenu du bloc 1
  </p>
</div>
<div id="bloc2" style="position: absolute; left: 8; top: 9;
    color : red; font-size: 0.5cm ">
  <p>
    *** Le bloc 2 ***
  </p>
</div>
```

Résultat obtenu :



# L'élément span

L'élément span est similaire à l'élément div. On utilise habituellement div pour des blocs de texte entiers et span pour des petites portions de texte (mots ou lettres).

# L'attribut media

Il s'agit d'un attribut de la balise link. Avec cet attribut, on demande au navigateur de charger une feuille de style différente en fonction du medium de sortie. On peut ainsi écrire :

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

Cette utilisation des feuilles de style est à privilégier sur des solutions où l'on met à la disposition de l'internaute une page "normale" pour l'affichage et une page "spéciale" pour l'impression.

D'autres types de media ont une valeur standard : projection, handheld, etc.

## Plan

1 Introduction - Généralités

2 HTML

3 CSS

- Bon croquis > long discours
- Exemple introductif
- Classes et Identificateurs
- Div et Span
- Attributs et Valeurs

## Quelques propriétés et valeurs

Propriété	Valeurs possibles	Description
font-size	small, medium, large. Taille définie en pixels (px), en cm (cm), en mm (mm), en pourcentage (%)	Taille de la police
font-family	serif, sans-serif, cursive, ... Nom de la police (Arial, Verdana, etc.)	Type de police
font-weight	normal, bold, bolder, lighter	Poids de la police
font-style	normal, italic, oblique	Style de la police

Propriété	Valeurs possibles	Description
color	Nom ou valeur hexadécimale de la couleur.	Couleur de la police
background-image	URL du fichier. Exemple : url(marble.jpg)	Image d'arrière plan d'un élément
background-repeat	repeat, no-repeat, repeat-x, repeat-y	Répétition de l'image d'arrière-plan
background-color	Nom ou valeur hexadécimale de la couleur.	Couleur d'arrière-plan d'un élément

## Quelques propriétés et valeurs

Propriété	Valeurs possibles	Description
text-align	left, right, center, justify	Alignement du texte
text-indent	Valeur en pixels ou en pourcentage	Retrait de la première ligne
text-decoration	none, underline, overline	Décoration du texte
text-transform	none, capitalize, uppercase, lowercase	Casse du texte

## Unités et Couleurs

- Il existe plusieurs unités pour les valeurs:
  - em (taille de la fonte), pt (point), px (pixel), % (pourcentage)
- De même, il existe plusieurs manières de décrire une couleur:
  - red
  - rgb(255, 0, 0)
  - rgb(100%, 0%, 0%)
  - #ff0000
  - #f00

## CSS: Texte et Polices

- **Fontes:** font-family, font-size, font-weight, font-style, font-variant
- **Mise en page:** line-height, letter-spacing, word-spacing, text-align, text-decoration, text-indent, text-transform, vertical-align, white-space

```
body {
  font-family: arial;
  font-size: 0.8em;
}

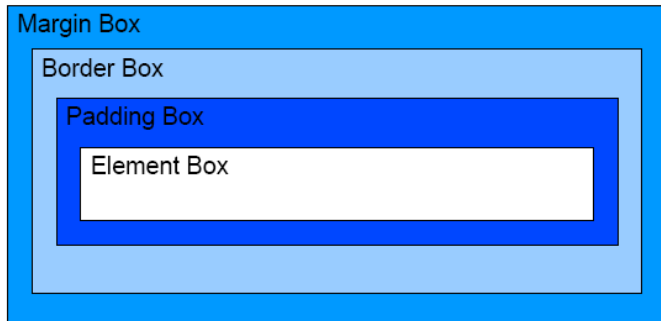
h1 { font-size: 2em; }
h2 { font-size: 1.5em; }
a { text-decoration: none; }

strong {
  font-style: italic;
  text-transform: uppercase;
}

p {
  letter-spacing: 0.5em;
  word-spacing: 2em;
  line-height: 1.5;
  text-align: center;
}
```

# CSS: Marge et Boîte englobante

- Un élément bloc possède:
  - une marge (`margin-*`)
  - une boîte englobante (`border-*`)
    - La boîte est configurable graphiquement (couleur, motifs des lignes...)
  - un alignement dans la boîte englobante (`padding-*`)



# Les liens hypertextes

On dispose de facilités pour attribuer un style aux différents liens hypertextes d'une page HTML.

Exemple :

```
a:link { color: red }      /* Style pour des liens non visités */
a:visited { color: blue } /* Style pour des liens déjà visités */
a:hover { color: green }  /* Style lorsque le pointeur de la souris survole le lien */
```

```
<!DOCTYPE ...>
<html>
  <head>
    <title>Titre</title>
    <style type="text/css">
      a:link { color: red }
      a:visited { color: blue }
      a:hover { color: green }
    </style>
  </head>
  <body>
    <a href="asuivre.html">
      A suivre...
    </a>
  </body>
</html>
```

« link », « visited », etc. sont appelées des pseudo-classes.

# CSS: classe et ID (1)

- Pour l'instant, nous avons vu comment appliquer des sélecteurs sur les éléments HTML
- Nous pouvons définir nos propres sélecteurs:
  - class (#) : permet de définir un sélecteur associé à plusieurs éléments
  - id (.) : permet de définir un sélecteur à un élément

```
<div id="top">
  <h1>Chocolate curry</h1>
  <p class="intro">The Recipe...</p>
  <p class="intro">Mmm mm mmmmm</p>
</div>
```

```
#top {
  background-color: #ccc;
  padding: 1em
}

.intro {
  color: red;
  font-weight: bold;
}
```

# CSS: classe et ID (2)

- Il est même possible d'optimiser:
  - si le document HTML / XHTML est bien formé, le CSS peut prendre compte de l'emboîtement de structures

```
<div id="top">
  <h1>Chocolate curry</h1>
  <p>The Recipe...</p>
  <p>Mmm mm mmmmm</p>
</div>
```

```
#top {
  background-color: #ccc;
  padding: 1em
}

#top h1 {
  color: #ff0;
}

#top p {
  color: red;
  font-weight: bold;
}
```