

# **Programmation WEB : HTML**

G. MOROY

[gautier.moroy@univ-paris-diderot.fr](mailto:gautier.moroy@univ-paris-diderot.fr)

Bâtiment Lamarck A, 5<sup>ème</sup> étage, bureau 514

2017-2018

# Généralités

## Internet :

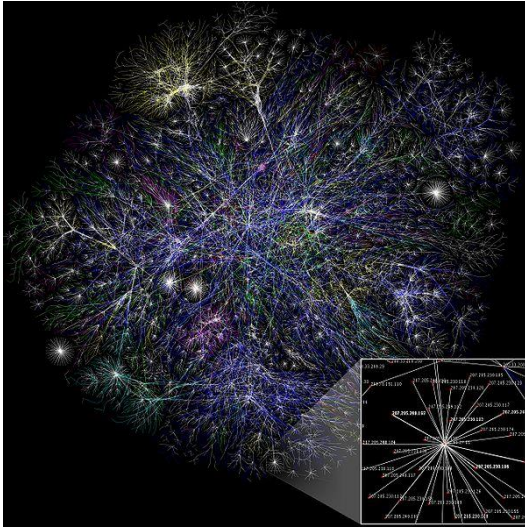
Réseau constitué d'ordinateurs pouvant communiquer les uns avec les autres grâce aux protocoles TCP/IP (Transmission Control Protocol / Internet Protocol).

Est utilisé par plusieurs applications :

- la messagerie instantanée
- le courrier électronique
- le World Wide Web (~ grande toile d'araignée)

## Page Web :

- Document contenant des informations et des hyperliens.
- Hébergé ou stocké sur des serveurs.
- Localisé par son URL (Uniform Resource Locator).
- Consulté via un protocole de communication (protocole HTTP) .
- Affiché par un navigateur (IE, Safari, Firefox, Chrome ...) pour l'utilisateur ("client").

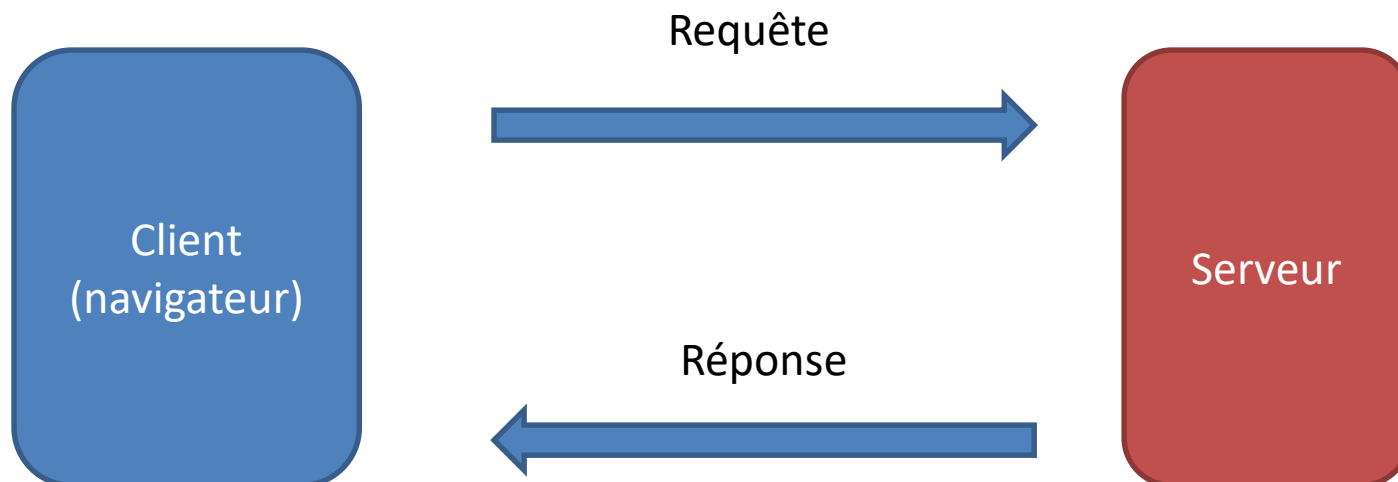


# Protocole HTTP

HTTP (**H**yper**T**ext **T**ransfert **P**rotocole) : protocole de transfert hypertexte.

Système hypertexte : réseau contenant des nœuds d'information (textes, images, sons ...) reliés entre eux par des hyperliens qui permettent de passer de l'un à l'autre.

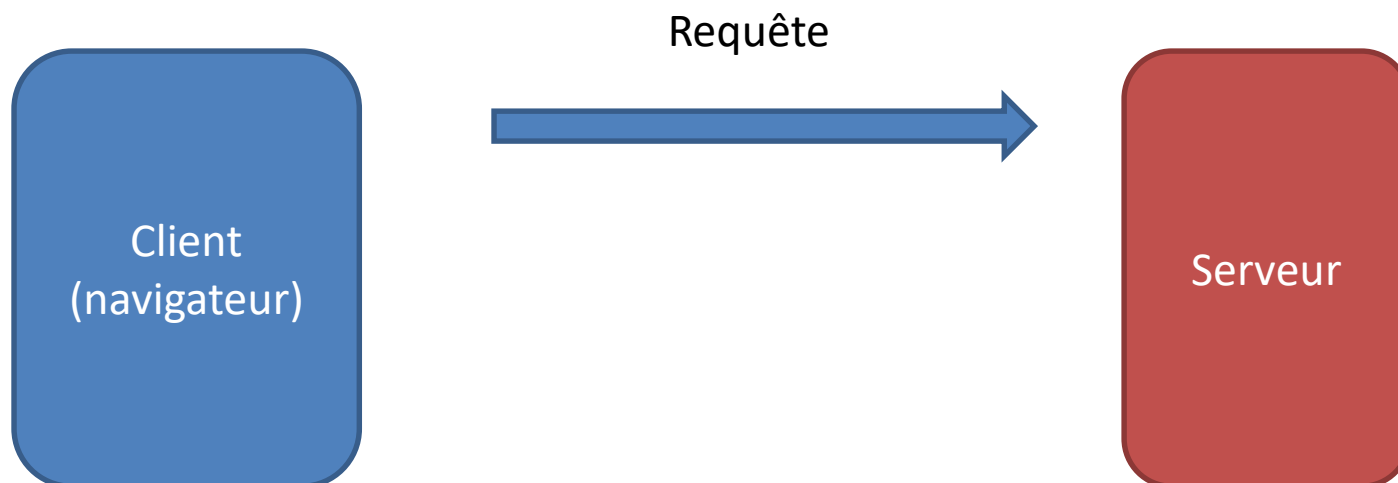
But : permettre des échanges entre client et serveur (protocole le plus utilisé pour la navigation web).



# Protocole HTTP

- Requête :
- ligne de requête :
    - la méthode utilisée (GET, POST, ...)
    - l'URL (Uniform Resource Locator)
    - la version du protocole utilisé (généralement *HTTP/1.0*)
  - en-tête de requête : informations facultatives (par ex: navigateur utilisé)
  - corps de requête: informations optionnelles (par ex: les valeurs des variables)

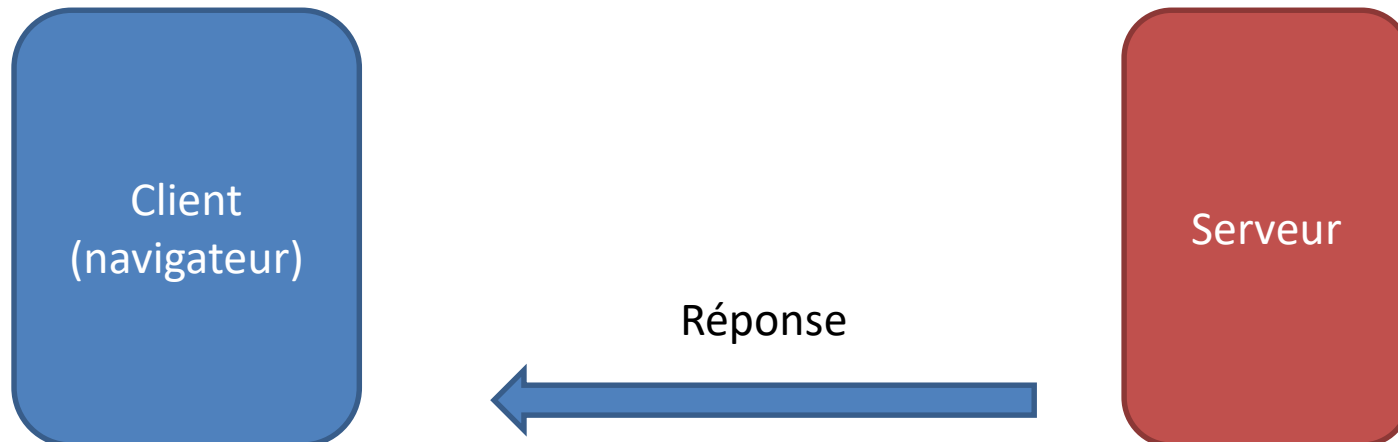
Exemple : `GET http://www.google.fr HTTP/1.0`



# Protocole HTTP

- Réponse :
- ligne d'état :  
*HTTP/1.0*
  - la version du protocole utilisée (généralement *HTTP/1.0*)
  - le code numérique de la réponse
  - la signification du code numérique
  - en-tête de réponse : informations facultatives (type de serveur, date ...)
  - corps de réponse : contient le document désiré

Exemple : `HTTP/1.0 200 OK`  
`HTTP/1.0 404 Not found`



# Langage HTML

## généralités

HTML (**H**yper **T**ext **M**arkup **L**anguage) :

- langage à balise ( $\neq$  langage de programmation).
- Balise : mot-clé prédéfini pour mettre en forme le texte, les images ...
- défini des hyperliens pour naviguer entre les différents documents.
- permet la conception de page web.

# Langage HTML

## balise HTML

### Balise HTML :

- mot-clé délimité par les symboles "<" et ">".

Exemple : `<balise>`

- fonctionne par paire avec une balise d'ouverture et une balise de fermeture.

Exemple : `<balise> Texte </balise>`

NB: Certaines balises d'ouverture n'ont pas besoin de balise de fermeture. Dans ce cas, on symbolise la fermeture : `<br />`

- est insensible à la casse : `<BALISE> = <balise> = <BaLiSe>`
- peuvent posséder un attribut donnant une information supplémentaire

Il est conseillé d'utiliser des sauts de ligne, d'indenter son texte et de le commenter pour rendre le code plus lisible

# Langage HTML

structure générale

- balise `<html>` : indique qu'il s'agit d'un document HTML
- balise `<head>` : l'en-tête : donne des informations valables pour l'ensemble du document
- balise `<body>` : corps du document, son contenu

```
<html>

    <head>
    </head>

    <body>
    </body>

</html>
```



# Langage HTML

## commentaires

- Commentaire :
- important pour la relecture du code.
  - invisible lors de la visualisation de la page web.

```
<!-- commentaire -->
```

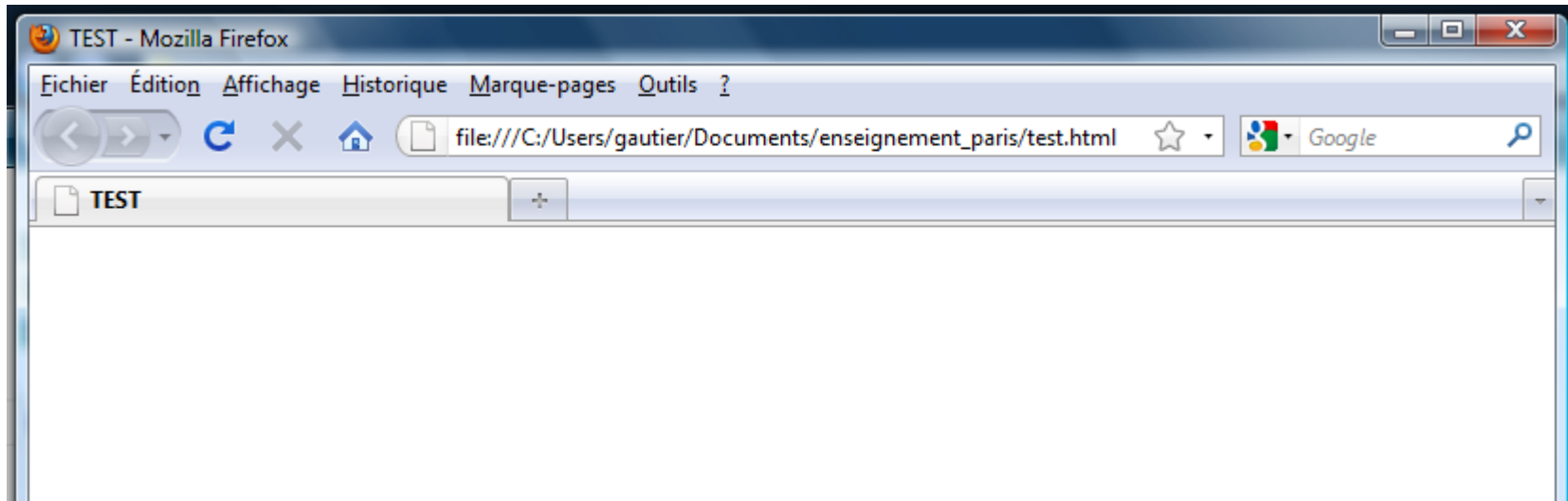
```
<!-- deux lignes  
de commentaire -->
```

# Langage HTML

en-tête / titre

Balise `<title>` permet de faire apparaître le titre de la page

```
<head> <title> TEST </title>  
</head>
```



# Langage HTML

en-tête / indexation

## Indexation :

- permet aux moteurs de recherche de référencer les sites.
- dans le <head> grâce à la balise <meta>
- attributs possibles :

auteur du site : <meta name="author" content="Bill Gates" />

langage du site : <meta http-equiv="content-language" content="fr" />

mots-clés : <meta name="keywords" content="..." />

description : <meta name="description" content="..." />

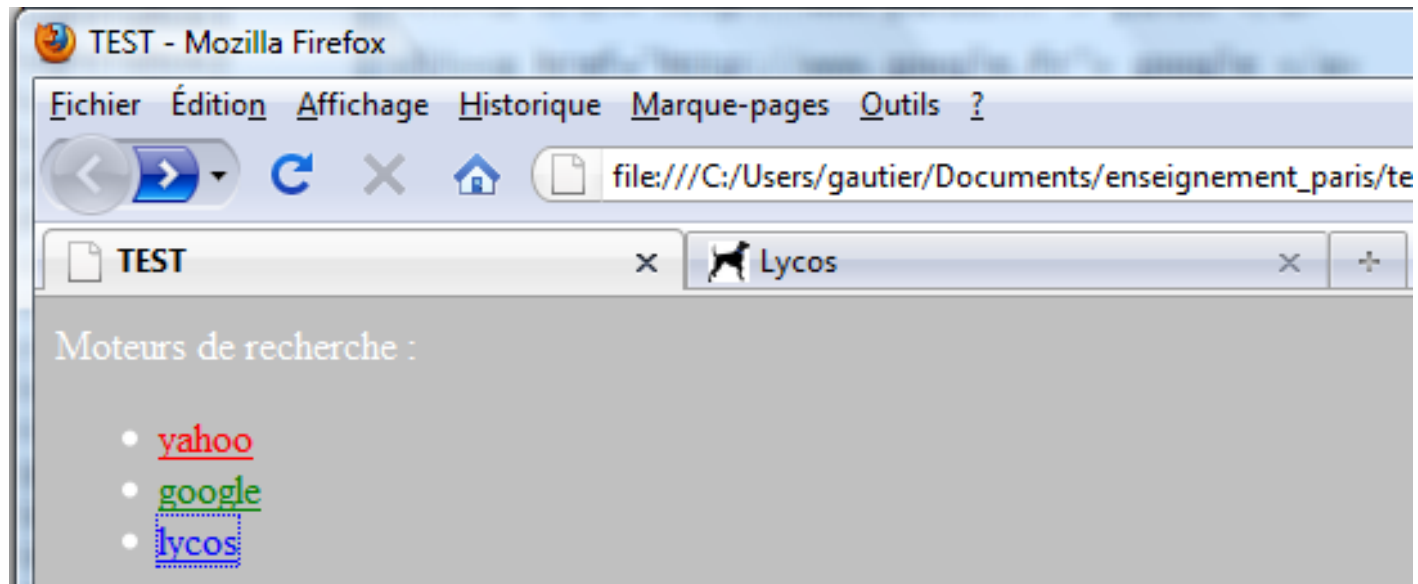
# Langage HTML

corps

Attributs de la balise <body> :

- bgcolor : couleur du fond
- text : couleur du texte
- link : couleur des liens non visités
- vlink : couleur des liens visités
- alink : couleur des liens actifs

```
<body bgcolor="silver" text="white" link="red" vlink="green" alink="blue">
```



# Langage HTML

## couleurs

Affichage :

- par nom : `bgcolor="silver"`

- par code RGB (**R**ed **G**reen **B**lue) :

la valeur de chaque couleur pouvant varier de 0 à 255, exprimé en format hexadécimal :

`rouge : bgcolor="#ff0000"`

`blanc : bgcolor="#ffffff"`

`rose : bgcolor="#ffc0cb"`

...

Liens : - <http://www.code-couleur.com/>  
- <http://html-color-codes.info/>  
- <http://www.computerhope.com/htmcolor.htm>

# Langage HTML

texte / mise en page

Navigateurs : plusieurs espaces à la suite comptabilisés comme un seul  
retours à la ligne présents dans le code HTML sont ignorés

Paragraphe : `<p> Texte </p>` (impose un retour à la ligne à la fin)

attributs : - `title` : info-bulle contenant du texte lorsque la souris est sur le texte.

- `align` : position du texte dans le paragraphe (`left`, `right`, `center` ou `justify`).

Retour à la ligne : `<br />`

Ligne : `<hr />`

# Langage HTML

texte / mise en page

**Texte en gras :** `<b>` ou `<strong>`

**Agrandir le texte :** `<big>`

Diminuer le texte : `<small>`

*Texte en italique :* `<em>` ou `<i>`

Texte en indice : `<sub>`

Texte en exposant : `<sup>`

Texte souligné : `<ins>`

~~Texte barré :~~ `<del>`

Titres :

`<h1>` Titre 1 `</h1>`

`<h2>` Titre 2 `</h2>`

`<h3>` Titre 3 `</h3>`

`<h4>` Titre 4 `</h4>`

`<h5>` Titre 5 `</h5>`

`<h6>` Titre 6 `</h6>`



# Langage HTML

texte / mise en page

Pour les acronymes :

balises `<abbr>` ou `<acronym>` avec l'attribut `title`

Exemple : `<abbr title="Fédération Française de Football">FFF</abbr>`

Pour les citations :

balise `<blockquote>`

Pour les adresses :

balise `<address>`



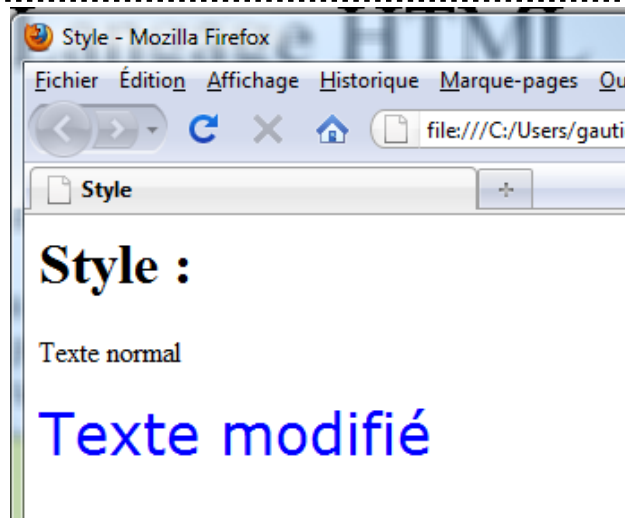
# Langage HTML

texte / mise en page

La balise <font> permet de modifier et d'embellir le texte, ses attributs sont :

- color : pour la couleur du texte.
- face : pour la police de caractère.
- size : pour la taille des caractères.

```
<h1> Style :</h1>
<p> Texte normal </p>
<p><font face="verdana" size="5" color="blue"> Texte modifié </font></p>
```



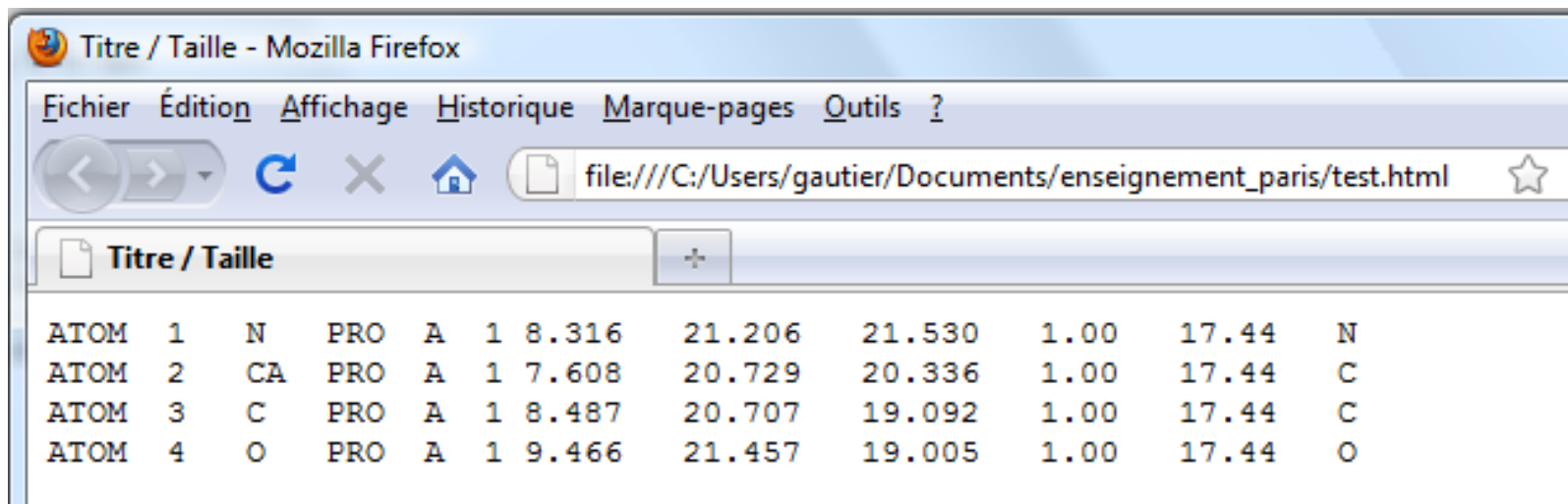
/!\ balise remplacée par le CSS

# Langage HTML

texte / mise en page

La balise `<pre>` conserve les espaces et les retours à la ligne.

```
<pre>
ATOM  1      N      PRO  A   1  8.316      21.206      21.530      1.00      17.44      N
ATOM  2      CA      PRO  A   1  7.608      20.729      20.336      1.00      17.44      C
ATOM  3      C       PRO  A   1  8.487      20.707      19.092      1.00      17.44      C
ATOM  4      O       PRO  A   1  9.466      21.457      19.005      1.00      17.44      O
</pre>
```



# Langage HTML

texte / mise en page

L'utilisation d'accent peut poser des problèmes aux navigateurs non-français.

Symbole	Code ISO	Code HTML
à	&#224;	&agrave;
Å	&#197;	&Aring;
ç	&#231;	&ccedil;
è	&#232;	&egrave;
é	&#233;	&eacute;
ê	&#234;	&ecirc;
î	&#238;	&icirc;
€	&#8364;	&euro;
œ	&#339;	&oelig;
<	&#60;	&lt;
>	&#62;	&gt;
@	&#64;	
α	&#945;	&alpha;
β	&#946;	&beta;

# Langage HTML

## liens

La balise `<a>` permet de gérer les liens externes et internes. Le lien peut être du texte, une image ...

Syntaxe générale : `<a href="adresse"> lien </a>`

### Lien externe

```
<a href="http://www.google.fr"> Google </a>
<a href="./accueil.htm">  Accueil </a>
```

Lien interne : on peut définir un endroit du document HTML en avec l'attribut `name` et le cibler avec l'attribut `target` (sensible à la casse).

```
<a name="Introduction"> Introduction </a>
. . .
<a href="#Introduction"> Aller à l'introduction </a>
```

# Langage HTML

## liens

Insérer un lien vers une adresse électronique :

```
<a href="mailto:gautier.moroy@univ-paris-diderot.fr"> Envoyez-moi un mail </a>
```

# Langage HTML

## image

La balise `<img>` contrôle l'introduction des images avec les attributs suivant :

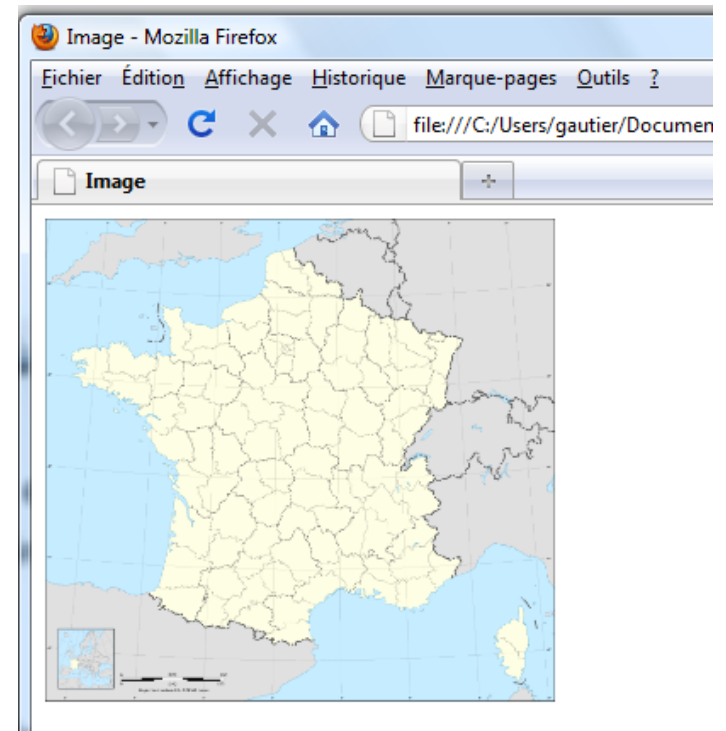
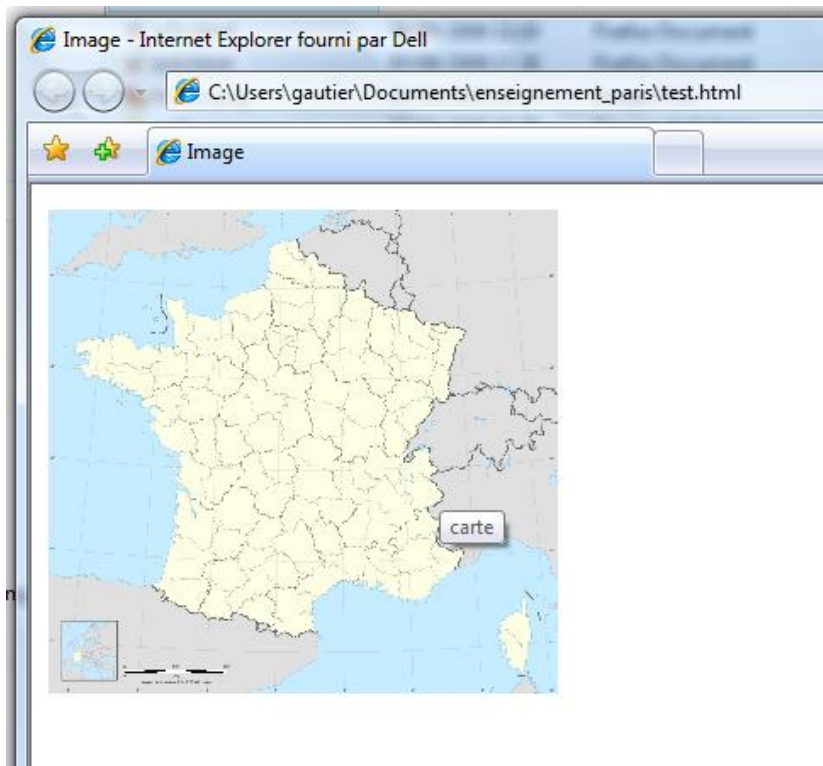
- `src` : donne l'URL (relative ou absolue) de l'image.
- `height` : spécifie la hauteur de l'image en pixel ou en % de la page.
- `width` : spécifie la largeur de l'image en pixel ou en % de la page.
- `align` : position de l'image suivant les autres éléments (`left`, `right`, `top`, `middle` and `bottom`).
- `border` : épaisseur de la bordure de l'image en pixel
- `alt` : texte apparaissant si l'image ne s'affiche pas.
- `title` : info-bulle contenant du texte lorsque la souris est dessus.
- `usemap` : donne un nom à l'image pour la balise `<map>`.

# Langage HTML

image

```

```

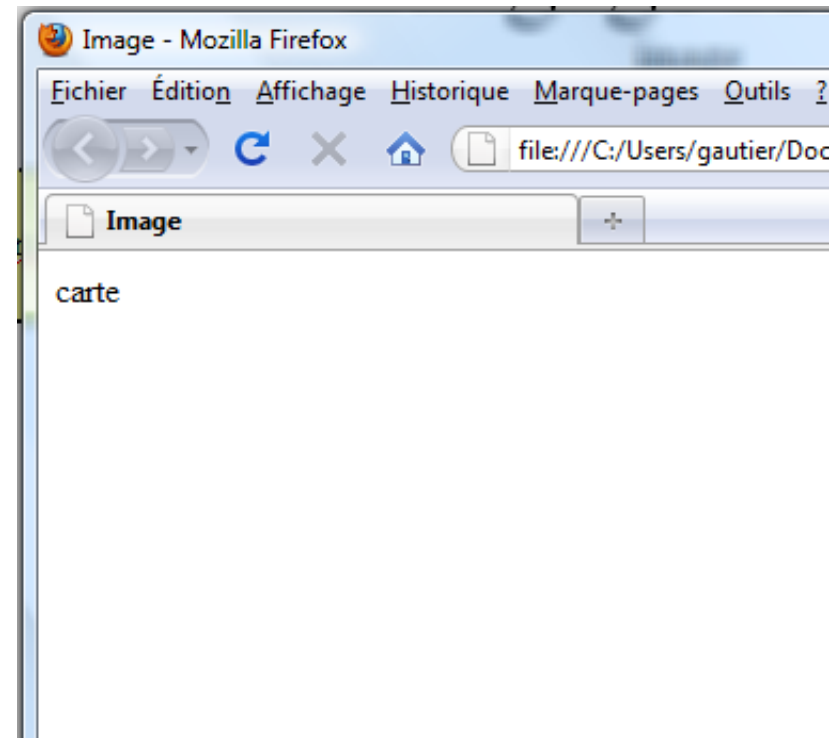
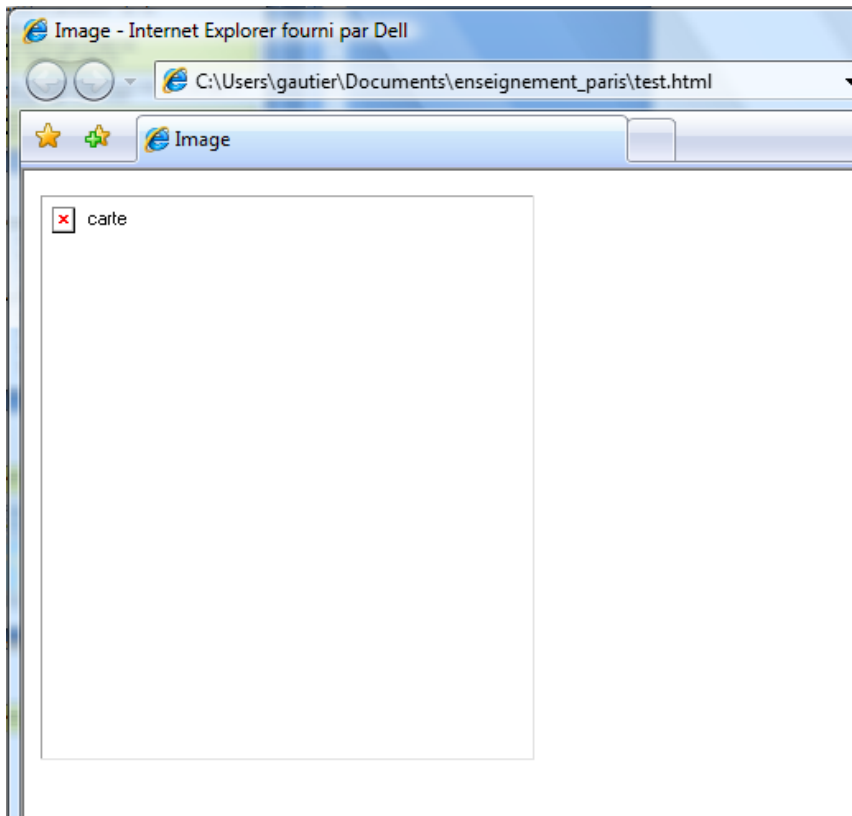


# Langage HTML

## image

```

```



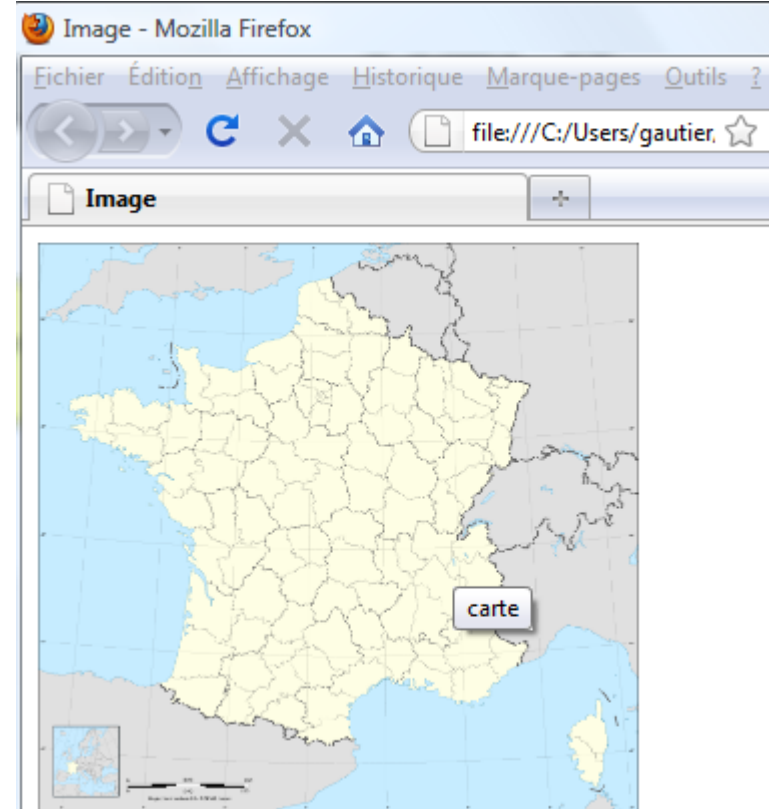
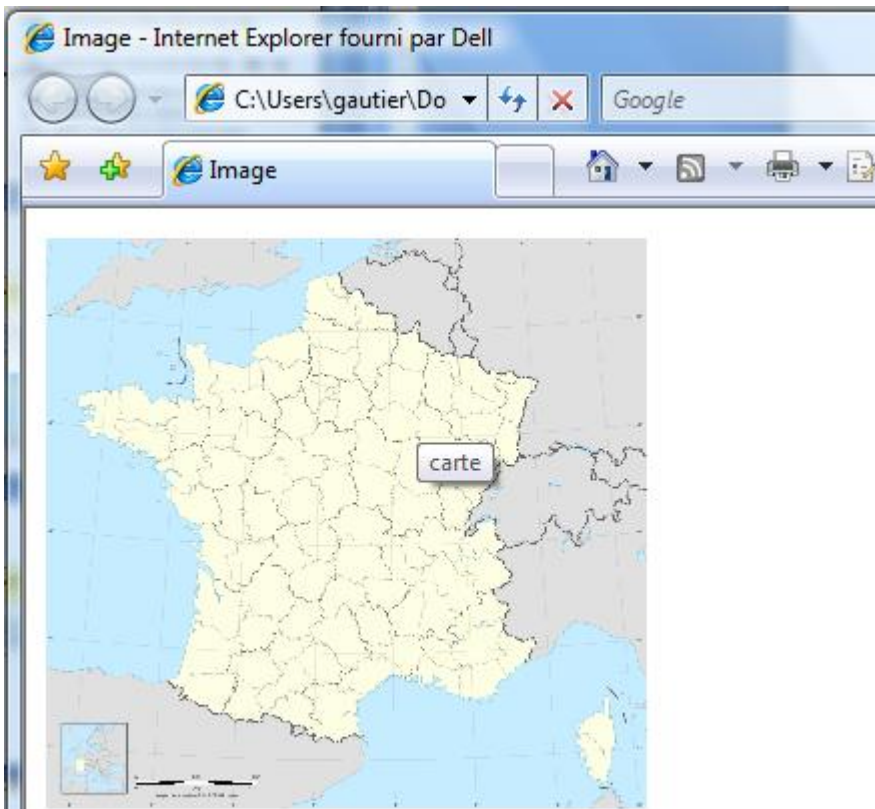


# Langage HTML

image

```

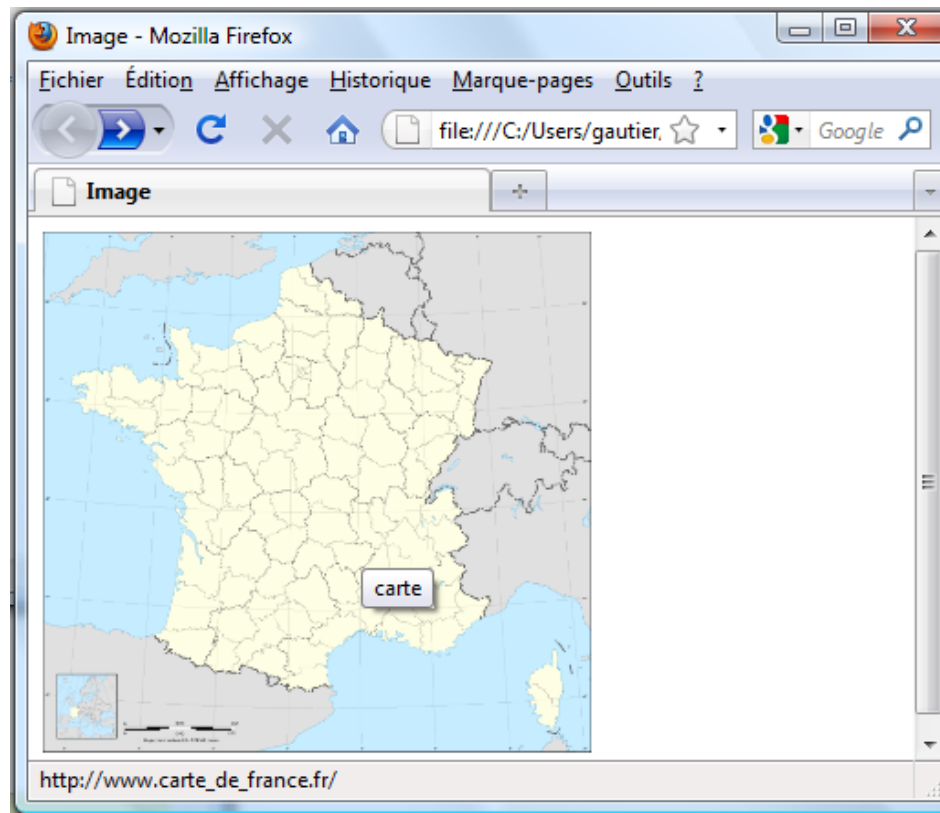
```



# Langage HTML

image

```
<a href="http://www.carte_de_france.fr">  
  
</a>
```



# Langage HTML

## image

Il est possible de "cartographier" une image avec la balise `<map>`, avec l'attribut `name` qui fait référence au nom donné avec l'attribut `usemap` de la balise `<img>`.

La balise `<area>` permet de préciser des zones sur l'image, avec les attributs :

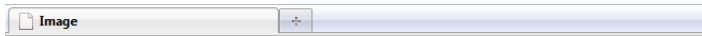
- `title` : info-bulle contenant du texte lorsque la souris est dessus.
- `href` : URL
- `nohref` : indique que la zone n'a pas de liens
- `coords` : donne les coordonnées de la zone
- `shape` : précise la forme de la zone :
  - `rect` : zone rectangulaire (2 points )
  - `circle` : zone circulaire (1 point + 1 rayon)
  - `poly` : zone à définir (plusieurs points)

# Langage HTML

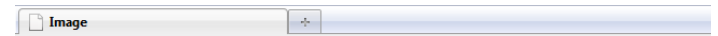
## image

```
<a href="http://www.france.org">
  
</a>

<map name="carte">
  <area shape="rect" coords="294,155,309,172" title="Paris" href="http://www.paris.fr/" />
</map>
```



Une image :



Une image :



# Langage HTML

## tableau

Balise `<table>` permet de générer un tableau :

- attribut `border` : épaisseur des lignes
- attribut `cellpadding` : nombre de pixel entre le texte et les lignes.
- attribut `cellspacing` : nombre de pixel entre les cellules.
- attribut `width` : taille du tableau en pixel ou % de la page.
- attribut `align` : position du tableau (`left`, `center` ou `right`).
- attribut `bgcolor` : couleur du tableau.

Balise `<tr>` : définit les lignes.

- attribut `align` : position horizontale (`right`, `left`, `center` ou `justify`)
- attribut `valign` : position vertical (`top`, `middle`, `bottom` ou `baseline`)
- attribut `bgcolor` : couleur du tableau.

# Langage HTML

## tableau

Balise `<td>` :

- attribut `align` : position horizontale (`right`, `left`, `center` ou `justify`)
- attribut `valign` : position vertical (`top`, `middle`, `bottom` ou `baseline`)
- attribut `colspan` : fusionne verticalement plusieurs cellules.
- attribut `rowspan` : fusionne horizontalement plusieurs cellules.
- attribut `width` : largeur de la ligne en pixel ou % de la page.
- attribut `height` : hauteur de la ligne en pixel ou % de la page.

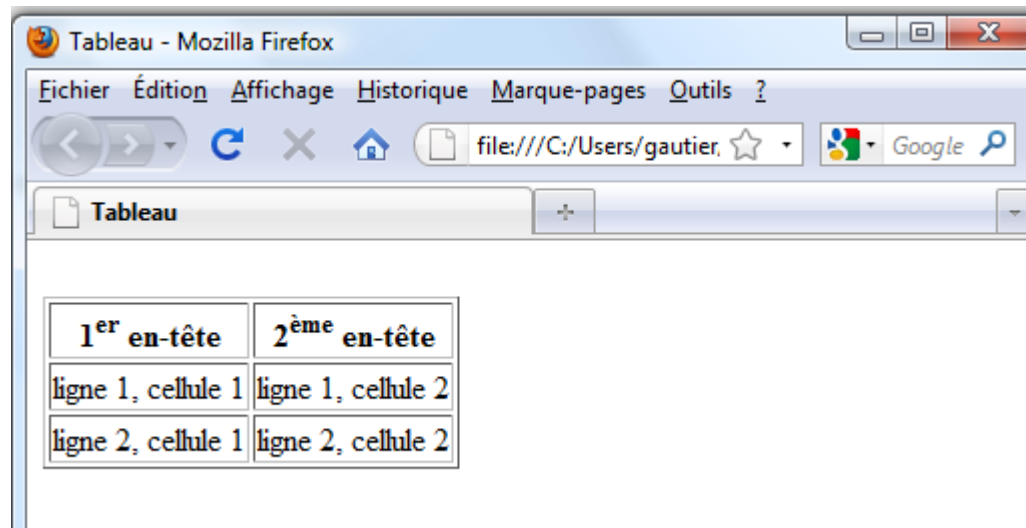
Balise `<th>` : définit les en-têtes (texte en gras et centré), même attributs que `<td>`.

Balise `<caption>` : donne une légende au tableau.

# Langage HTML

tableau /exemple

```
<table border="1">
<tr>
<th> 1<sup>er</sup> En-tête </th>
<th> 2<sup>ème</sup> en-tête </th>
</tr>
<tr>
<td> ligne 1, cellule 1</td>
<td> ligne 1, cellule 2</td>
</tr>
<tr>
<td> ligne 2, cellule 1</td>
<td> ligne 2, cellule 2</td>
</tr>
</table>
```



# Langage HTML

tableau / exemple fusion

```
<table border="1" align="left">

  <tr>
    <th> 1<sup>er</sup> en-tête </th>
    <th> 2<sup>ème</sup> en-tête </th>
  </tr>

  <tr align="center" valign="center">
    <td rowspan="2"> ligne 1, cellule 1</td>
    <td> ligne 1, cellule 2</td>
  </tr>

  <tr align="center" valign="center">
    <td> ligne 2, cellule 2</td>
  </tr>

</table>
```

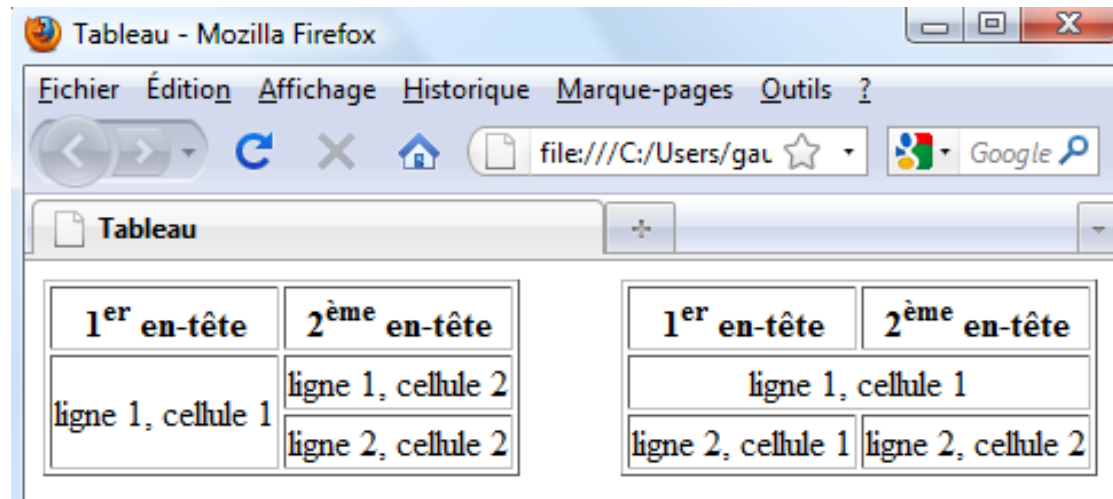
```
<table border="1" align="right">

  <tr>
    <th> 1<sup>er</sup> en-tête </th>
    <th> 2<sup>ème</sup> en-tête </th>
  </tr>

  <tr align="center" valign="center">
    <td colspan="2"> ligne 1, cellule 1</td>
  </tr>

  <tr align="center" valign="center">
    <td> ligne 2, cellule 1</td>
    <td> ligne 2, cellule 2</td>
  </tr>

</table>
```



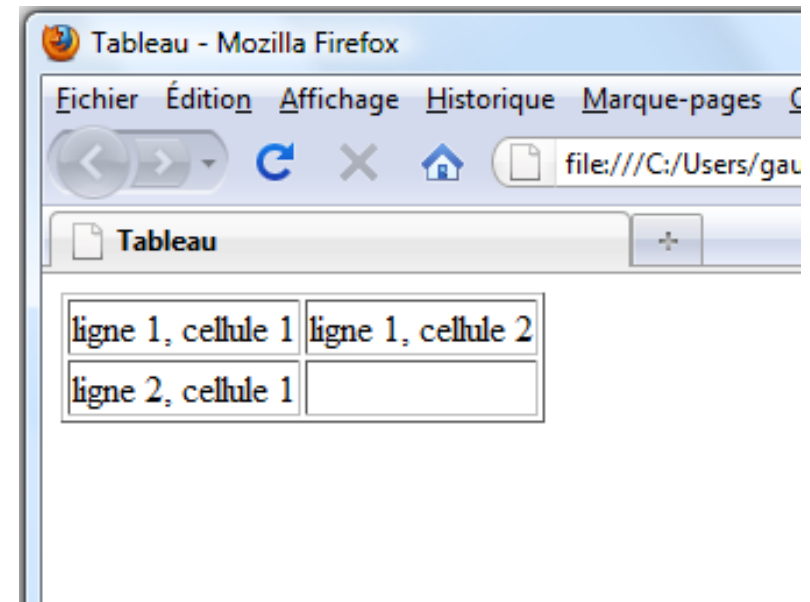
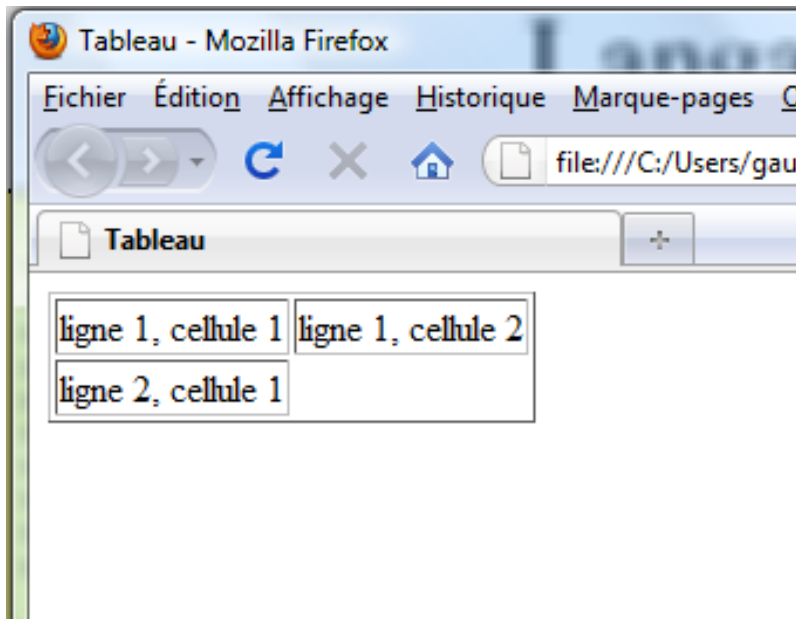


# Langage HTML

tableau / cellule vide

```
<table border="1">
<tr>
<td> ligne 1, cellule 1</td>
<td> ligne 1, cellule 2</td>
</tr>
<tr>
<td> ligne 2, cellule 1</td>
<td></td>
</tr>
</table>
```

```
<table border="1">
<tr>
<td> ligne 1, cellule 1</td>
<td> ligne 1, cellule 2</td>
</tr>
<tr>
<td> ligne 2, cellule 1</td>
<!-- Ajout du caractère espace -->
<td>&nbsp;</td>
</tr>
</table>
```



# Langage HTML

## liste

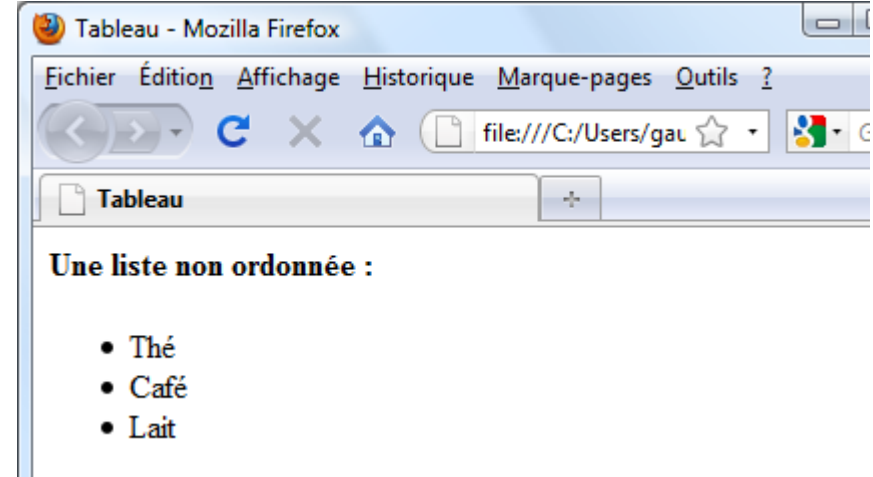
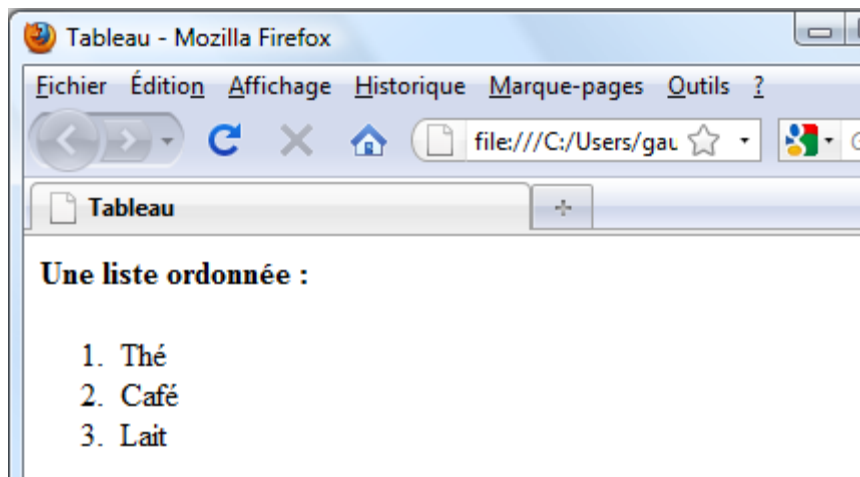
Balise `<ol>` : liste ordonnée

Balise `<ul>` : liste non ordonnée.

Balise `<li>` : élément de la liste.

```
<h4> Une liste ordonnée :</h4>
<ol>
  <li>Thé</li>
  <li>Café</li>
  <li>Lait</li>
</ol>
```

```
<h4> Une liste non ordonnée :</h4>
<ul>
  <li>Thé</li>
  <li>Café</li>
  <li>Lait</li>
</ul>
```



# Langage HTML

## cadre

Il est possible de diviser la page HTML en plusieurs pages HTML grâce à la balise `<frameset>`. Chaque page HTML est indépendante des autres.

La balise `<frameset>` permet de définir comment se fait la séparation de la page HTML :

- la page HTML ne doit être contenu dans le corps (`<body>`) de la page.
- attribut `rows` : séparation horizontale, en pixel, % ou le symbole \*.
- attribut `cols` : séparation verticale, en pixel, % ou le symbole \*.

Balise `<frame>` définit les pages HTML à insérer dans `<frameset>` :

- attribut `name` : donne le nom au cadre.
- attribut `src` : donne l'URL de la page HTML à insérer.
- attribut `scrolling` : représentation de l'ascenseur (yes ou no).
- attribut `noresize` : interdit de modifier la taille des cadres.

# Langage HTML

cadre / exemple

```
</html>
<head> <title> Cadre / exemple </title> </head>

<frameset cols="10%,20%,*">
  <frame src="./gauche.html" />
  <frame src="./centre.html" />
  <frame src="./droite.html" />
</frameset>

</html>
```

gauche.html :

```
<html>
<head> <title> Gauche </title> </head>
<body>
<h1> Cadre de gauche </h1>
</body>
</html>
```

droite.html :

```
<html>
<head> <title> Droite </title> </head>
<body>
<h1> Cadre de droite</h1>
</body>
</html>
```

```
<html>
<head> <title> Centre </title> </head>
<body>
<h1> Cadre du centre</h1>
</body>
</html>
```

centre.html

# Langage HTML

cadre / exemple



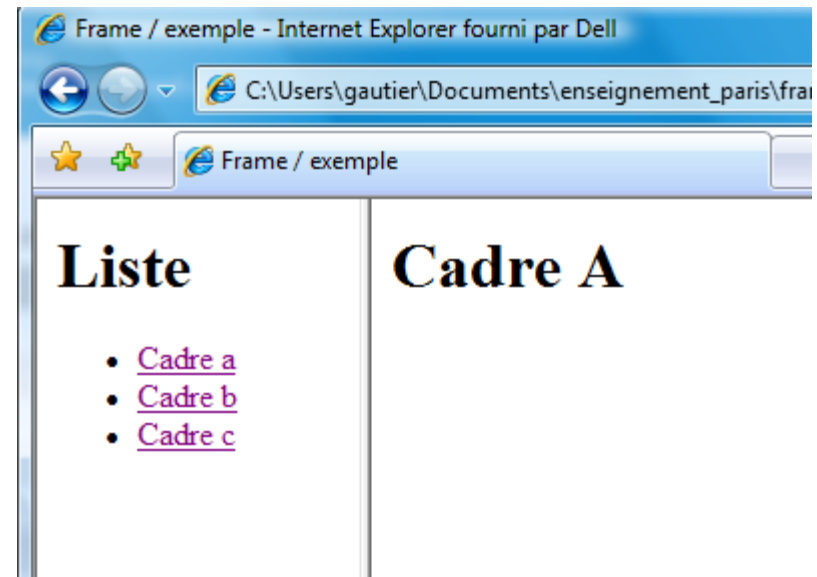
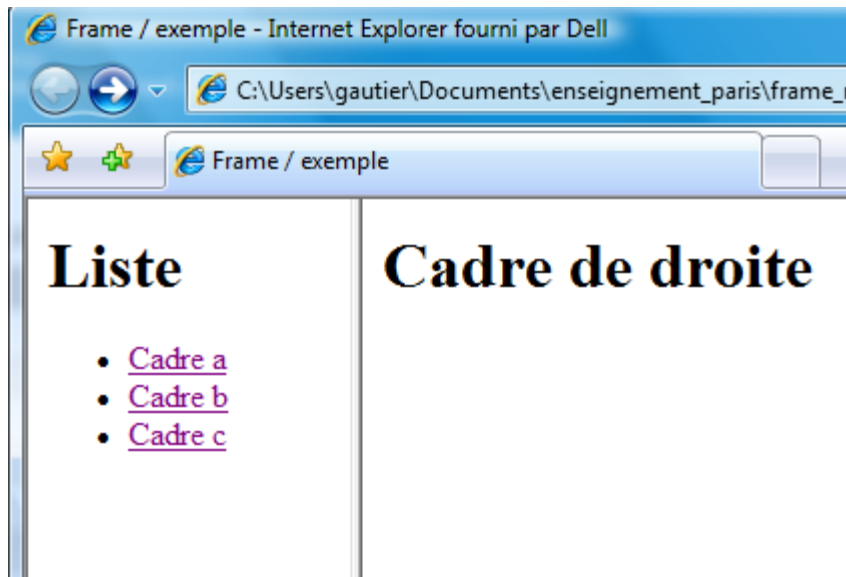
# Langage HTML

## cadre

```
<frameset cols="20%,*">  
  <frame src="./liste.html" name="liste">  
  <frame src="./droite.html" name="droite">  
</frameset>
```

liste.html :

```
<h1> Liste </h1>  
<ul>  
<li> <a href="cadre_a.html" target="droite"> Cadre a </a> </li>  
<li> <a href="cadre_b.html" target="droite"> Cadre b </a> </li>  
<li> <a href="cadre_c.html" target="droite"> Cadre c </a> </li>  
</ul>
```



# Langage HTML

## formulaire

Un formulaire est une zone dans laquelle on permet au visiteur de la page HTML d'y entrer des informations qui peuvent être traitées par le serveur.

Un formulaire est défini avec la balise `<form>`.

- attribut `name` : donne le nom du formulaire.

- attribut `action` : où envoyer les informations (email ou script)

**Exemple:** `<form action="mailto:gautier.moroy@univ-paris-diderot.fr">`  
`<form action="http://www.formulaire/analyse.html">`

- attribut `method` : précise par quelle méthode sont transmises les informations :
  - `get` : les informations sont transmises par l'URL.

**Exemple:**

`http://www.formulaire/analyse.html?champ1=valeur1&champ2=valeur2`

- `post` : les informations sont transmises dans le corps de la requête.

# Langage HTML

## formulaire

La balise `<input>` est la plus importante, ses attributs sont :

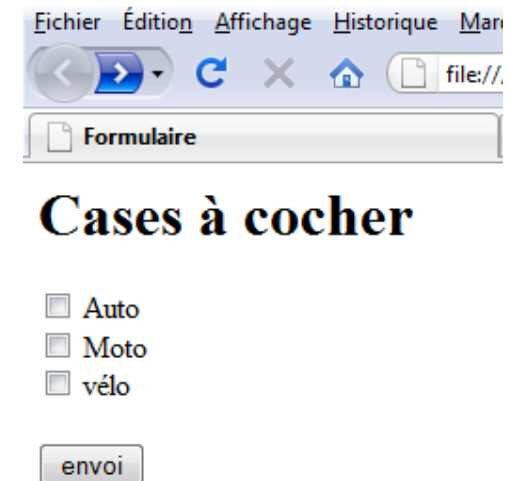
- name : nom du champ
- value : valeur par défaut
- type :
  - checkbox : case à cocher
  - password : champ de saisie (les caractères sont cachés)
  - radio : bouton
  - reset : efface le contenu du formulaire
  - submit : envoie les informations
  - text : zone de texte (+ size pour limiter la taille du texte)

```
<h1> Cases à cocher </h1>

<form action="./centre.html" method="get" name="case">

<input type="checkbox" name="auto" /> Auto <br />
<input type="checkbox" name="moto" /> Moto <br />
<input type="checkbox" name="velo" /> vélo <br /><br />

<input type="submit" value="envoi" />
```



Fichier Édition Affichage Historique Man

file://

Formulaire

## Cases à cocher

☐ Auto

☐ Moto

☐ vélo

envoi



# Langage HTML

## formulaire

La balise `<textarea>` définit une zone de saisie de texte plus importante que `<input>` :

- `cols` : nombre de caractères dans une ligne
- `name` : nom du champ
- `rows` : nombre de ligne

```
<h1> Zone de texte </h1>

<form action="./centre.html" method="get" name="formulaire">

<textarea name="texte" rows="3" cols="15"></textarea>

<input type="submit" value="envoi" />
```



# Langage HTML

## formulaire

La balise `<select>` crée une liste déroulante, dont les attributs sont :

- `multiple` : sélection multiple possible.
- `name` : nom du champ.
- `size` : nombre de ligne.

La balise `<option>` définit les éléments de la liste, attributs possible :

- `selected` : option sélectionnée par défaut.
- `value` : texte envoyé au serveur

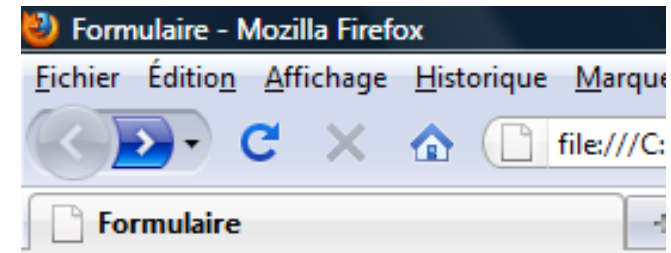
```
<h1> Liste déroulante </h1>

<form action="./centre.html" method="get" name="formulaire">

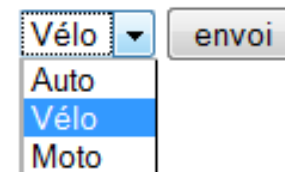
<select name="liste">
<option value="auto"> Auto </option>
<option value="velo" selected> Vélo </option>
<option value="moto" > Moto </option>
</select>

<input type="submit" value="envoi" />

</form>
```



## Liste déroulante



# **Programmation WEB : Feuille de style (CSS)**

# Feuille de style (CSS)

## généralités

Cascading Style Sheets (CSS) : feuille de style en cascade.

But : éviter de préciser pour chaque balise les différents attributs pour structurer la page web

→ modifications de la mise en page d'une page web rapide

→ meilleure lisibilité du code HTML

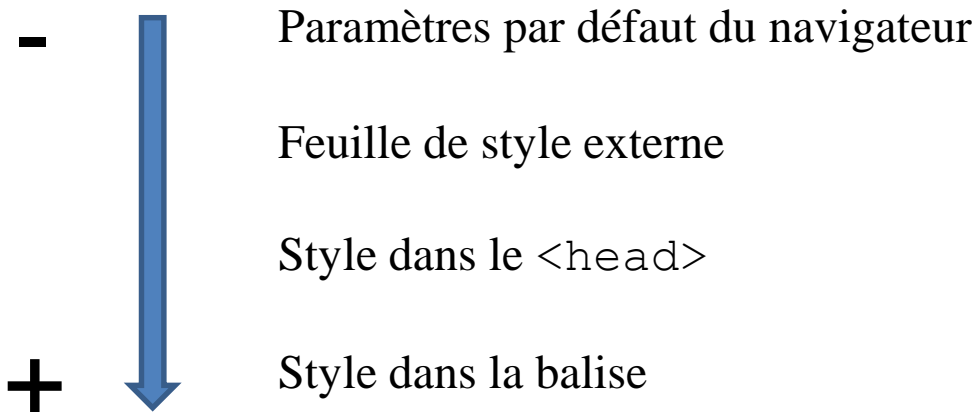
Peut être utilisé en tant que :

- attribut interne à une balise dans le corps du document.
- attribut interne à la balise `<head>`.
- fichier externe.

# Feuille de style (CSS)

## généralités

Les priorités d'une manière ou d'une autre de prendre en compte les CSS :



fichier externe

```
p { color:red; font-family:arial }
```

document HTML

```
<p style="color:green; text-align:center">
```

→ texte en arial, de couleur verte et centré

# Feuille de style (CSS)

attribut interne

On introduit l'attribut `style` dans les balises dans le corps du document:

```
<balise style="propriété : valeur"> </balise>
```

Dans le `<head>` :

- déclaration de l'utilisation de CSS : `<style type="text/css">`
- puis : `sélecteur { propriété_1 : valeur_1 ; propriété_2 : valeur_2 }`

Exemple :

```
<head>

  <style type="text/css">
    h1 {font-size: 30px; font-family: arial;}
    p {font-size: 18px;}
  </style>

</head>
```

# Feuille de style (CSS)

fichier externe

On utilise la balise `<link>` qui définit un lien entre le document HTML et d'autres fichier externe avec les attributs :

- `rel` : donne le style de fichier
- `type` : précise le type de feuille de style
- `href` : donne l'URL

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

Dans le fichier CSS, la syntaxe est :

```
sélecteur { propriété_1 : valeur_1 ; propriété_2 : valeur_2 }
```

```
p  
{  
text-align:center;  
color:red;  
font-family:arial  
}
```

# Feuille de style (CSS)

syntaxe (1)

On peut affecter les mêmes propriétés à différents sélecteurs en les séparant par "," :

```
h1,h2,h3,h4,h5,h6 { color:green }
```

Les sélecteurs de classe permettent de définir plusieurs styles différents :

```
selecteur.classe {propriété_1 : valeur_1 ; propriété_2 : valeur_2 }
```

```
p.centre {text-align:center;color:red}  
p.droite {text-align:right;color:green}
```

```
<p class="centre"> Texte centré rouge </p>  
<p class="droite"> Texte vert à droite </p>
```

/!\ ne pas commencer le nom de la classe par un nombre



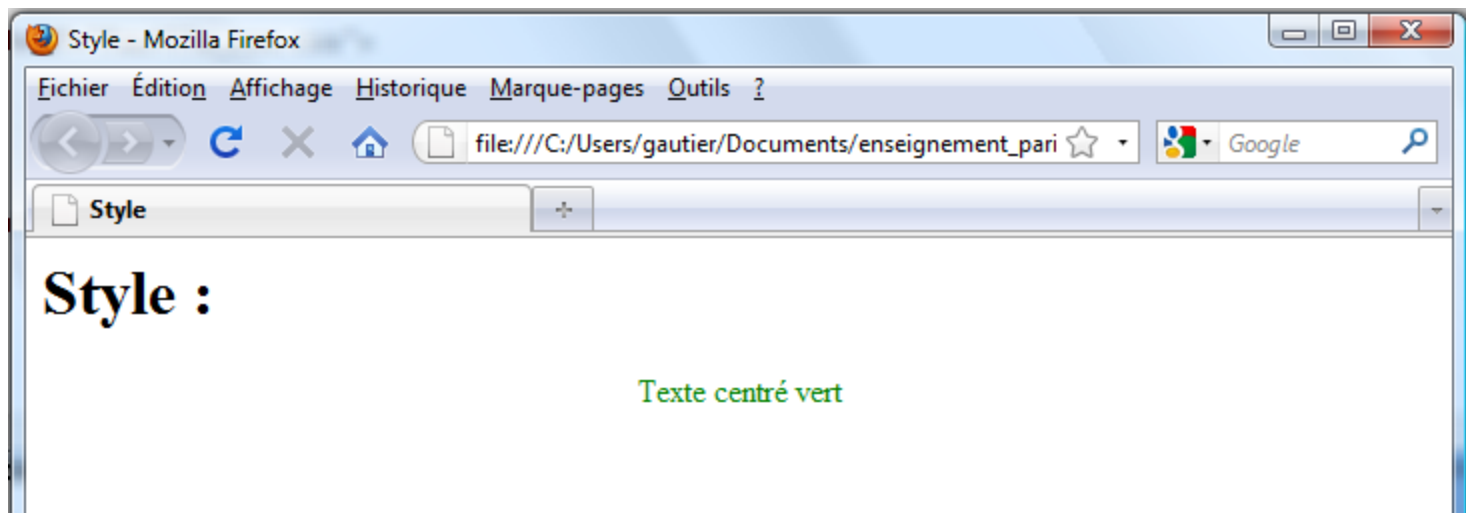
# Feuille de style (CSS)

syntaxe (2)

On peut combiner les classes :

```
p.centre {text-align:center}  
p.vert {color:green}
```

```
<p class="centre vert"> Texte centré vert </p>
```



# Feuille de style (CSS)

syntaxe (3)

Il est possible de ne pas spécifier de sélecteur à la classe :

```
.vert {color:green}
```

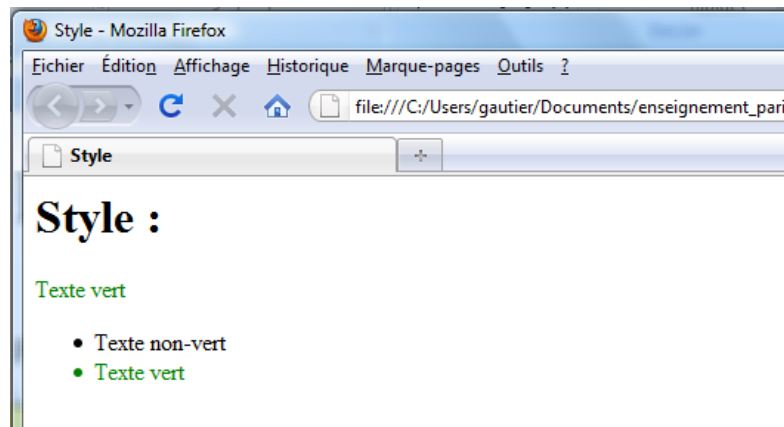
```
<p class="vert"> Texte vert</p>
```

```
<ul >
```

```
  <li> Texte non-vert </li>
```

```
  <li class="vert"> Texte vert </li>
```

```
</ul>
```



# Feuille de style (CSS)

syntaxe (4)

Les sélecteurs peuvent être combiner pour être :

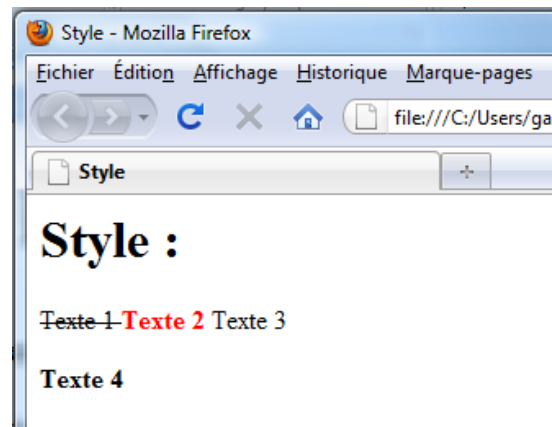
- imbriqués : sélecteur\_1 sélecteur\_2 { propriété : valeur\_1 }

Exemple :

```
p b { color:red; }
```

```
<p> <del> Texte 1 </del> <b> Texte 2 </b> Texte 3 </p>
```

```
<b> Texte 4 </b>
```



# Feuille de style (CSS)

syntaxe (5)

Les sélecteurs peuvent être combiner pour être :

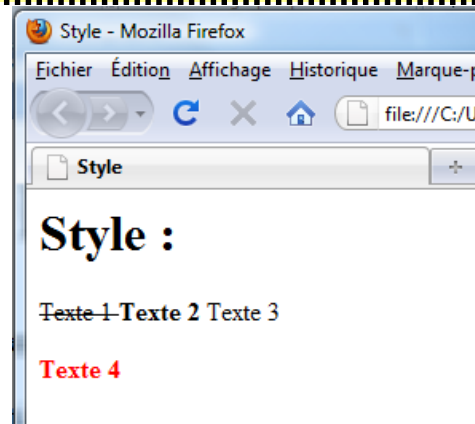
- consécutifs : sélecteur\_1 + sélecteur\_2 { propriété : valeur\_1 }

Exemple :

```
p + b { color:red; }
```

```
<p> <del> Texte 1 </del> <b> Texte 2 </b> Texte 3 </p>
```

```
<b> Texte 4 </b>
```



# Feuille de style (CSS)

## syntaxe (6)

Les commentaires entre "/\*" et "\*/" :

```
p.centre {text-align:center;color:red}  
  
/* Commentaire */  
  
p.droite {text-align:right;color:green}
```

# Feuille de style (CSS)

unité de longueur

Les unités de longueur des valeurs des propriétés peuvent être définies :

- de manière absolue : unité indépendante : `cm`, `in` (inch : 2,54 cm), `pt` ou `pc` (pica : 12 pt).
- de manière relative : dépendant de l'élément :
  - `em` : dépendant de la hauteur de la police (1 `em` = 100% de la taille de la police).
  - `px` : basé sur la résolution de l'écran
  - `%` : relative à la taille de l'élément

# Feuille de style (CSS)

Des balises ont été créées pour contrôler les styles :

- la balise `<span>` pour gérer des morceaux de paragraphes.
- la balise `<div>` pour affecter un style à un ou des paragraphes, ainsi que pour positionner des blocs d'éléments.

Exemple :

```
<div style="color:black">  
  
<p> Le bon sens est <span style="color:red">la chose du monde </span> la  
mieux partagée ... </p>  
  
<p> En quoi il n'est pas vraisemblable que tous se trompent ... </p>  
  
</div>
```

# Feuille de style (CSS)

body / propriétés (1)

Propriétés associés au sélecteur `<body>` :

- `background-color` : définit la couleur de l'arrière-plan de la page (nom ou code RGB)
- `background-image` : spécifie une image comme arrière-plan, valeurs :
  - `url('XXX.jpg')`
  - `none`
- `background-repeat` : contrôle comment doit être répéter l'image, valeurs :
  - `repeat` : valeur par défaut
  - `repeat-x` : répétition en ligne
  - `repeat-y` : répétition en colonne
  - `no-repeat`
- `background-attachment` : règle le comportement de l'image, valeurs :
  - `scroll` : l'image suit les déplacements de l'écran
  - `fixed` : l'image reste fixe

/!\ problème entre les navigateurs.



# Feuille de style (CSS)

body / propriétés (2)

Propriétés associés au sélecteur `<body>` :

- `background-position` : position de l'image, valeurs:
  - `top / center / bottom`    `left / center / right`
  - `X% Y%` : en % de la page, 0% 0% étant le coin haut gauche et 100% 100% le coin bas droite.
  - `X Y` : en unité : `px, cm ...` (0 0 étant le coin haut gauche)

Il est possible de spécifier toutes les propriétés en une seul, à conditions de respecter l'ordre  
`color image repeat attachment position`.

Exemple :

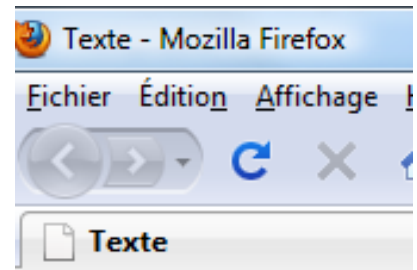
```
body {background:red url('bg-picture.jpg') no-repeat top left}
```

# Feuille de style (CSS)

texte / propriétés (1)

Gestion du texte (pour n'importe quel sélecteur) :

- `color` : couleur du texte avec les valeurs :
  - `nom` : red, magenta ...
  - `RGB hexadécimal` : "#ff0000", #ff00ff" ...
  - `RGB` : "rgb(255, 0, 0)", "rgb(255, 0, 255)" ...
- `text-align` : positionnement du texte, valeurs :
  - left
  - right
  - center
  - justify
- `text-decoration` : décoration du texte, valeurs :
  - none : valeur par défaut
  - overline : surligné
  - line-through : barré
  - underline : souligné
  - blink : clignotant



**Texte:**

- Texte 1
- ~~Texte 2~~
- Texte 3
- Texte 4

# Feuille de style (CSS)

texte / propriétés (2)

Gestion du texte (pour n'importe quel sélecteur) :

- `text-transform` : modification du texte, valeurs :
  - `none`
  - `capitalize` : ajoute une majuscule au début du mot
  - `uppercase` : texte en majuscule
  - `lowercase` : texte en minuscule
- `text-indent` : indentation du texte, valeur en px, cm, % ...
- `letter-spacing` : espace entre les lettre, valeur en en px, cm ...
- `word-spacing` : espace entre les mots, valeur en en px, cm ...
- `direction` : sens du texte, 2 valeurs :
  - `rtl` : droite vers la gauche
  - `ltr` : gauche vers la droite

# Feuille de style (CSS)

police / propriétés (1)

Gestion de la police de caractère (pour n'importe quel sélecteur) :

- `font-style` : style de la police, valeurs :
  - `normal` : par défaut
  - `italic` : texte en italique
  - `oblique` : proche de l'italique
- `font-variant` : variante de la police, valeurs :
  - `normal` : par défaut
  - `small-caps` : petite majuscule
- `font-weight` : définit l'épaisseur de la police de caractère, valeurs :
  - `normal` : défaut
  - `bold` : épais
  - `bolder` : plus épais
  - `lighter` : moins épais
  - 100, 200, 300, 400 ( `normal` ), 500, 600, 700 ( `bold` ), 800, 900

# Feuille de style (CSS)

police / propriétés (2)

Gestion de la police de caractère (pour n'importe quel sélecteur) :

- `font-size` : taille de la police, valeurs :

- `xx-small`, `x-small`, `small`, `medium` (défaut), `large`, `x-large`, `xx-large`, `smaller` ou `larger`

- en `px`, `cm` ...

- en `%`

- `font-family` : type de police de caractère, plusieurs valeurs peuvent être spécifier en les séparant par une virgule :

`"times"`, `"monospace"`, `"cursive"`, `"courier"`, `"arial"`...

La propriété `font` permet de contrôler en respectant l'ordre suivant :

`font-style`, `font-variant`, `font-weight`, `font-size`, `font-family`

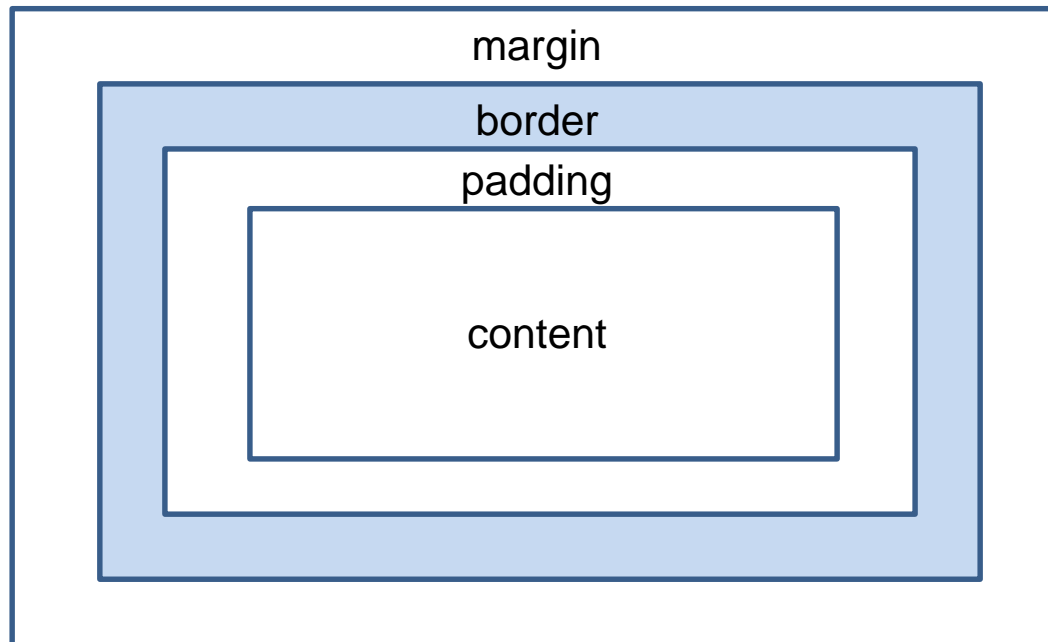
Exemple : `p{font:italic bold normal 20px arial,sans-serif;}`

# Feuille de style (CSS)

disposition des blocs

La disposition des structures ou blocs de la page web (ex: menu) sont contrôlées :

- via, préférentiellement, la balise `<div>`
- par les propriétés :
  - `margin` : zone transparente entre le "border" et les autres éléments ou la limite de la page web.
  - `border` : bordure.
  - `padding` : zone entre le contenu et le "border".
  - `content` : contenu.



# Feuille de style (CSS)

disposition des blocs / `margin`

Contrôle des marges par les propriétés :

- `margin-bottom` : marge inférieure
- `margin-left` : marge de gauche
- `margin-right` : marge de droite
- `margin-top` : marge supérieure

Les valeurs peuvent être négatives pour superposer des blocs et sont exprimées en :

- `px`, `cm` ...
- `%`

Les marges peuvent être gérées avec la seule propriété `margin` :

- avec 4 valeurs : `top right bottom left`
- avec 3 valeurs : `top (right/left) bottom`
- avec 2 valeurs : `(top/bottom) (right/left)`
- avec 1 valeur : `(top/bottom/right/left)`

```
p
{
margin:25px 50px 75px 100px;
}
```

# Feuille de style (CSS)

disposition des blocs / `margin`

Contrôle automatique des marges horizontal :

`auto` : avec une largeur définie, via `width`, centre le bloc en répartissant équitablement les marges de part et d'autre.

```
<html>
<head>
<style>
div {
    width : 250px;
    margin: auto ;
}
</style>
</head>

<body>
<div> Exemple de bloc centré </div>
</body>
</html>
```



# Feuille de style (CSS)

disposition des blocs / border (1)

Contrôle des bordures par les propriétés :

- `border-style` : style des bordures, valeurs :

`none`, `hidden`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset` ou `outset`

- `border-color` : couleur des bordures (nom, hexadécimal, RGB)

- `border-width` : taille de la bordure, valeur :

- `px`

- `thin` (fin), `medium` ou `thick` (épais)

On peut également contrôler les bordures individuellement :

`border-top-style`

`border-top-color`

`border-top-width`

`border-right-style`

`border-right-color`

`border-right-width`

`border-bottom-style`

`border-bottom-color`

`border-bottom-width`

`border-left-style`

`border-left-color`

`border-left-width`

# Feuille de style (CSS)

disposition des blocs / border (2)

On peut gérer l'épaisseur des différentes bordures avec une seule propriété :

- avec 4 valeurs : top right bottom left
- avec 3 valeurs : top (right/left) bottom
- avec 2 valeurs : (top/bottom) (right/left)
- avec 1 valeur : (top/bottom/right/left)

Exemple :

```
p
{
border-width: thick thin thin;
}
```

On peut également gérer l'ensemble des bordures avec la seule propriété `border`.

Exemple :

```
p
{
border:solid thick red;
}
```

# Feuille de style (CSS)

disposition des blocs / padding

On peut gérer l'espace entre le contenu et les bordures avec les propriétés suivantes:

- padding-top
  - padding-right
  - padding-bottom
  - padding-left
- en px, pt, cm ...
  - %

La seule propriété padding peut suffire en respectant l'ordre padding-top, padding-right, padding-bottom et padding-left :

- avec 4 valeurs : top right bottom left
- avec 3 valeurs : top (right/left) bottom
- avec 2 valeurs : (top/bottom) (right/left)
- avec 1 valeur : (top/bottom/right/left)

Exemple :

```
p
{
padding:15pt 15pt 25pt 10pt
}
```

# Feuille de style (CSS)

## liste (1)

Il est possible d'utiliser des listes supplémentaires par rapport à l'HTML avec les propriétés :

- `list-style-type` : type de liste pouvant avoir les valeurs suivantes :

`none`, `disc`, `circle`, `square`, `decimal`, `decimal-leading-zero`, `armenian`, `georgian`, `lower-alpha`, `upper-alpha`, `lower-greek`, `lower-latin`, `upper-latin`, `lower-roman` et `upper-roman`.

/!\ IE ne reconnaît pas toutes ces propriétés.

- `list-style-position` : contrôle l'indentation de la liste, valeurs :

- `inside` : pas de retrait du texte

- `outside` : retrait du texte (défaut)

Exemple :

### Valeur inside :

1. Le bon sens est la chose du monde la mieux partagée ... En quoi il n'est pas vraisemblable que tous se trompent ...
2. texte

### Valeur outside :

1. Le bon sens est la chose du monde la mieux partagée ...  
En quoi il n'est pas vraisemblable que tous se trompent  
...
2. texte

# Feuille de style (CSS)

## liste (2)

Il est possible d'utiliser des listes supplémentaires par rapport à l'HTML avec les propriétés :

- `list-style-image` : remplace les puces pré-définies, valeurs:
  - `none`
  - `url("XXX.jpg")`

La propriété `list-style` permet de gérer entièrement une liste, à condition de respecter l'ordre : `list-style-type`, `list-style-position` et `list-style-image`.

Exemple :

```
ol
{
list-style:none inside url("../puce_1.jpg");
}
```

# Feuille de style (CSS)

## tableau (1)

On peut définir plusieurs propriétés pour contrôler un tableau :

- `table-layout` : agencements des cellules , valeurs:
  - `auto` : la largeur des cellules s'adapte à son contenu (défaut).
  - `fixed` : la largeur des cellules est strictement limitée (attribut `width` de la balise `<td>`).

Exemple :

```
table.tab1 {table-layout:auto}
table.tab2 {table-layout:fixed}
```

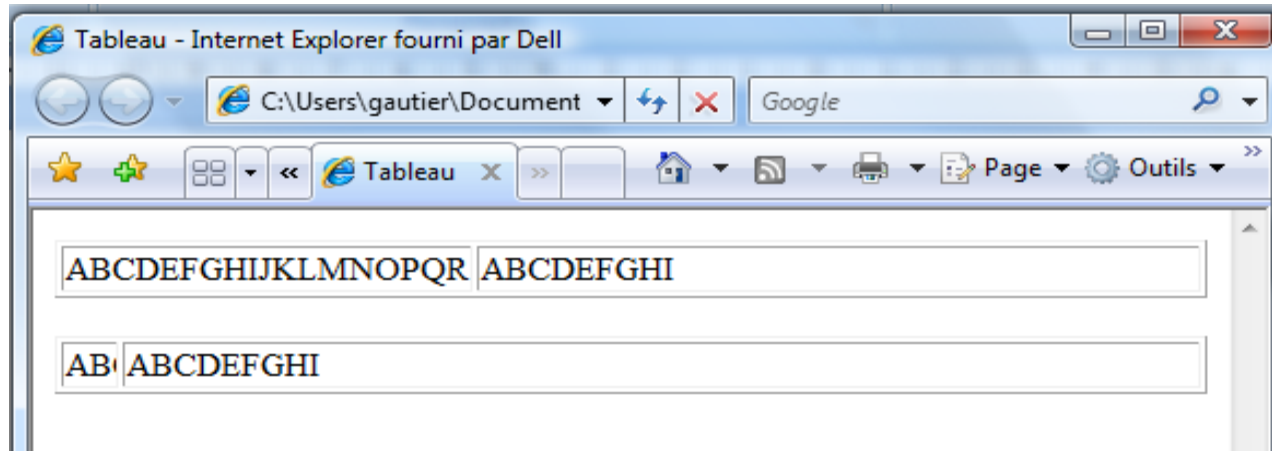
```
<table class="tab1" border="1" width="100%">
<tr>
<td width="5%">ABCDEFGHIJKLMNOPQR</td>
<td width="95%">ABCDEFGHI</td>
</tr>
</table>
```

```
<table class="tab2" border="1" width="100%">
<tr>
<td width="5%">ABCDEFGHIJKLMNOPQR</td>
<td width="95%">ABCDEFGHI</td>
</tr>
</table>
```

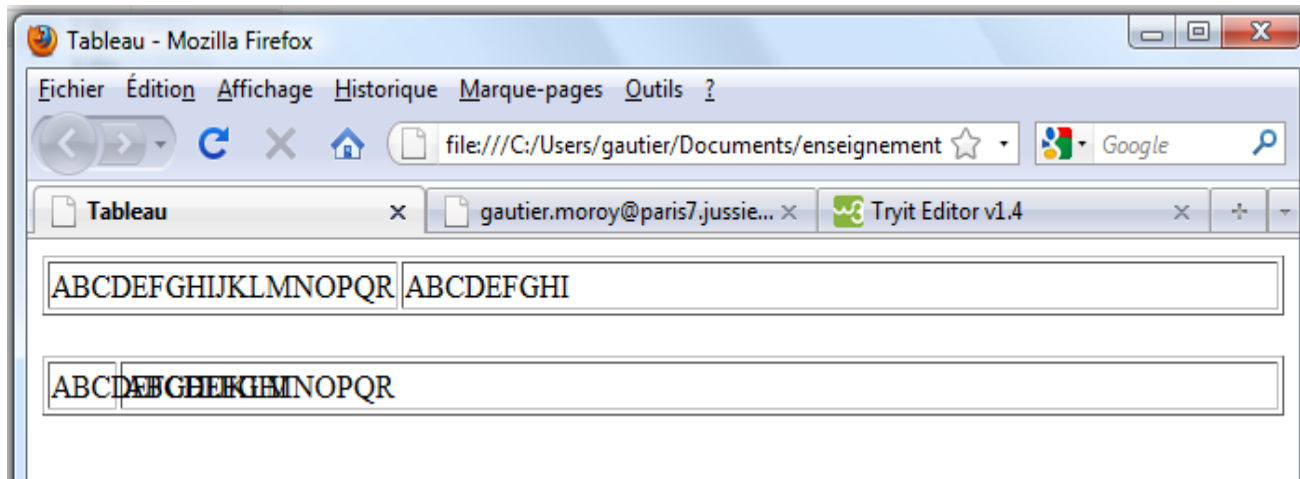
# Feuille de style (CSS)

tableau (2)

IE



Firefox



# Feuille de style (CSS)

## tableau (3)

On peut définir plusieurs propriétés pour contrôler un tableau :

- `empty-cells` : cellules vides, valeurs:
  - `hide` : les traits autour des cellules vides n'apparaissent pas
  - `show` : les cellules vides sont visibles (défaut).
  
- `caption-side` : position de la légende définie avec `<caption>`, valeurs :
  - `top` : au-dessus du tableau (défaut).
  - `bottom` : en-dessous du tableau.
  
- `border-spacing` : espace entre les cellules et les autres éléments du tableau, valeur en px, pt ... :
  - 1 valeur : valable pour les espaces verticaux et horizontaux.
  - 2 valeurs : définit l'espace vertical puis l'espace horizontal.
  
- `border-collapse` :
  - `collapse` : les bordures entre les cellules sont fondues .
  - `separate` : les bordures sont séparées (défaut).



# Feuille de style (CSS)

## dimensions

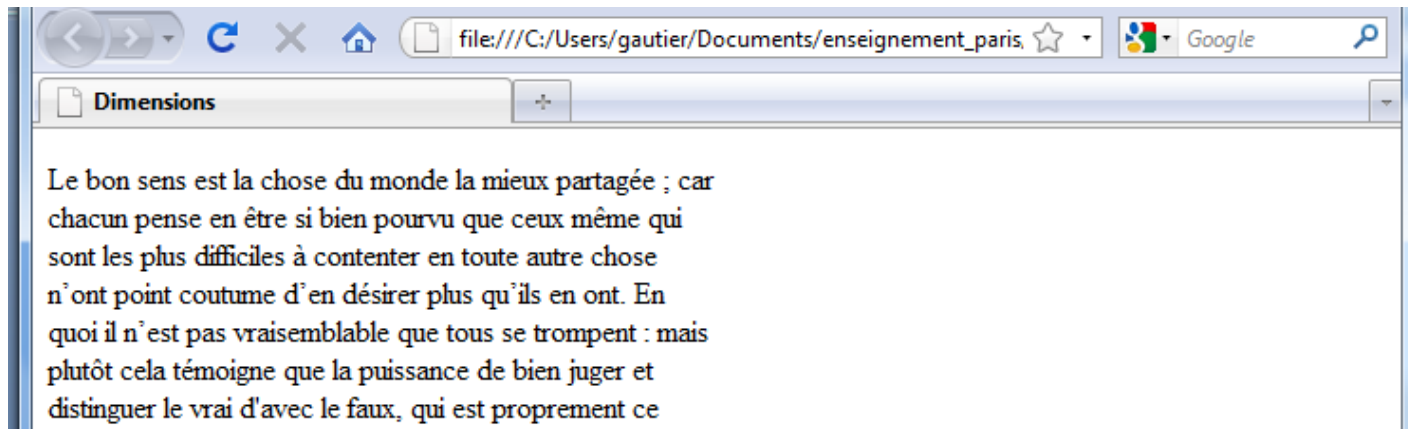
Les dimensions d'un élément peuvent être gérées par les propriétés suivantes :

- `height` : hauteur de l'élément
- `max-height` : hauteur maximale de l'élément
- `min-height` : hauteur minimale de l'élément
- `width` : largeur de l'élément
- `max-width` : largeur maximale de l'élément
- `min-width` : largeur minimale de l'élément

Les valeurs de ces propriétés peuvent être en px, pt, em, % ou auto (défaut, calculer par le navigateur).

Exemple :

```
p { max-width:50%; }
```



# Feuille de style (CSS)

## positionnement (1)

Le positionnement d'un élément se fait, via la propriété `position`, et peut être :

- absolu (`absolute`) : a pour origine le coin supérieur droit
- relatif (`relative`) : par rapport à sa position normale

Les propriétés `top`, `right`, `bottom` et `left` spécifient le positionnement en `px`, `pt` ... ou % de la page.

Exemple :

```
.absolu
{
position:absolute;
top: 70px;
left: 80px;
}

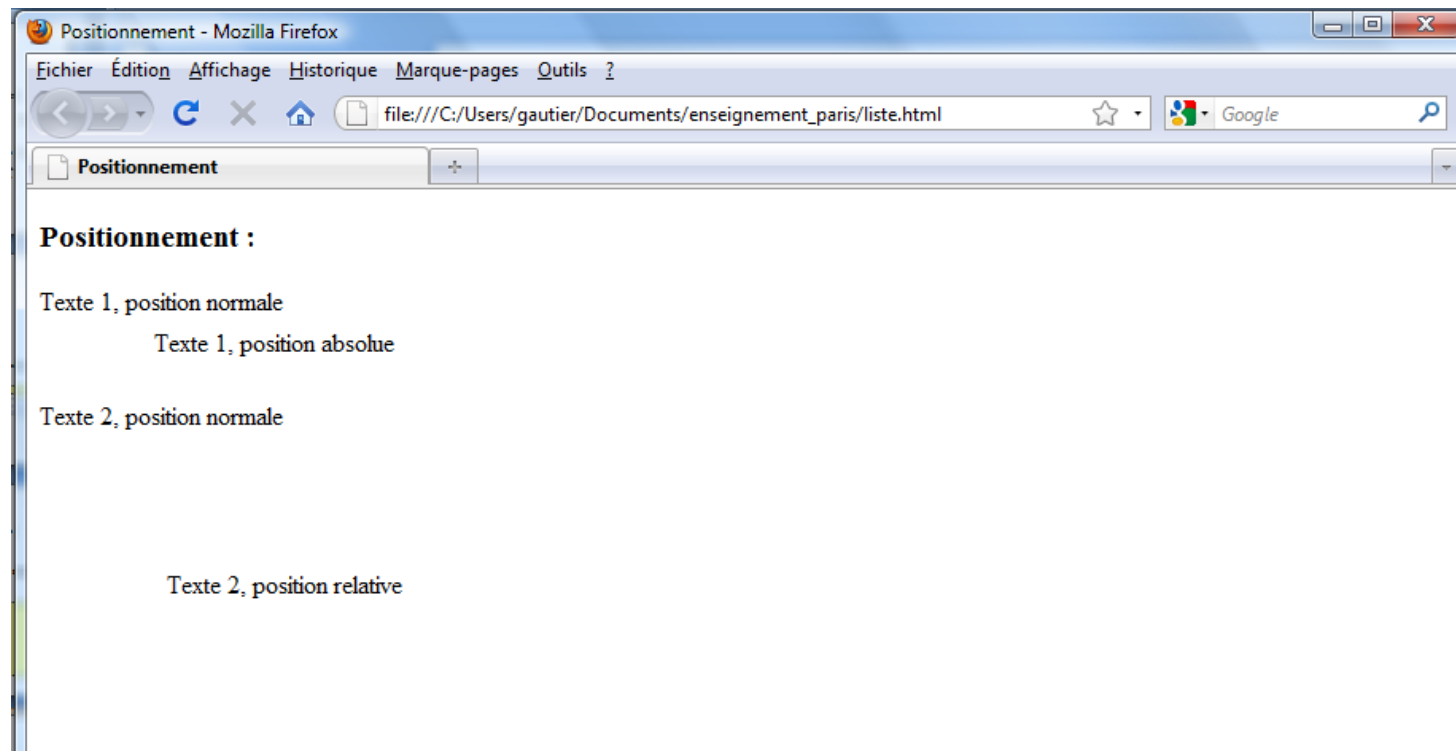
.relative
{
position:relative;
top: 70px;
left: 80px;
}
```

# Feuille de style (CSS)

## positionnement (2)

```
<h3> Positionnement :</h3>

<p> Texte 1, position normale</p>
<p class="absolu"> Texte 1, position absolue</p>
<br />
<p> Texte 2, position normale</p>
<p class="relative"> Texte 2, position relative </p>
```



# Feuille de style (CSS)

## liens (1)

Les liens peuvent être mis en page avec n'importe quelles propriétés CSS (`color`, `text-align`, `font-family` ...).

Il existe 4 sélecteurs sont spécifiques aux ancrages :

- `a:link` : lien non visité
- `a:visited` : lien déjà visité
- `a:hover` : lien lorsque la souris est au-dessus
- `a:active` : lien actif

Exemple :

```
a:link {color:red ; text-decoration:none}  
  
a:visited {color: green }
```

# **Programmation WEB : XHTML**

# XHTML

## généralités

XHTML (**EX**tensible **H**yper **T**ext **M**arkup **L**anguage) :

- combinaison entre HTML et XML.
- version stricte et "propre" du HTML.

XML (**EX**tensive **M**arkup **L**anguage) :

- langage à balise, complémentaire du HTML.
- conçu pour décrire, transporter et stocker les données.
- syntaxe stricte

But :

- permettre au créateur de site web d'introduire de nouvelles balises.
- volonté d'interopérabilité entre les différentes plateformes d'accès à internet.

→ utilisation du XML pour décrire les données et l'HTML pour les afficher.

→ besoin d'homogénéiser la syntaxe des 2 langages, en se basant sur celui du XML.

→ éviter l'utilisation d'HTML "sale".

# XHTML

## généralités

Exemple à proscrire, même si le navigateur l'affiche "correctement" :

```
<html>
<head>
<title> HTML "sale" </title>
<body>

<h1> HTML non conforme : </h1>

  <ul>
    <li> absence de balise de fermeture
  </ul>
</body>
```

# XHTML

## syntaxe

Principes à respecter :

- les balises doivent être correctement imbriquées :

```
<ul>  
<li> <b> Texte </b> </li>  
</ul>
```

- les balises doivent toujours être fermées.

dans le cas où il n'existe pas de balises de fermeture, il faut symboliser la fermeture en ajoutant "/" avant la fin de la balise :

```
<br /> ; <hr /> ; <img /> ; <input /> ...
```

- les balises et les attributs doivent être écrits en minuscule.

- les valeurs des attributs doivent être bordés par des apostrophes ou des guillemets.

- les minimisations d'attributs sont interdits :

Exemple : `<input checked>` → `<input checked="checked" />`



# XHTML

syntaxe

Principes à respecter :

- toujours débiter avec la déclaration du type de document en présence pour que le navigateur puisse l'interpréter correctement :

DOCTYPE ("DOCument TYPE")

Exemple :

```
<!DOCTYPE ...>  
<html>  
<head> ... </head>  
  
<body> ... </body>  
</html>
```

# XHTML

## DOCTYPE

syntaxe :

1: `<!DOCTYPE`

2 : Définition du Type de Document (DTD) : `html`

3: identification publique à l'origine de la DTD : `W3C`

4 : identité de la DTD : par exemple `DTD XHTML 1.0 Strict`

5 : langage dans lequel est écrit la DTD : `EN`

6 : identifiant signal : adresse du document :

`http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd`

# XHTML

## DOCTYPE

3 catégories de DTD possibles en HTML:

- strict :

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

- transitionnel : + utilisé

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

- frameset : si on utilise le système de frame

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

# XHTML

## validité

Le consortium W3C met à disposition un validateur de XHTML à l'adresse :

<http://validator.w3.org/>

Pour que la page web soit complètement validé, il faut :

- modifier la balise `<html>` de la manière suivante pour respecter les normes XML :

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr" xml:lang="fr">
```

- spécifier dans l'entête le type de contenu et le l'encodage des caractères :

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

Il existe également un validateur de CSS :

<http://jigsaw.w3.org/css-validator/>

# **Programmation WEB : HTML5**

# HTML 5

## généralités

- Issu d'une volonté de mettre à jour le langage HTML 4.01 datant de 1999.
  - Doit remplacer le HTML 4 et le XHTML.
  - Est conçu pour ne pas devoir utiliser des plugins externes (ex : Flash).
  - Introduit de nouvelles fonctionnalités pour gérer plus facilement le contenu multimédia et graphique (but de remplacer le scripting par du balisage).
  - volonté d'être multi-plateforme (tablette, Smartphone, ordinateur ...).
  - est toujours en développement, mais le W3C conseille son utilisation dès à présent.
- Tous les navigateurs prennent correctement en charge l'HTML 5.

# HTML 5

## structure générale

La déclaration du type document se fait simplement :

```
<!DOCTYPE html>
```

La structure minimale est :

```
<!DOCTYPE html>
<html>

<head>
<meta charset="UTF-8">
<title> Titre du document </title>
</head>

<body>
Contenu du document
</body>

</html>
```

# HTML 5

## nouveaux éléments sémantiques

**Élément sémantiques** : éléments possédant une signification "claire" du contenu

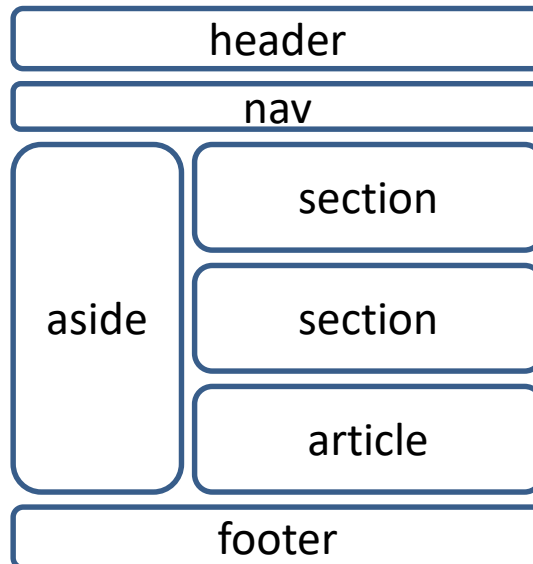
Exemples : `<img>`, `<table>` ...

Exemples d'éléments non sémantiques : `<div>`, `<span>` ...

**Nouveaux éléments sémantiques :**

→ évite d'utiliser des calques : `<div id="nav">`, `<div class="header">` ou `<div id="footer">` ...

→ définit clairement différentes parties d'une page web.





# HTML 5

## nouveaux éléments sémantiques

`<header>` : spécifie l'entête d'une section ou d'un document, doit être utilisé comme une introduction.

`<footer>` : spécifie le pied de page d'une section ou d'un document, doit contenir des informations des éléments contenus (auteur, copyright, contact ...).

`<nav>` : définit les liens pour la navigation, doit contenir les liens les plus importants et pas tous les liens du document.

`<aside>` : définit un contenu à part d'une section ou d'un document mais qui est en rapport avec la section ou le document, par exemple une barre latérale.

`<section>` : définit un groupement thématique de contenus, avec typiquement un entête.

`<article>` : précise un contenu autonome, peut être distribué indépendamment du reste du site web (ex: un post de blog ou de forum).

# HTML 5

## nouveaux éléments sémantiques

`<figure>` : spécifie un contenu autonome tel qu'une figure, illustration ...

`<figcaption>` : définit une légende à l'élément `<figure>`.

```
<figure>

<figcaption>Figure 1 : logo de Paris Diderot</figcaption>
</figure>
```

# HTML 5

## nouveaux éléments sémantiques

`<mark>` : spécifie du texte à mettre en évidence.

`<time>` : spécifie une heure ou une date.

`<progress>` : représente une progression grâce à une jauge.

```
<h2> Progression du téléchargement </h2>  
  
<progress value="72" max="100"></progress>
```

/!\ Il existe d'autres nouveaux éléments sémantiques, mais qui ne sont pour l'instant pas acceptés par tous les principaux navigateurs :

`<details>`

`<summary>`

`<bdi>`

`<wbr>`

`<dialog>`

`<command>`

`<meter>`

# HTML 5

balises obsolètes

<acronym>  
<applet>  
<basefont>  
<big>  
<center>  
<dir>  
<font>  
<frame>  
<frameset>  
<noframes>  
<strike>  
<tt>

# HTML 5

media : vidéo

Avant le HTML 5, il n'y avait pas norme standard pour gérer l'affichage de vidéo ou de film dans des pages web.

→ `<video>`

La balise `<source>` permet de contrôler le contenu, 2 attributs principaux :

- `src` : donne la localisation du fichier video.
- `type` : précise le type de format.

3 formats sont supportés : MP4, WebM et Ogg

```
<video>

<source src="./matrix.mp4" type="video/mp4">
<source src="./matrix.webm" type="video/webm">
<source src="./matrix.ogg" type="video/ogg">

</video>
```

# HTML 5

media : vidéo

Voici les attributs disponible pour la balise `<video>` :

- `autoplay` : débute la lecture de la vidéo au chargement de la page.
- `controls` : permet d'afficher des boutons pour contrôler la lecture de la vidéo.
- `loop` : recommence la lecture à la fin de la vidéo.
- `muted` : désactive le son de la vidéo.
- `poster` : spécifie l'image qui apparaît pendant le téléchargement de la vidéo ou tant que l'utilisateur n'a pas débuté la lecture.
- `src` : donne l'URL de la vidéo
- `height` : hauteur demandée en pixel.
- `width` ; largeur demandée en pixel.

# HTML 5

media : audio

Avant le HTML 5, il n'y avait pas norme standard pour gérer des fichiers audio dans des pages web.

→ `<audio>`

La balise `<source>` permet de contrôler le contenu, 2 attributs principaux :

- `src` : donne la localisation du fichier video.

- `type` : précise le type de format.

3 formats sont supportés : MP3, Wav et Ogg

```
<audio>
```

```
<source src="./johnny.mp3" type="audio/mpeg">
```

```
<source src="./johnny.wav" type="audio/wav">
```

```
<source src="./johnny.ogg" type="audio/ogg">
```

```
</audio>
```

# HTML 5

media : audio

Voici les attributs disponible pour la balise `<audio>` :

- `autoplay` : débute la lecture du fichier audio au chargement de la page.
- `controls` : permet d'afficher des boutons pour contrôler la lecture.
- `loop` : recommence la lecture à la fin du fichier audio.
- `muted` : désactive le son du fichier audio.
- `src` : donne l'URL de la vidéo



# HTML 5

## graphisme

L'HTML 5 utilise le langage **SVG** (Scalable Vector Graphics) pour créer des graphiques. Les graphiques sont vectoriels (pas de perte de qualité lors de zoom ou modification de taille).

Le SVG est basé sur le langage XML (eXtensible Markup Language).

Le SVG permet de créer des graphiques en 2D.

On définit la zone du graphique grâce à la balise `<svg>`

Elle a 2 attributs :

- `height` : hauteur de la zone
- `width` : largeur de la zone

```
<svg width="200" height="400">  
  
</svg>
```

# HTML 5

## graphisme

### Formes disponibles :

- rectangle : `<rect>`, attributs :
  - height : hauteur
  - width : largeur
  - x : coordonnée horizontale du coin supérieur gauche
  - y : coordonnée verticale du coin supérieur gauche
  - rx : rayon horizontal pour arrondir les coins
  - ry : rayon vertical pour arrondir les coins

```
<svg width="200" height="400">  
  
<rect width="100" height="200" x="10" y="20" />  
  
</svg>
```

# HTML 5

## graphisme

### Formes disponibles :

- cercle : `<circle>`, attributs :
  - `r` : rayon
  - `cx` : coordonnée horizontale du centre du cercle
  - `cy` : coordonnée verticale du centre du cercle

```
<svg width="200" height="400">  
  
<circle cx="100" cy="200" r="100" />  
  
</svg>
```

# HTML 5

## graphisme

### Formes disponibles :

- ellipse : `<ellipse>`, attributs :
  - rx : rayon horizontal
  - ry : rayon vertical
  - cx : coordonnée horizontale du centre de l'ellipse
  - cy : coordonnée verticale du centre de l'ellipse

```
<svg width="200" height="400">  
  
<ellipse cx="100" cy="200" rx="100" ry="50" />  
  
</svg>
```

# HTML 5

## graphisme

### Formes disponibles :

- ligne : `<line>`, attributs :
  - `x1` : coordonnée horizontale du point de départ
  - `y1` : coordonnée verticale du point de départ
  - `x2` : coordonnée horizontale du point final
  - `y2` : coordonnée verticale du point final

```
<svg width="200" height="400">  
  
<line x1="0" y1="0" x2="200" y2="400" stroke="red" />  
  
</svg>
```

# HTML 5

## graphisme

### Formes disponibles :

- polygone : `<polygon>`, attribut :
  - `points` : coordonnées horizontales et verticales de chaque point

```
<svg width="200" height="400">  
  
<polygon points="0,0 60,30 40,160"    />  
  
</svg>
```

# HTML 5

## graphisme

### Formes disponibles :

- ligne brisée : `<polyline>`, attribut :
  - `points` : coordonnées horizontales et verticales de chaque point

```
<svg width="200" height="400">  
<polyline points="0,0 60,30 40,160 100,400" stroke="red" fill="white" />  
</svg>
```

- texte : `<text>`, attributs :
  - `x` : coordonnée horizontale du texte
  - `y` : coordonnée verticale du texte

# HTML 5

## graphisme

### Formes disponibles :

- attributs non-spécifiques :
  - fill : couleur du contenu
  - fill-opacity : opacité de la couleur du contenu (en %)
  - stroke : couleur du trait
  - stroke-width : épaisseur du trait (en pixel)
  - stroke-dasharray : création de ligne pointillé
  - stroke-opacity : opacité de la couleur du trait (en %)

```
<svg width="200" height="400">  
  
<polyline points="0,0 60,30 40,160 100,400" stroke="red" stroke-width="5"  
fill="white" stroke-dasharray="10 10 5 5"/>  
  
</svg>
```

