

# Web mining

## Une vue d'ensemble des technologies Web

26 Septembre 2011





## Généralités

### Les champs de saisie

CSS

Langages côté serveur

JavaScript

Outils et Références





## Généralités

Les champs de saisie

CSS

Langages côté serveur

JavaScript

Outils et Références





- Permettent d'interagir avec l'utilisateur en lui proposant d'entrer des informations
- En HTML : uniquement l'interface de formulaire
- L'essentiel du travail sera fait par le **script** qui traitera la soumission du formulaire





Un formulaire HTML est placé à l'intérieur d'une balise `<form>` .

- Celle-ci prend les attributs suivants :

**action** URL du script auquel sera soumis le formulaire.

**method** Méthode HTTP, valant soit `"get"` soit `"post"` .

**enctype** Encodage HTTP. Peut valoir `"application/x-www-form-urlencoded"` (valeur par défaut) ou `"multipart/form-data"` .

## Exemple (Formulaire élémentaire)

```
<form action="action.php" method="get">  
  <div><input type="submit"></div>  
</form>
```



En HTML, il est interdit de mettre des champs de formulaire

directement à l'intérieur d'un `<form>`. Il faut d'abord les regrouper :

- Dans des paragraphes `<p>` si les champs de formulaires sont à l'intérieur de paragraphes de textes (rare).
- Dans des ensembles de champ `<fieldset>` pour regrouper des champs de formulaire de sémantique proche. On pourra alors donner une légende à l'ensemble de champs avec la balise `<legend>`.
- Dans des divisions `<div>` sans contenu sémantique sinon.

## Exemple (Ensemble de champ)

```
<fieldset>
  <legend>Mensurations</legend>
  <input type="text" name="taille">
  <input type="text" name="poids">
</fieldset>
```





Tout part des champs sont naturellement accompagnés d'une **étiquette** ( `<label>` ).

- On peut la placer où on veut, en général juste à gauche ou à droite du champ.
- Son attribut `for` référence l'attribut `id` du champ correspondant.
- Dans les navigateurs graphiques, un clic sur l'étiquette d'un champ permet en général de sélectionner le champ.

## Exemple (Étiquette)

```
<label for="taille">Taille :</label>  
<input type="text" name="taille" id="taille">
```





CSS

Langages côté serveur

JavaScript

Outils et Références





La balise `<input>` a une utilisation très vaste dans les formulaires. Elle représente un champ de saisie.

- L'attribut `type` détermine le type (texte, mot de passe, liste, etc.) du champ.
- L'attribut `name` (nom du paramètre de la requête HTTP) est **obligatoire** (sauf pour les types `"reset"` et `"submit"`); il permet de préciser au serveur à quelle saisie on fait référence.

## Exemple (Zone de texte pour écrire un commentaire)

```
<input type="text" name="Commentaire">
```





- `type="text"` est utilisé pour la saisie d'un texte dont la taille est inférieure à une ligne.
- L'attribut `value` permet de préciser la valeur par défaut (facultatif).
- La taille maximale de la chaîne de caractères à saisir peut être spécifiée à l'aide de l'attribut `maxlength` (facultatif).

## Exemple

```
<input type="text" name="prenom" value="Jordy" maxlength="50">
```





`type="password"` est utilisé pour la saisie d'un texte dont les caractères sont remplacés par des astérisques : c'est généralement utilisé pour la saisie des mots de passe. Le mot de passe est quand même transmis en clair au serveur !

- L'attribut `value` permet de préciser la valeur par défaut (facultatif).
- La taille maximale de la chaîne de caractères à saisir peut être spécifiée à l'aide de l'attribut `maxlength` (facultatif).

## Exemple

```
<input type="password" name="pwd" value="12345678">
```



- `type = "checkbox"` permet de choisir plusieurs éléments parmi un grand nombre de possibilités.
- Cela se matérialise sous forme de cases à cocher.
- La valeur retournée est **obligatoirement** précisée à l'aide de l'attribut `value`.
- L'attribut `checked = "checked"` permet de cocher la case par défaut.
- Certains langages côté serveurs imposent que les noms de champs de formulaire à choix multiples se terminent par [].

## Exemple

```
<input type="checkbox" name="pub[]" value="site"
  checked="checked" id="pub-site">
<label for="pub-site">Recevoir des offres de notre site</label>

<input type="checkbox" name="pub[]" checked="checked"
  value="externe" id="pub-externe">
<label for="pub-externe">Recevoir des offres externes</label>
```

- `type="radio"` permet de choisir un seul élément parmi une liste de possibilités.
- Cela se matérialise sous forme de boutons radio.
- La valeur retournée est **obligatoirement** précisée à l'aide de l'attribut `value`.
- L'attribut `checked="checked"` permet de préciser la valeur par défaut.

## Exemple

Recevoir de la pub:

```
<input type="radio" name="pub" value="oui" id="pub-oui"
  checked="checked">
```

```
<label for="pub-oui">oui</label>
```

```
<input type="radio" name="pub" value="non" id="pub-non">
```

```
<label for="pub-non">non</label>
```





- `type="file"` permet de joindre au formulaire un fichier.
- À cause de la taille de la requête due au téléchargement (**upload**) du fichier, il faut impérativement utiliser la méthode POST et l'encodage `multipart/form-data`.

## Exemple

```
<label for="fichier">Fichier:</label>  
<input type="file" name="fichier" id="fichier">
```





`="hidden"` permet de cacher des champs au client mais leur contenu est envoyé avec le formulaire.

- Ceci permet de préciser des informations, en utilisant l'attribut `value`, concernant l'interaction client/serveur.
- C'est à utiliser **avec précaution** car cela peut être à l'origine de problèmes de sécurité assez graves : ne pas oublier que le client peut éditer la page à la main pour changer la valeur de ces champs !

## Exemple

```
<input type="hidden" name="monnaie_utilisee" value="EUR">  
<input type="hidden" name="customerCB" value="c2415-345-8563">
```





- `type="reset"` permet de réinitialiser le formulaire en affectant aux différents champs leur valeur par défaut.
- L'attribut `value` permet de changer le texte du bouton correspondant.

## Exemple

```
<input type="reset" value="Tout effacer">
```





- `type = "submit"` permet de soumettre le formulaire.
- Le client envoie le contenu du formulaire à l'adresse précisée par l'attribut `action` de la balise `<form>`.
- L'attribut `value` permet de changer le texte du bouton correspondant.

## Exemple

```
<input type="submit" value="Envoyer">
```





Pour les saisies multiligne, on utilise la balise `<textarea>`.

- Le texte délimité par cette balise permet d'initialiser la valeur par défaut du champ.
- La balise fermante est **obligatoire** même si le champ est vide.
- Les attributs `rows` et `cols` (obligatoires) permettent de spécifier la taille en lignes et colonnes de la fenêtre de saisie.

## Exemple

```
<textarea name="bio" cols="40" rows="5">  
Fille de Josiane Balasko,  
Marilou Berry fait ses premiers pas au cinéma à 8 ans...  
</textarea>
```



- La balise `<select>` permet d'ajouter une liste de sélection :

L'attribut facultatif `size` permet de préciser le nombre de choix apparaissant sur la page Web. Par défaut, ce nombre est initialisé à 1.

- L'attribut `multiple = "multiple"` permet d'autoriser des réponses multiples. Dans ce cas, pour PHP, on donnera toujours un nom se terminant par `[]`.
- Les choix du menu sont indiqués à l'aide de la balise `<option>` :
  - L'attribut `selected = "selected"` permet de spécifier le(s) choix par défaut.
  - L'attribut `value` permet de spécifier la valeur associée au choix.

## Exemple

```
<select name="age">
  <option value="20">Moins de 20 ans</option>
  <option value="35" selected="selected">21 à 35 ans</option>
  <option value="50">36 à 50 ans</option>
  <option value="51">Plus de 51 ans</option>
</select>
```



## CSS

CSS et HTML

Sélecteurs de CSS

Mise en forme

Mise en page

Fonctionnalités mal supportées

Langages côté serveur

JavaScript

Outils et Références



## CSS

CSS et HTML

Sélecteurs de CSS

Mise en forme

Mise en page

Fonctionnalités mal supportées

Langages côté serveur

JavaScript

Outils et Références



# CSS : Cascading StyleSheets

## Recommandation du W3C

- Plusieurs niveaux : CSS1 (1996), CSS2 (1998), CSS2.1 (2005), CSS3 (en cours)
- Support par les navigateurs très inégal, jamais complet (en particulier, IE6 et IE7 ont plusieurs limitations importantes) ⇒ on se concentrera sur ce qui marche bien dans l'ensemble des navigateurs courants.

## Principe

- HTML décrit la **structure** et le **contenu**
- CSS décrit :
  - la **mise en forme** du texte
  - la **mise en page** des **boîtes** les unes par rapport aux autres



- Manière la plus simple d'utiliser les CSS.

Rajouter un attribut `style` sur les balises HTML.

- On peut utiliser `<span>` si on a besoin d'une balise supplémentaire.
- Encombre le code HTML avec des indications de mise en forme : ce n'est pas ce qu'on veut !

## Exemples

- Ce mot en `<em style="color: red;">emphase</em>` est aussi en rouge.

```
<span style="text-decoration: underline;">Cette expression de  
plusieurs mots</span> est soulignée.
```



- Intégration des propriétés de style à l'**en-tête** de la page avec la balise `<style>`
- Utilisation des **sélecteurs** pour définir à quels élément les propriétés s'appliquent
- Inconvénients : mélange HTML et CSS dans le même document, impossible de réutiliser les propriétés CSS dans plusieurs documents

## Exemple

```
<!DOCTYPE ... >
<html>
  <head> ...
    <style type="text/css">
      em { color: red; }
    </style>
  </head>
  <body> ... </body>
</html>
```



■ Mettre la feuille de style CSS dans un fichier à part (en général, on utilise l'extension .css).

- Permet d'utiliser la même feuille de style pour plusieurs pages Web.
- Rajouter une balise `<link>` dont l'attribut `rel` est positionné à "stylesheet" dans l'en-tête du document.
- Possibilité d'ajouter `media="screen"` ou `media="print"`, etc., pour choisir différentes feuilles de style suivant le mode d'affichage.

## Exemple

```
<!DOCTYPE ... >
<html>
  <head> ...
    <link rel="stylesheet" href="feuille.css" type="text/css">
  </head>
  <body> ... </body>
</html>
```



- On veut parfois rajouter encore plus de structure et de sémantique à un document HTML.
- On utilise l'attribut `class` sur n'importe quelle balise (ou, à défaut, sur une balise `<span>`).
- Après, on peut utiliser CSS pour appliquer une mise en forme commune à tout ce qui fait partie d'une `class` particulière.

## Exemple (Mettre en bleu italique les noms de personnes)

### ■ HTML

```
<p>Je voudrais remercier en particulier  
<span class="personne">Madame Machin</span>  
et <span class="personne">Monsieur Bidule</span>.</p>
```

### ■ CSS

```
.personne { color: blue; font-style: italic; }
```





Ensemble de règles de la forme :

```
sélecteur {  
  propriété: valeur;  
}
```

- **sélecteur** : indique à quelles parties du documents la règle s'applique
- **propriété** : propriété spécifique de mise en forme à modifier
- **valeur** : son sens dépend de la propriété.
- Les feuilles de style peuvent être **validées** avec un validateur approprié, cf. <http://jigsaw.w3.org/css-validator/>
- Commentaires entre /\* et \*/.





Plusieurs feuilles de style peuvent s'appliquer simultanément :

Plusieurs balises `<link rel="stylesheet">`

- Directive `@import` d'une feuille de style  
`@import url(feuille_annexe.css);`
- Feuille de style de l'utilisateur (Mozilla, Opera...)
- Au sein même d'une feuille de style, plusieurs règles peuvent être en conflit.
- **Cascade** : mécanismes gérant ces conflits.

## Cascade

- Si **!important** est précisé après la valeur, la règle sera prioritaire.
- Sinon, plus une règle est spécifique, plus elle est prioritaire.
- Sinon, la dernière règle s'applique.



## CSS

CSS et HTML

Sélecteurs de CSS

Mise en forme

Mise en page

Fonctionnalités mal supportées

Langages côté serveur

JavaScript

Outils et Références





**Sélecteur simple** : nom d'une balise.

**Sélecteur multiple** : plusieurs sélecteurs séparés par des virgules.

**Sélecteur universel** : '\*', sélectionne tout.

## Exemples

- `ul { color: blue; }` met l'ensemble du contenu des listes non ordonnées en bleu.
- `h1,h2,h3,h4,h5,h6 { color: red; }` met l'ensemble des titres de rubrique en rouge.
- `* { color: black; }` met tout en noir. Dans ce cas précis, on préférera une règle `body { color: black; }`.





**Sélecteur de classe** : nom d'une classe, préfixée d'un '.', tel qu'il apparaît dans un attribut `class` d'une balise HTML.

## Exemples

- `.personne { font-weight: bold; }` met l'ensemble des balises de classe `personne` en gras.
- `p.comment { font-style: italic; }` met l'ensemble des balises `<p>` de classe `comment` en italique.





**id** : défini par l'attribut `id` d'une balise HTML. Similaire aux classes, mais il ne peut y avoir qu'une seule balise ayant un `id` donné dans tout le document HTML.

**Sélecteur d'identifiant** : nom d'une classe, préfixée d'un '#', tel qu'il apparaît dans un attribut `id` d'une balise HTML.

## Exemples

- `#introduction { font-size: 120%; }` met la balise d'identifiant `introduction` en plus gros.
- `p#introduction { font-size: 120%; }` met la balise `<p>` d'identifiant `introduction` en plus gros.





**Sélecteur contextuel** : 2 sélecteurs ou plus séparés par des espaces.  
 $A B$  sélectionne les  $B$  seulement s'ils sont contenus dans des  $A$ .

## Exemples

- `h1 em { color: blue; }` met les mots en emphase à l'intérieur d'un h1 en bleu.
- `ul ol, ol ul, ul ul, ol ol { font-size: 80%; }` diminue la taille du texte des listes imbriquées.





**o-classes** : permet de faire une sélection d'un élément uniquement dans certains contextes.

- `a:link` : les `<a>` qui sont des liens.
- `a:visited` : les `<a>` qui sont des liens vers des pages qui ont déjà été visitées.
- `a:hover` : les `<a>` qui sont des liens qu'on est en train de désigner (i.e., la souris est en train de passer dessus).
- `a:active` : les `<a>` qui sont des liens qu'on est en train d'activer (i.e., de cliquer).

## Remarque

**Attention !** Toujours définir ces pseudo-classes dans ce sens, à cause des règles de cascade. (**Las Vegas (Forest) Has Animals**).



## CSS

CSS et HTML

Sélecteurs de CSS

**Mise en forme**

Mise en page

Fonctionnalités mal supportées

Langages côté serveur

JavaScript

Outils et Références



Les propriétés admettent différentes unités de mesure se répartissant  
suivant trois types de valeurs :

- sous forme de pourcentage (par rapport à la valeur courante)
- les valeurs relatives :
  - em** la valeur de la propriété `font-size` de la police utilisée (hauteur du bloc dans lequel s'inscrit naturellement un caractère)
  - ex** la hauteur du caractère `x` dans la police utilisée
- les valeurs relatives à l'écran (**à éviter pour avoir une conception de la page indépendante de la résolution !**)
  - px** le nombre de pixels
- les valeurs absolues (**N'ont aucun sens pour une page Web destinée à s'afficher sur un écran d'ordinateur !**) : mm, cm, in, pt, pc





Il existe différents moyens pour contrôler les polices de caractères.

- Les principales propriétés (`font-family`, `font-size`, `font-weight` et `line-height`) sont responsables de l'aspect du texte.

## Exemple

```
p {  
  font-family: "Times New Roman";  
  font-size: 130%;  
  font-weight: bold;  
  line-height: 150%;  
}
```





`text-align` définit la justification et l'alignement d'un texte : `left`, `right`, `center` et `justify` (aligne le texte sur les marges gauche et droite).

`vertical-align` définit l'alignement vertical de la ligne de base d'un élément en fonction de la propriété `line-height` : `super` (texte en exposant), `sub` (texte en indice), `baseline` (normal), etc.

`text-indent` permet de décaler le début de la première ligne de texte d'une valeur fixe ou proportionnelle à la valeur du paragraphe.





- La couleur doit être utilisée avec **discernement**.
- Son rôle est de différencier les éléments d'un texte en jouant sur la notion de contraste.
- Il existe différentes manières de spécifier les valeurs d'une couleur :
  - En indiquant le nom d'une couleur prédéfinie parmi `aqua`, `black`, `blue`, `fuchsia`, `gray`, `green`, `lime`, `maroon`, `navy`, `olive`, `purple`, `red`, `silver`, `teal`, `white` et `yellow`
  - En utilisant le code RGB pour (Red, Green, Blue) avec des valeurs de 0 à 255
  - En utilisant la valeur hexadécimale





- La propriété *couleur* s'applique généralement au texte ( `color` ) ou à l'arrière-plan ( `background-color` ).
- La propriété `background-image` définit l'image d'arrière-plan d'un ou plusieurs éléments ou encore de tous les éléments. Il est fortement recommandé de spécifier aussi une couleur normale d'arrière-plan, qui sera utilisé en remplacement.

## Exemple

```
background-image: url("images/monImage.jpg");  
background-color: white;
```



## CSS

CSS et HTML

Sélecteurs de CSS

Mise en forme

**Mise en page**

Fonctionnalités mal supportées

Langages côté serveur

JavaScript

Outils et Références





- Il existe deux sortes d'éléments HTML :
  - Les **blocs** : `<p>`, `<h1>`, `<ul>` ...  
`<div>` est un bloc générique.
  - Les éléments **en ligne**, qui doivent être placés à l'intérieurs de blocs : `<a>`, `<img>`, `<em>` ...  
`<span>` est un élément en ligne générique.
- On s'intéresse ici principalement aux **blocs** et à la manière dont les placer les uns par rapport aux autres.





margin (marge)

border (bordure)

padding (espacement)





- Le **flottement** et le **positionnement** sont des outils permettant une mise en page complexe : mise en forme en colonnes, chevauchement d'éléments, etc.
- Le positionnement consiste à préciser où doivent apparaître des éléments de manière relative, par rapport à un autre élément ou encore par rapport à la fenêtre du navigateur.
- Le flottement n'est pas vraiment un positionnement : les éléments dits flottants sont pris dans le flux et les autres éléments les contournent.





- Le flottement est défini par la propriété `float` qui peut prendre les valeurs :
  - `left` : l'élément sera contourné par la droite.
  - `right` : l'élément sera contourné par la gauche
  - `none` : valeur par défaut, sert principalement pour écraser un style existant.
- Tout bloc (image, texte, etc.) peut être défini comme un élément flottant.
- Il existe une dizaine de règles précises qui gouvernent le comportement des éléments flottants.





- Le positionnement `static` est le comportement par défaut :
  - l'élément est une boîte rectangulaire faisant partie du flux i.e. l'ordre des déclarations contenues dans le **code source est respecté**.
- Le positionnement `relative` permet de **décaler** un élément d'une certaine distance.





Le positionnement `absolute` permet de retirer totalement un élément du flux et de le positionner par rapport à son conteneur (plus précisément, par rapport au conteneur le plus proche ayant un positionnement relatif).

- Le positionnement `fixed` permet de retirer totalement un élément du flux et de le positionner par rapport à la fenêtre d'affichage.  
**Attention** : Ne fonctionne pas avec Internet Explorer 6.

## Remarque

Pour un positionnement absolu par rapport à la page (cas le plus fréquent), on utilisera `body { position: relative; }`.





- Une fois le type de positionnement choisi, on peut spécifier les propriétés de décalage : `top` , `right` , `bottom` et `left` .
- Le décalage se décrit à partir du bord le plus proche du bloc conteneur.
- Leur valeur peut être exprimée via une longueur, un pourcentage (par rapport à la taille du bloc conteneur) ou être choisie automatiquement `auto` .





- La visibilité d'un élément peut être complètement contrôlée en utilisant la propriété `visibility` :
  - `visible` permet de rendre visible l'élément
  - `hidden` permet de rendre invisible un élément ; la mise en page continue à le prendre en compte



## CSS

CSS et HTML

Sélecteurs de CSS

Mise en forme

Mise en page

**Fonctionnalités mal supportées**

Langages côté serveur

JavaScript

Outils et Références





- `p>em` : `<em>` directement à l'intérieur d'un `<p>` (pas IE6)
- `li+li` sélectionne les `<li>` qui suivent juste un autre (pas IE6)
- `*:lang(fr)` sélectionne les éléments dont la langue est définie comme le français (avec un attribut `lang=`, pas IE6 et IE7)
- `em:after {content: "!"}` ajoute un « ! » après tous les `<em>` (`*:before` existe aussi, pas IE6 et IE7)
- Valeur `inherit` indique que la valeur est héritée des éléments de la hiérarchie (pas IE)
- `max-width` / `min-height` /etc. permettent de donner des hauteurs et largeurs minimales et maximales aux blocs (pas IE6)
- `display: table-cell;` (et `table`, `table-row` ...) permet de mettre en page n'importe quels éléments comme des tables (pas IE6 et IE7)
- Compteurs automatiques (pas IE6 et IE7), etc.





Plus de noms de couleurs (marche **presque dans IE** !)

- Différentes feuilles de style suivant la résolution de l'écran, etc. (**seulement Opera**)
- Sélecteurs suivant la valeur d'un attribut (ex. `a[href$=".pdf"]` , **pas IE6**)
- Sélectionne les  $n^{\text{es}}$ ,  $2n^{\text{es}}$ , etc., éléments à l'intérieur d'un autre élément (**quasiment pas supporté**)
- Polices de caractères téléchargées automatiquement `@font-face` (tous navigateurs, mais formats de polices différents requis par les différents navigateurs). cf <http://craigmod.com/journal/font-face/>
- Et énormément d'autres choses, toujours en développement



## CSS

## Langages côté serveur

- Introduction

- Un exemple : PHP

- Introduction aux bases de données

## JavaScript

## Outils et Références



## CSS

## Langages côté serveur

### Introduction

Un exemple : PHP

Introduction aux bases de données

## JavaScript

## Outils et Références






- Le Web, ce n'est pas qu'un ensemble de documents HTML statiques !
- Les programmes côté serveur permettent :
  - de traiter des soumissions de formulaire ;
  - d'afficher de manière uniforme l'ensemble des pages d'un site ;
  - de proposer des applications interactives ;
  - de permettre à l'utilisateur d'ajouter ou modifier du contenu ;
  - etc.



- CGI (**Common Gateway Interface**) : interface normalisée permettant de faire communiquer le serveur Web avec un programme s'exécutant sur le serveur ;
- CGI permet d'utiliser n'importe quel langage de programmation (**compilés** comme C, C++, Java, ou **interprétés** comme Perl, Python, Ruby, etc.) pour écrire des langages côté serveur.
- Mais certains langage sont plus adaptés au développement Web :
  - comportent des fonctions spécialement dédiées à HTTP, HTML, etc. ;
  - s'intègrent de manière plus efficace et plus pratique avec le serveur Web (moyennant des extensions logicielles) ;
  - ont une syntaxe spécialement conçue, qui mêle code HTML envoyé tel quel et instructions de programmation interprétées.
- Quelle que soit la technologie utilisée, **le code du programme n'est pas accessible par le navigateur Web**, seulement le résultat de son exécution.



 : un des langages les plus populaires, s'intègre très facilement avec Apache (libre)

**ASP et ASP.NET** : destiné à être utilisé avec IIS (Microsoft, commercial)

**ColdFusion** (Adobe, commercial)

**JSP (Java Server Pages)** : permet de mêler instructions Java et code HTML ; nécessite un serveur d'applications Java (p. ex., Tomcat) en plus d'Apache (Sun, gratuit voire libre)

**Servlets Java** : véritables programmes Java, plutôt pour les applications complexes côté serveur avec peu d'interaction côté client ; nécessite un serveur d'applications Java en plus d'Apache (Sun, gratuit voire libre)



- Les langages présentés ci-avant restent assez basiques et généralistes.
- N'encouragent pas forcément une organisation propre d'un site Web.
- **Framework** : ensemble d'un langage de programmation, d'une bibliothèque de fonctions, d'outils externes, de bonnes pratiques à suivre. . .
- Permet d'abstraire la création d'une page Web.
- Suit en général le modèle **MVC** (voir transparent suivant).
- Inclut parfois la génération de code JavaScript **côté client** pour créer directement une application Web fortement dynamique (p. ex., validation de formulaire) ; intégration Ajax également.
- Fortement recommandé pour créer des applications complexes. . . mais également complexe à maîtriser !





- Principe de génie logiciel, utilisé dans d'autres domaines
- Particulièrement adapté au cas des applications Web !
- Séparation propre entre :
  - Modèle** : données manipulées par l'application et fonctions de manipulation de ces données ; **réutilisable** pour d'autres applications Web
  - Vue** : présentation des données ; facilement **échangeable** pour changer l'apparence et la structure du site
  - Contrôleur** : contrôle la manière dont l'utilisateur interagit, au travers de la vue, avec les données du modèle





ASPI.NET : DotNetNuke

ColdFusion : Model-Glue, Fusebox

Java : Struts, Spring, JavaServer Faces, Google Web Toolkit

Perl : Catalyst

PHP : CakePHP, Symphony, Zend

Python : Django

Ruby : Ruby on Rails (a eu beaucoup d'influence !)

Smalltalk : Seaside

... et beaucoup d'autres !



Permettent de créer des sites Web sans aucun développement.

## ■ Fonctionnalités :

- édition simplifiée de page (syntaxe wiki ou bbcode, ou contrôle JavaScript texte enrichi) ;
- ajout de contenu externe (images, documents annexes, etc.) ;
- gestion d'utilisateurs, contrôle d'accès, etc. ;
- modules de gestion de forums, de blogs ;
- thèmes graphiques prêt à l'emploi ;
- contrôle de version.

## ■ Suivant les CMS, extensions pouvant être nombreuses.

## ■ Certains CMS spécialisés : blogs (Dotclear, Movable Type, TypePad), commerce électronique (PrestaShop, Magento), forums (phpBB, MyBB), etc.



# Nombre de téléchargements hebdomadaires :



WordPress	PHP	433 767
Joomla!	PHP	189 429
Drupal	PHP	62 500
Umbraco	.NET/XSLT	5 670
eZ Publish	PHP	5 612
CMS Made Simple	PHP	4 903
SilverStripe	PHP	2 500
e107	PHP	2 242
Xoops	PHP	1 209
TikiWiki	PHP	373
phpWebSite	PHP	347
Typo3	PHP	100
Alfresco	Java	57
DotNetNuke	ASP.NET	?
Jahia	Java	?
Liferay	Java	?
modx	PHP	?
OpenCMS	Java	?
Plone	Python	?
TextPattern	PHP	?

(Chiffres difficiles à obtenir, source : Water&Stone)





Même en utilisant un CMS, il est utile de connaître les technologies de base du Web (HTML, CSS, JavaScript, langage côté serveur) :

- pour créer ses propres styles CSS (quasi-indispensable) ;
- pour développer des applications complexes, propres au site ;
- pour développer ou adapter des petites extensions ;
- pour comprendre ce qui se passe en cas de problème ;
- pour s'assurer que les pages respectent certaines conditions (validité W3C, accessibilité).





Frontière peu nette entre les deux... Les Wikis mettent l'accent sur :

- a priori, tout est éditable par un grand nombre d'utilisateurs ; pas de notion de « propriété » d'une page ou d'un document.
- insistance sur le contrôle de versions

Nombreux wikis : MediaWiki, TWiki, Dokuwiki...





## Point auxquels faire attention :

- la licence (la plupart des CMS populaires sont libres, mais pas tous)
- le langage sous-jacent, important s'il y a besoin de modifier/personnaliser le logiciel, et également important pour savoir comment le déployer
- la disponibilité de fonctionnalités spécifiques au site : blogs, forums, support vidéo, etc. Il est possible d'utiliser des composants externes pour chacune des tâches, mais plus simple de rester sur un seul système.





- Plus lent qu'un site classique ; fonctionnalités de cache permettant d'accélérer, mais peut poser problème dans certains contextes.
- Failles de sécurité, code sur lequel on n'a pas de contrôle.
- Logiciels orphelins.
- Migration vers un autre système potentiellement coûteuse.
- Non adapté à tous les usages.





nécessaire d'avoir accès au code source, donc comment savoir avec quelle technologie côté serveur un site a été réalisé ?

- URL : (.php pour PHP, .jsp pour JSP, .asp pour ASP, /servlet/ pour une servlet. .). Tout cela est configurable, mais bien souvent la configuration par défaut est utilisée.
- Ça peut être écrit en toutes lettres sur le site (souvent pour un CMS)
- Regarder les commentaires dans le code HTML, CSS. . .
- Organisation des fichiers et répertoires, bibliothèques JavaScript ou CSS chargées, etc.
- Cookies, en particulier identifiants de session



## CSS

## Langages côté serveur

Introduction

Un exemple : PHP

Introduction aux bases de données

## JavaScript

## Outils et Références



- Script PHP : document HTML (par exemple), dans lequel est incorporé du code PHP.

- Le code PHP est à l'intérieur d'une pseudo-balise `<?php ... ?>` (ou `<? ... ?>`, ou `<?= ... ?>` qui est un raccourci pour `<? echo ... ?>`).

## Exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">

<html>
  ...
  <body>
    <h1><?php echo 2+2; ?></h1>
  </body>
</html>
```





- Un script PHP est une suite d'instructions terminées par des points-virgules.

## Exemple (Écrire une phrase avec l'instruction echo)

```
echo 'Ceci apparaîtra dans la page générée.';
```

- Ces instructions contiennent des éléments variables ou constants, peuvent être conditionnées ou encore itérées plusieurs fois.





Les paramètres HTTP peuvent être récupérées en PHP grâce au tableaux associatifs `$_GET` et `$_POST`.

- Les valeurs de ce tableau peuvent être des variables simples ou des tableaux indicés : ces derniers sont les paramètres à choix multiples dont on a suffixé le nom de `[]` dans le code HTML.

## Exemple

```
echo "<p>Votre login est : " . $_POST["login"] . "</p>";  
echo "<p>Vous avez coché les genres : ";  
for($i=1;$i<=count($_POST['genre']);$i=$i+1) {  
    echo $_POST['genre'][$i] . " ";  
}  
echo "</p>";
```



## CSS

## Langages côté serveur

Introduction

Un exemple : PHP

Introduction aux bases de données

## JavaScript

## Outils et Références





- **SGBD** : Système de Gestion de Bases de Données
- Fournit des méthodes **efficaces** pour gérer des données qui répondent à une structure (un **schéma**) précis.
- Rechercher des données : **requêtes**
- Ajouter, supprimer, modifier des données : **mises à jour**
- Traite de manière rapide de grandes quantités de données.
- Gérer des transactions (ne pas donner à deux clients la même place dans un train !)





Modèle le plus répandu et le plus classique.

Les données sont organisées en des **tables**, chacune des colonnes représentant un **attribut** des données.

## Exemple

Prénom	Nom	Date de naissance
Jean	Dupont	1967-08-07
Pascale	Dupuis	1981-09-12
Alfred	Lambert	NULL

- Chaque attribut (colonne) est **typé**.
- SQL (**S**tructured **Q**uery **L**anguage) : langage standard de requête et de mise à jour des données (petites variantes suivant les SGBD).





Oracle

IBM DB2

Microsoft SQL Server

Microsoft Access

MySQL

PostgreSQL

commercial, le plus utilisé en entreprise

commercial

commercial

commercial, pauvre en fonctionnalités

libre, le plus utilisé sur le Web

libre, plus abouti et plus complexe que MySQL





## (Types MySQL)

**INT** : entier ( 42 )

**REAL** : nombre en virgule flottante ( 3.14159 )

**VARCHAR(N)** : chaîne de caractères ayant au plus N caractères ; les valeurs sont délimitées par des apostrophes ( 'Ceci est une chaîne' ).

**TEXT** : longue chaîne de caractères, sans limite de taille

**DATE** : date ( 2005-11-08 )

**TIME** : temps ( 09:30:00 )





- **NULL** : valeur spéciale
- Dénote l'absence de valeur.
- Différent de `0`, de `''` ...
- Une comparaison normale (`=`, `<>`) avec **NULL** renvoie toujours FAUX.
- **IS NULL**, **IS NOT NULL** peuvent être utilisées pour tester une valeur.
- Chacune des colonnes doit être déclarée comme acceptant ou non la valeur **NULL**.





Langage standardisé pour interagir avec un SGBD.

■ Exemple de requête :

```
SELECT a.name, COUNT(*)  
FROM actors a, casting m  
WHERE a.id=m.actor_id  
GROUP BY a.name
```

■ Exemple de mise à jour :

```
UPDATE availabilities  
SET nb_available=nb_available-1  
WHERE train_id=1234
```





- SGBD très utiles pour beaucoup d'applications Web (annuaires, catalogues, réservations, forums de discussion, blogs, etc.)
- Possibilité d'accéder aux bases de données plus ou moins simplifiée suivant les langages :
  - En PHP, bibliothèques de fonctions propres à chaque SGBD (p. ex., `mysql` ou `mysqli` pour MySQL).
  - En JSP, bibliothèque JDBC avec les scriptlets et balises `<sql:*>` avec la JSTL, pour accéder à n'importe quel SGBD.



```
mysql_pconnect("server","login","password")
{ echo "<p>Desolé, connexion impossible</p>"; exit; }
if(!mysql_select_db('database'))
{ echo "<p>Desolé, accès à la base impossible</p>"; exit; }

$resultat= mysql_query("SELECT * FROM Films");
if($resultat) {
    while($film=mysql_fetch_assoc($resultat)){
        echo "<p>".$film["Titre"]." est paru en ".
            $film["Annee"]." </p>";
    }
} else {
    echo "<p>Erreur dans l'exécution de la requête.</p>";
    echo "<p>Message de MySQL: ".mysql_error()."</p>";
}
```



CSS

Langages côté serveur

**JavaScript**

Introduction

L'objet Node

Frameworks JavaScript

Outils et Références





CSS

Langages côté serveur

**JavaScript**

Introduction

L'objet Node

Frameworks JavaScript

Outils et Références



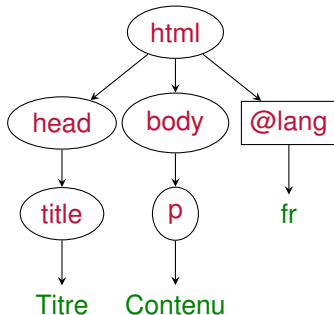


- PHP, CGI... : permettent des comportements dynamiques **côté serveur**. Nécessitent un échange entre le navigateur et le serveur Web (soumission d'un formulaire, clic sur un lien) pour chaque comportement dynamique souhaité.
- JavaScript : permet des comportements dynamiques **côté client** : manipulation des fenêtres, changement dynamique du code HTML/CSS, interaction fine avec les formulaires...
- Permet la manipulation du DOM (**D**ocument **O**bject **M**odel), la représentation du document HTML comme un arbre, les balises étant les nœuds de l'arbre.
- « Dynamic HTML » (**DHTML**) : JavaScript + DOM + CSS
- **Alternatives** : VBScript (Internet Explorer uniquement), Java/Flash/Silverlight (plus isolé du reste de la page).
- Rien à voir avec Java !



## Example

```
<html lang="fr" xmlns="...">  
  <head><title>Titre</title></head>  
  <body><p>Contenu</p></body>  
</html>
```





- Langage de Programmation
- Script (ou programme) JavaScript : fichier texte, instructions terminées par des points-virgules
- Normalisé sous le nom d'**EcmaScript** (**JavaScript** est historiquement le nom de l'implémentation Netscape, **JScript** étant l'équivalent chez Microsoft)
- En pratique, actuellement, seuls les navigateurs graphiques supportent JavaScript. Conséquence : sauf applications complexes, **un site doit être utilisable sans JavaScript.**



■ toto.js contenant des fonctions JavaScript (**function**)

■ Dans le `<head>` du HTML :

```
<script src="toto.js" type="text/javascript"></script>
```

- Gestionnaires d'événement comme attributs des balises HTML (cf plus loin).
- On peut aussi mettre directement du code JavaScript à l'intérieur des balises `<script>` mais :
  - Mélange de plusieurs langages dans le même document : ajoute de la complexité.
  - Le document complet doit rester du HTML/XHTML valide !
  - Impossible de partager les mêmes fonctions dans plusieurs pages Web sans les recopier à chaque fois.
- Dernière possibilité : utiliser le pseudo-protocole **javascript:** dans un lien.





- Les événements sont des actions de l'utilisateur, qui vont pouvoir donner lieu à une **interactivité**.
- L'événement correspond à un clic de souris, une sélection d'un champ, une touche pressée au clavier. . .
- Il est possible d'associer des fonctions à des événements.
- Dans le gestionnaire d'événements, **this** représente le nœud courant.
- La syntaxe est la suivante :

```
onevenement="Action_Javascript_ou_Fonction()"
```





**onblur** se produit lorsque l'élément ( `<input>` , `<textarea>` , `<select>` ) perd le focus, c'est-à-dire que l'utilisateur clique hors de cet élément, celui-ci n'est alors plus sélectionné comme étant l'élément actif.

**onchange** se produit lorsque l'utilisateur modifie le contenu d'un champ de données ( `<input>` , `<textarea>` , `<select>` ).

**onclick** se produit lorsque l'utilisateur clique sur l'élément ( `<a>` , `<input type="radio">` , `<input type="checkbox">` ) associé à l'événement. Si le gestionnaire d'événement **renvoie** `false` (avec `return false`), le clic n'est pas effectué.

**onfocus** se produit lorsque l'utilisateur donne le focus à un élément ( `<input>` , `<textarea>` , `<select>` ), c'est-à-dire que cet élément est sélectionné comme étant l'élément actif.





- onload** se produit lorsque le navigateur de l'utilisateur charge la page en cours ( `<body>` ).
- onreset** se produit lorsque l'utilisateur efface les données d'un formulaire ( `<form>` ) à l'aide du bouton Reset.
- onsubmit** se produit lorsque l'utilisateur clique sur le bouton de soumission d'un formulaire ( `<form>` ). Si le gestionnaire d'événement **renvoie** false, le formulaire n'est pas soumis.





CSS

Langages côté serveur

**JavaScript**

Introduction

L'objet Node

Frameworks JavaScript

Outils et Références




■ L'objet **Node** est l'objet central du modèle DOM (Document Object Model).

- Chaque élément, chaque attribut et chaque donnée en caractères représentent des nœuds distincts. Ces nœuds forment une arborescence.
- L'objet Node dispose de propriétés et de méthodes pour accéder aux différents nœuds, peu importe s'ils sont placés très bas dans l'arborescence.
- La propriété **nodeType** permet de connaître le type du nœud (élément, attribut, texte, document, commentaire...)

## Exemple

```
if (node.nodeType===Node.ELEMENT_NODE) {  
    ...  
}
```



 **document** est un objet prédéfini représentant le document actuel affiché dans le navigateur.

- `node = document.documentElement` met dans la variable `node` l'**élément racine** (c'est-à-dire le nœud correspondant à l'élément `html` )
- `node = document.getElementById("titi")` accède à un élément HTML qui possède un attribut `id` valant `titi`.
- `valeur = node.nodeValue` sauvegarde la valeur ou le contenu d'un nœud :
  - pour les nœuds texte, c'est le texte,
  - pour les nœuds attribut la valeur affectée à l'attribut
- Il existe aussi une fonction `document.getElementsByTagName` qui renvoie un tableau d'éléments.



- `node.className="nouvelle_classe"` pour changer le nom de la classe CSS à laquelle appartient le nœud.
- `node.style.borderStyle="valeur"` pour changer le style de bordure d'un nœud.
- `node.style.visibility="valeur"` pour changer la visibilité d'un nœud
- `node.style.display="valeur"` pour changer la propriété CSS display d'un nœud.

## Règle générale

On peut changer de cette façon n'importe quelle propriété CSS, d'un nœud. Le nom de la propriété en JavaScript est identique au nom CSS, sauf que les traits d'unions sont remplacés par une majuscule sur la lettre suivante. Les valeurs des propriétés en JavaScript sont identiques aux valeurs CSS (mais doivent être mis entre guillemets).



## JavaScript :

```
function Test() {  
    document.getElementById("paragraphe").style.color = "blue";  
}
```

## HTML :

```
<p id="paragraphe" style="color: red;">un texte</p>  
<a href="" onclick="Test()">Test</a>
```

**Explication :** L'exemple contient un paragraphe avec le nom id *paragraphe* et un lien qui si on le clique appelle la fonction *Test()*. Cette fonction change la propriété CSS *color* du paragraphe, de telle sorte que le paragraphe perde sa couleur rouge et devienne bleu.

CSS

Langages côté serveur

**JavaScript**

Introduction

L'objet Node

Frameworks JavaScript

Outils et Références





- **Très grande disparité** dans le support JavaScript des différents navigateurs.
- La norme EcmaScript n'est arrivée que très tardivement.
- Comme d'habitude, IE est le plus éloigné de la norme, mais même au sein de Firefox/Opera/Safari, nombreuses différences de comportement.
- Implémentation encore en cours de nombreuses fonctionnalités.
- Présenté dans ce cours : fonctionnalités élémentaires, fonctionnant dans tous les navigateurs.





3 bibliothèques de fonctions JavaScript permettant de travailler à un plus haut niveau et d'abstraire les différences de comportement des navigateurs.

■ Fonctionnalités :

- Détection du type de navigateur
- Effets d'animation
- Interactions complexes (p. ex., cliquer/glisser)
- Meilleure gestion des événements
- AJAX
- Outils prêts être utilisés comme calendriers, tables dynamiques, etc.

■ Important de vérifier quelles versions de navigateurs sont pris en charge !





- Yahoo! User Interface Library
- Prototype/Scriptaculous/Rico
- jQuery/jQuery UI
- Qooxdoo
- Dojo
- voir aussi certains frameworks d'applications Web (notamment, Google Web Toolkit)





Formulaires HTML

CSS

Langages côté serveur

JavaScript

Outils et Références

26 Septembre 2011



- N'importe quel éditeur de texte

## ■ Une variété de navigateurs graphiques

- Fonction **Aucun style** de Firefox
- Console JavaScript de nombreux navigateurs
- Extension **Firebug** de Firefox
- **Valideur** CSS : <http://jigsaw.w3.org/css-validator/>
- Un serveur Web (p. ex., Apache)
- L'interpréteur du langage côté serveur (p. ex., mod\_php pour Apache)
- Pour JSP, Servlets : serveur d'applications Java (p. ex., Tomcat)
- Un SGBD (p. ex., MySQL) ; sous Windows, EasyPHP regroupe Apache+PHP+MySQL
- Un moyen de transférer les programmes vers le serveur Web distant (SSH, FTP, application Web. . .)





- Les spécifications de CSS :
  - <http://www.w3.org/TR/REC-CSS1>
  - <http://www.w3.org/TR/CSS21/>
  - <http://www.w3.org/Style/CSS/current-work>
- Wiki de support des standards du Web :  
<http://www.webdevout.net/>
- Tests Acid de conformité des navigateurs :
  - <http://acid1.acidtests.org/>
  - <http://acid2.acidtests.org/>
  - <http://acid3.acidtests.org/>





## HTTP

- <http://www.faqs.org/rfcs/rfc2616.html>
- <http://www.ietf.org/rfc/rfc2109.txt>
- <http://www.ietf.org/rfc/rfc2965.txt>

## PHP

- <http://www.php.net/>
- *Pratique de MySQL et PHP*, Philippe Rigaux, O'Reilly

MySQL : <http://dev.mysql.com/doc/>





- Standards (beaucoup moins lisibles que pour HTML ou CSS)
  - Langage ECMAScript, <http://www.ecma-international.org/publications/standards/Ecma-262.htm>
  - Spécifications DOM, <http://www.w3.org/DOM/DOMTR>
- <http://www.webdevout.net/>





- *HTML et XHTML: The Definitive Guide*, O'Reilly
- *CSS: The Definitive Guide*, O'Reilly
- *Pratique de PHP et MySQL*, Dunod
- *JavaScript: The Definitive Guide*, O'Reilly
- *jQuery Cookbook*, O'Reilly





**Par le téléchargement ou la consultation de ce document, l'utilisateur accepte la licence d'utilisation qui y est attachée, telle que détaillée dans les dispositions suivantes, et s'engage à la respecter intégralement.**

La licence confère à l'utilisateur un droit d'usage sur le document consulté ou téléchargé, totalement ou en partie, dans les conditions définies ci-après et à l'exclusion expresse de toute utilisation commerciale.

Le droit d'usage défini par la licence autorise un usage à destination de tout public qui comprend :

- le droit de reproduire tout ou partie du document sur support informatique ou papier,
- le droit de diffuser tout ou partie du document au public sur support papier ou informatique, y compris par la mise à la disposition du public sur un réseau numérique,
- le droit de modifier la forme ou la présentation du document,
- le droit d'intégrer tout ou partie du document dans un document composite et de le diffuser dans ce nouveau document, à condition que :
  - L'auteur soit informé.

Les mentions relatives à la source du document et/ou à son auteur doivent être conservées dans leur intégralité.

Le droit d'usage défini par la licence est personnel et non exclusif.

Tout autre usage que ceux prévus par la licence est soumis à autorisation préalable et expresse de l'auteur : [sitopedago@telecom-paristech.fr](mailto:sitopedago@telecom-paristech.fr)

