Architecture des applications de la toile - L3
Introduction Système Réseaux (partie web) - Master IC
-
(Introduction aux) Technologies côté-client :
-
HTML - Feuilles de style CSS (initiation) - JavaScript (initiation)

Pierre Pompidor

Table des matières

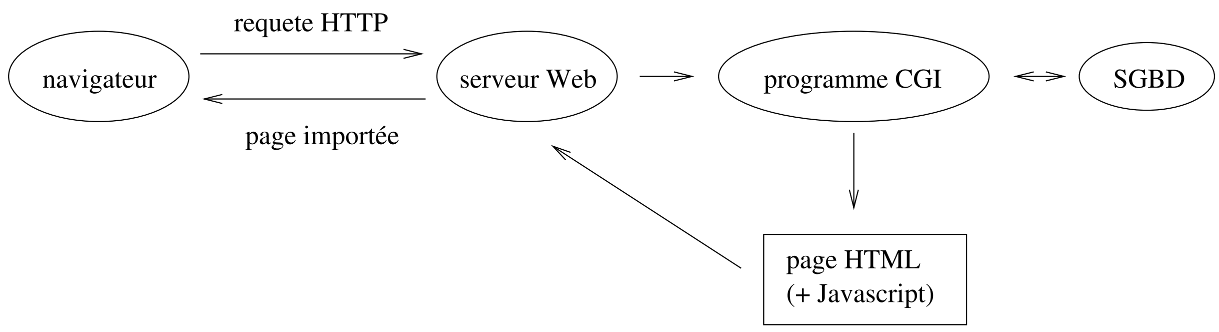
1 Architectures possibles (client : navigateur / serveur : serveur HTTP) 2

2 Présentation rapide des technonologies de programmation "Web" 3

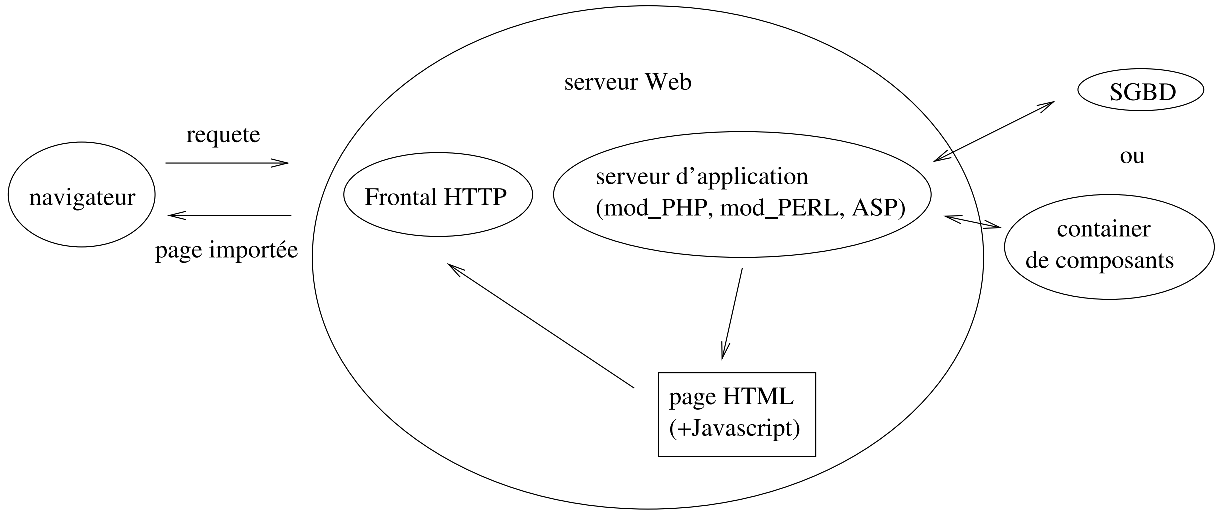
3 Programmation côté client 4
- 3.1 HTML (HyperText Mark-up Language : un langage (à balises) de présentation : 4
  - 3.1.1 Entête d'un document HTML : 4
  - 3.1.2 Corps d'un document HTML : 4
- 3.2 Les feuilles de style CSS (Cascading Style Sheet) : 7
  - 3.2.1 Types de redéfinitions possibles : 7
  - 3.2.2 Quelques propriétés css courantes 7
- 3.3 JavaScript 8
  - 3.3.1 Hiérarchie de classes et principales propriétés de l'API DOM Javascript 8
  - 3.3.2 Comment ne pas perturber un navigateur ne supportant pas Javascript ? (vérification un peu désuète) 9
  - 3.3.3 Comment tester le navigateur ? 9
  - 3.3.4 Où créer les fonctions javascript ? 9
  - 3.3.5 Où appeler des fonctions javascript ? 9
  - 3.3.6 Comment différer ou planifier l'appel des fonctions javascript ? 10
  - 3.3.7 Comment créer une fenêtre standard de saisie ou une fenêtre d'alerte ? 10
  - 3.3.8 Comment créer une nouvelle fenêtre ? 10
  - 3.3.9 Comment agir sur une fenêtre mère à partir de la fenêtre fille ? 10
  - 3.3.10 Comment vérifier que tous les champs d'un formulaire ont bien été saisis ? 10
- 3.4 JavaScript/DOM/DHTML : 11
  - 3.4.1 Méthodes de l'API DOM de Javascript permettant d'accéder à l'arbre DOM : 11
  - 3.4.2 La gestion des divisions : 12
- 3.5 XML (eXtensible Markup Language) : 13
  - 3.5.1 XML et les CSS : 13
  - 3.5.2 XHTML 14

1 Architectures possibles (client : navigateur / serveur : serveur HTTP)

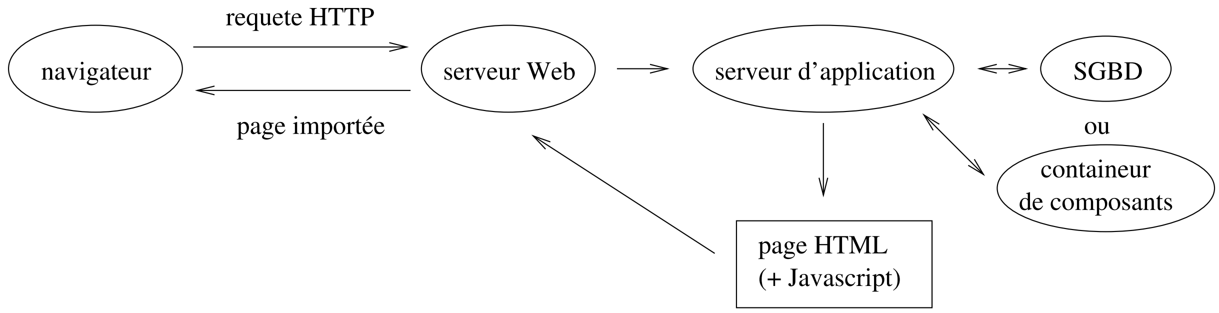
Première école : CGI



Deuxième école : serveur d'application intégré au serveur Web



Troisième école : serveur d'application externe
architecture trois (ou quatre) tiers
(généralement présentation/métier/données (ou composants)



2 Présentation rapide des technonologies de programmation "Web"

Sur le client (navigateur) :

Tous les exemples cités dans ce support de cours sont cencés être compatibles IE7 / Firefox 2.

- **HTML** (HyperText Mark-up Language) → langage permettant :
  - la présentation et d'agencement de textes et d'images
  - la création d'ancres (liens hypertextes)
  - la création de formulaires qui sont des ensembles de composants permettant à l'internaute de sélectionner ou de saisir des données :
    - dans des **zones de saisies**
    - en cliquant sur des **boutons**
    - en sélectionnant des items dans des **listes déroulantes**, des listes de **boutons radio** ou de **boîtes à cocher**
- **Les feuilles de style CSS** (Cascading Style Sheet) → ensemble de styles :
  permettant, via des styles attachés aux balises HTML, de (re)définir la présentation des textes, des liens, et plus généralement de tous les composants des pages HTML.
- **JavaScript** → langage de script permettant principalement sur le client :
  - le contrôle des saisies dans les formulaires ;
  - la modification des valeurs des composants ;
  - la création de fenêtres séparées ;

  Attention, JavaScript n'a rien à voir avec le langage **Java** (enfin presque).
- **DHTML** (Dynamic HTML) → collaboration entre HTML (notamment la balise div) et l'accès à la représentation interne de la page web dans le navigateur par Javascript permettant la mise à jour, le déplacement, l'apparition ou la disparition de composants de la page web.
- **XML** (eXtensible Markup Language) → :
  "langage" (plutôt ici format de données) permettant de ne représenter que la sémantique des documents web (XHTML → HTML respectant une syntaxe stricte (application des règles syntaxiques des documents XML))

Ne sont pas présentés dans cette première partie :

- **La programmation avancée en Javascript** qui tire partie de la grande dynamicité de ce langage.
- **Les frameworks Javascript** qui :
  - standardisent des "gadgets" graphiques (appelés **widgets**) comme les menurs déroulants ou contextuels, les onglets, le glisser-déposer..., assez pénibles à coder ex nihilo (j'aime bien ce terme) ;
  - interfacent l'instanciation de l'objet **XMLHTTPRequest** permettant la mise à jour asynchrone de la page web (partie prenante d'un synchrétisme de technologies appelé **AJAX**).
- **XSL** (eXtensible Stylesheet Language) :
  langage permettant d'écrire des feuilles de style pour médiatiser les documents XML (→ est enseigné dans tous ses états dans le module "Galaxie XML" de la première année du Master).
- **Applets java** → exécution de "petits" programmes Java interfacés par différentes bibliothèques graphiques (AWT, SWING, ...) (→ sont enseignées dans différents modules de Master) ;
- Les technologies **Flash/Flex** qui en utilisant les langages **MXML** (langage de spécification graphique basé sur XML) et **ActionScript** (langage à objets) permettent de réaliser des **RIA** (Rich Internet Applications) (→ enseignées en Technologies avancées du Web en M2).
- et évidemment les vieux **ActiveX** → composants (souvent graphiques) téléchargeables (Windows exclusivement) qui ont la douce particularité de s'exécuter directement sur la machine cliente en multipliant les failles de sécurité (et ne sont évidemment enseignés nulleart).

3 Programmation côté client

3.1 HTML (HyperText Mark-up Language : un langage (à balises) de présentation :

3.1.1 Entête d'un document HTML :

L'entête d'un document HTML, définie entre les deux balises <head> et <head>, spécifie le contrôle effectué sur celui-ci. Nous allons principalement y trouver :
- des **directives** à placer dans la réponse HTTP ;
- des **déclarations de styles CSS** (ou l'importation de feuilles de styles CSS externes) ;
- du **code Javascript** gérant les interactions avec l'utilisateur.

Balises principales :
- Titre (banière, bookmarks ..) : <title> ... </title>
- Insertion de directives dans la réponse HTTP : <meta ...>
  - encodage : <meta http-equiv="Content-type" content="text/html; charset=type_encodage">
    exemple : <meta http-equiv="Content-type" content="text/html; charset=ISO-8859-1">
  - raffraîchissement : <meta http-equiv="refresh" content="nb_secondes; URL=URl_de_la_page">
- Déclaration de styles CSS :
  - internes : <style type="text/css"> ... </style>
  - externes : <link rel="stylesheet" type="text/css" href="URL" />
- Codes Javascript : <script language="javascript"> ... </script>

3.1.2 Corps d'un document HTML :

Le corps d'un document HTML, défini entre les deux balises <body> et <body>, spécifie le contenu de la page web.
Exemples d'attributs de la balise body spécifiant les couleurs et les images d'avant et d'arrière-plan :
- attribut fixant la couleur d'arrière-plan : bgcolor=...
- attribut fixant l'image d'arrière-plan : background=...

Présentation des chaînes de caractères :
- **Taille** :
  - par le biais des balises hx de la plus grosse : <h1> ... </h1> à la plus petite : <h7> ... </h7>
  - ou par le biais de l'attribut size de la balise ().
- **Style** :
  - gras : ... ;
  - italique : <i> ... </i> ;
  - souligné : <u> ... </u>.
- **Couleur** du texte par le biais de l'attribut color de la balise :
   ...
  liste des couleurs prédéfinies : Aqua, Black, Blue, Gray, Green, Lime, Maroon, Navy, Olive, Purple, Red, Silver, Teal, White, Yellow (je recommande particulièrement Navy ou Olive qui sont particulièrement laides)

Présentation des images : Création d'une image :

Création des ancres (liens hyper-textes) :
- Création d'une ancre externe :
   texte de l'ancre
  Exemple : mais cliquez donc ici
- Création d'une ancre interne à la page :
  pour créer l'ancre : texte de l'ancre
  pour définir le point d'ancrage :

Structures d'agencements (les listes et les tableaux) :

- Les **listes** :
  - liste d'éléments ordonnés : `<ol>`
  - liste d'éléments non ordonnés : `<ul>`
  - item d'une liste : `<li>`

  → des styles CSS pourront modifier l'apparence des listes.

  Voici deux niveaux de listes imbriquées :

```
<ol>
   <li> Entête de la première sous-liste
   <ol>
       <li> ... </li>
       ...
       <li> ... </li>
   </ol>
   </li>
   ...
   <li> Entête de la dernière sous-liste
   <ol>
       <li> ... </li>
       ...
       <li> ... </li>
   </ol>
   </li>
</ol>
```

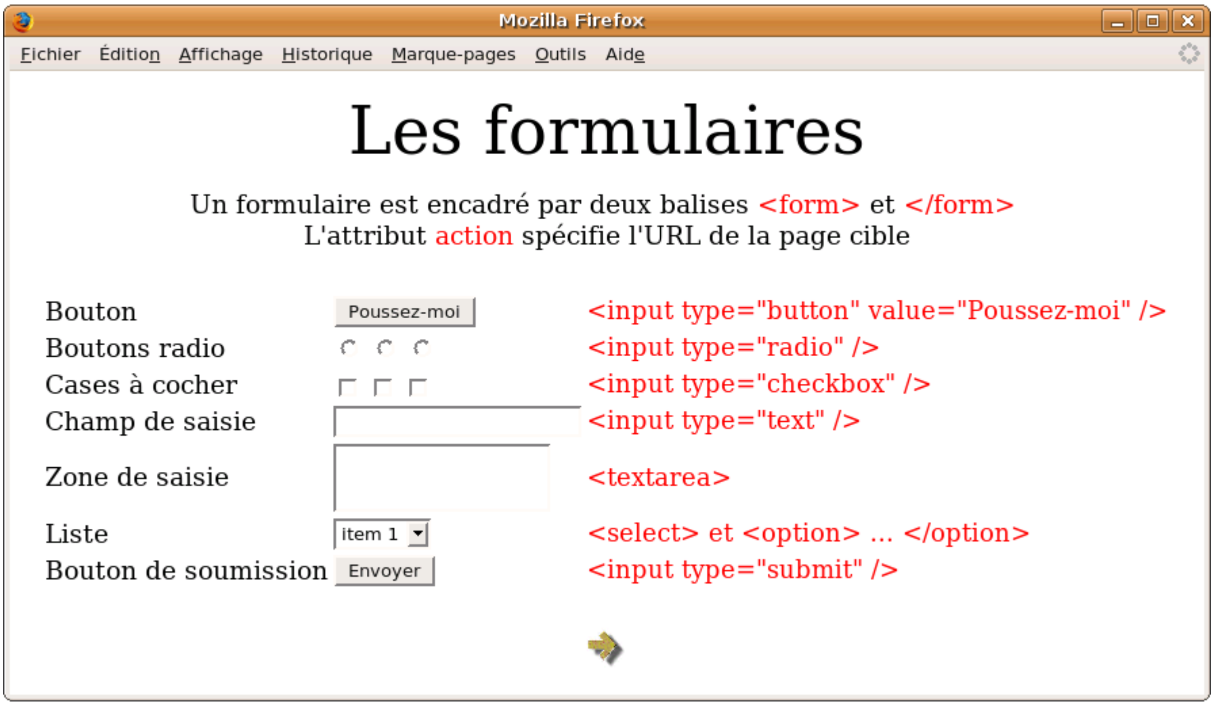
- Les **tableaux** :
  - balise d'encadrement du tableau : `<table border="épaisseur_de_la_bordure"> ... </table>`
  - création d'une ligne : `<tr>`
  - création d'une cellule : `<td>`
  - balise de fin : `</table>`

  Voici la structure HTML d'un tableau de trois lignes sur trois colonnes :

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

Création d'un formulaire : Un formulaire est encadré par la balise de début et de fin <form> :
<form method=GET action="URL_de_la_page_à_importer"> ... </form>

- Bouton : <input name="..." type="button" value="..." />
- Bouton personnalisé (contenant une image ou plus généralement du code HTML) :
  <button> </button>
- Champs de saisie : <input name="..." type="text" value="..."> champ de texte </input>
- Zones de saisie : <textarea name="..." rows="..." cols="..."> texte </textarea>
- Liste déroulante : <select name="..."> <option> texte de l'item </option> ...
  </select>
- Liste de boutons radio : <input type="radio" name="btr" value="1" checked="checked" />
  <input type="radio" name="btr" value="2" />
- Liste de boîtes à cocher : <input type=checkbox name=boite value=1 />
  <input type=checkbox name=boite value=2 />
- Bouton de soumission du formulaire : <input type="submit" value="Envoyez" />
- Bouton de réinitialisation : <input type="reset" value="Réinitialiser" />



Soumission du formulaire - traitement côté serveur :

Un script CGI pourra exploiter :
- si il a été programmé en PERL : une chaîne de caractères composée des noms et des valeurs de chaque composant (nom1=valeur1&nom2=valeur2&...) dans **$ENV{'QUERY_STRING'}**.
  (Le module Perl cgi.pm permet d'accéder plus confortablement à ces paramètres...)
- si il a été programmé en PYTHON : un tableau associatif d'objets **MiniFieldStorage** correspondants à chaque composant du formulaire en retour de la fonction **FieldStorage()** accessible dans le module **cgi**.

Un script PHP pourra accéder :
- aux tableaux associatifs $_GET et $_POST ;
- des variables de mêmes noms que les paramètres et contenant leurs valeurs, nativement si la version de PHP est antérieure à 4.2, ou après modification de la configuration de PHP si sa version est supérieure.

3.2 Les feuilles de style CSS (Cascading Style Sheet) :

Les feuilles de style CSS permettent :
– dans le cas d'une page HTML, d'associer des propriétés d'affichage aux balises HTML prédéfinies : utiliser la balise **link** : <link href="..." rel="styleSheet" type="text/css">
– dans le cas d'une page XML, d'associer des propriétés d'affichage aux balises XML : utiliser la balise **xml-stylesheet** : <?xml-stylesheet href="..." type="text/css" ?>

3.2.1 Types de redéfinitions possibles :

– redéfinition du **style associé à une balise HTML prédéfinie** : nom_de_la_balise {...}
– redéfinition du **style de plusieurs balises HTML prédéfinies** : nom_balise_1, ... {...}
– redéfinition du **style d'une balise HTML fille d'une autre balise HTML** : nom_balise_1 nom_balise_2 {...} ou nom_balise_1 > nom_balise_2 {...}
– redéfinition du **style des balises HTML filles d'une autre balise HTML** : nom_balise_mère > * {...}
– redéfinition du **style des balises HTML conditionnées par la valeur d'une propriété** : nom_balise[nom_propriété=valeur] {...}
– définition d'une nouvelle **classe de style** :
  – une classe peut être universelle si elle peut s'appliquer à n'importe quelle balise : .maClasseDeStyle {liste de propriétés}
    → cette classe est appliquée à toutes les balises dotées d'un attribut de nom class et de valeur maClasseDeStyle
  – ou spécifique si elle ne peut être associée qu'à un nombre déterminé de balises : balise.maClasseDeStyle {...}
– les **pseudo-classes** et les **pseudo-éléments** :
  lié à un événement particulier ou à des positions relatives :.
  – modification de la présentation d'un lien suivant son état → la pseudo-classe A :
    A:link : lien hypertexte normal
    A:hover : lien hypertexte sélectionné lorsque la souris le pointe
    exemple : A:hover {text-decoration: none; color: red;} A:active : lien hypertexte actif
    A:visited : lien hypertexte visité
  – Formatage du texte d'un paragraphe → les pseudo-éléments appliqués à la balise <P> : Exemple : P:first-line { text-transform: uppercase; } pour mettre en majuscule la première lettre de chaque paragraphe.

3.2.2 Quelques propriétés css courantes

– structuration : **display**, valeurs possibles : **block**, **inline**, **invisible**, **none**
– marges : **margin-top**, **margin-bottom**, **margin-left**, m**argin-right**
– positionnements : **position**, **left**, **top**
– nom de la police : **font-family**, valeurs possibles : times, arial ...
– taille de la police : **font-size**, valeurs possibles :
  – symbolique : xx-small, x-small, small, medium, large, x-large, xx-large
  – en points : 10pt, ...
– poids de la police : **font-weight** (valeurs possibles : lighter, normal, bolder, bold)
– alignements : **text-align** (valeurs possibles : left, right, center, justify)
– couleur d'avant-plan : **color**
– couleur ou image d'arrière-plan : **background-color**, **background-image**
– propriétés sur les listes :
  – type de puce : **list-style-type** (none pour supprimer les puces, decimal, lower-roman, ...)
  – image en guise de puce : **list-style-image** (url(...))

3.3 JavaScript

Javascript permet de créer des fonctions et de les appeler lors d'un événement particulier (soumission d'un formulaire, clic sur un bouton, sur une ancre, ...). Javascript est un **langage orienté objets à prototypes**, (les objets ne sont pas instanciés au sein de classes mais sont créées par clonage).
Javascript permet d'accéder aux composants de la pages web (instanciés sous forme d'objets à partir d'une hiérarchie de classes définie par le DOM du navigateur (**DOM = Document Object Model**)). Le code Javascript peut donc lire, modifier ou supprimer n'importe quelle valeur ou n'importe quel composant de la page web.

3.3.1 Hiérarchie de classes et principales propriétés de l'API DOM Javascript

- window
  - document
    - anchors : collection des références aux liens
    - applets : collection des références aux applets
    - forms
      - elements (désuet)
    - images : collection des références aux images
    - links : collection des références aux liens
    - *all*

Attention la prpriété window.document.all n'est plus à utiliser !
Les objets particuliers seront accessibles par les fonctions **getElementBy...()** de l'API DOM de Javascript.

- window (suite)
  - frames : collection des références aux cadres
  - history : historique (s'emploie avec la méthode go)
  - location : URL courante de la fenêtre
  - name : nom de la fenêtre
  - navigator : navigateur
  - opener : fenêtre qui a ouvert cette fenêtre courante
  - parent : élément window parent de la fenêtre ou du cadre courant

- window (suite et fin)
  - screen
    - screen.height : hauteur de l'écran en pixels
    - screen.width : largeur de l'écran en pixels
    - screen.availHeight : hauteur disponible de l'écran en pixels
    - screen.availWidth : largeur disponible de l'écran en pixels
  - status : texte dans la barre d'état en bas du navigateur
  - top : fenêtre la plus haute dans la hiérarchie des fenêtres

Quelques propriétés/valeurs différentes entre Firefox 2 (exécuté sous Linux) et IE 7 (exécuté sous Vista) :

Propriété	Firefox 2	IE 7
système d'exploitation du client	navigator.platform (Linux i686)	idem (Win32)
nom du navigateur	navigator.appName (Netscape)	idem (Microsoft Internet Explorer)
hauteur intérieure de la fenêtre en pixels	window.innerHeight	document.body.clientHeight
largeur intérieure de la fenêtre en pixels	window.innerWidth	document.body.clientWidth

Quelques recettes de cuisine :

3.3.2 Comment ne pas perturber un navigateur ne supportant pas Javascript ? (vérification un peu désuète)

→ en incluant des commentaires HTML dans les balises `script` (balises qui ne seront pas interpétées si Javascript n'est pas reconnu (subtil)) :

```
<script language="javaScript" type="text/javascript">
<!--
...
//-->
</script>
```

3.3.3 Comment tester le navigateur ?

→ en utilisant la valeur de la propriété **appName** de l'objet **navigator**

```
var navigateur;
navigateur = navigator.appName;
switch (navigateur)
{
   case "Microsoft Internet Explorer" : window.location.href = "pageie.htm";
   break;
   case "Netscape" :   window.location.href = "pagefirefox.htm";
   break;
   default: ...;
}
```

3.3.4 Où créer les fonctions javascript ?

→ en les plaçant généralement dans l'entête HTML entre les deux balises `<script>` et `</script>` :

```
<script language="javaScript" type="text/javascript">
function maFonction () {
   ...
}
</script>
```

Les fonctions javascript peuvent être rassemblées dans un fichier séparé du document html et **suffixé par .js**. Il sera lié au document html par la balise `script` :

```
<script src="fonctions.js" language="javaScript" type="text/javascript"> </script>
```

3.3.5 Où appeler des fonctions javascript ?

- lors du chargement de la page (**onLoad**) : `<body onLoad="javascript:fonction();">`
- lors du déchargement de la page (**onUnLoad**) : `<body onUnLoad="javascript:fonction();">`
- lors du retaillage de la page (**onResize**) : `<body onResize="javascript:fonction();">`
- dans un formulaire :
  - sur un bouton (**onClick**) : `<input type=button value="..." onClick="fonction();">`
  - sur le changement d'une valeur ou d'une sélection (**onChange**)
  - sur la soumission du formulaire (**onSubmit**) : `<form onSubmit="..."`

– lors de la sélection d'une ancre : `<a href="javascript:fonction();"> ... </a>`
– lors du survol d'un objet (**onMouseover**) :
`<img src="image.gif" onMouseover="javascript:fonction();" />`

3.3.6 Comment différer ou planifier l'appel des fonctions javascript ?

– appel d'une fonction JavaScript après n millisecondes :
`setTimeout ("fonction_javascript()", n_millisecondes);`
– appel d'une fonction JavaScript toutes les n millisecondes :
`setInterval ("fonction-javascript()", n_millisecondes);`

3.3.7 Comment créer une fenêtre standard de saisie ou une fenêtre d'alerte ?

Voici un exemple qui demande une chaîne de caractères dans une première fenêtre et l'affiche dans une seconde :

```
var nom;
nom=prompt("Your name please", "");  // fenêtre standard de saisie
                                   // (le second paramètre correspond à une valeur par défaut)
alert("Hello "+nom+" !");      // fenêtre d'alerte (+ permet la concaténation de chaines)
```

3.3.8 Comment créer une nouvelle fenêtre ?

→ en utilisant la méthode **open** sur l'objet **window**.

```
function creerFenetre()
{
  f = window.open ("page.html",           // url du document web à charger
       "fenetre",             // nom de cette fenêtre
       "width=500,height=500, // largeur et hauteur de la fenêtre
       scrollbars=no,         // pas d'ascenceurs puis suppression des différentes barres
       locationbar=no, menubar=no, personalbar=no, statusbar=no, toolbar=no,
       location=no, directories=no, status=no");
}
```

3.3.9 Comment agir sur une fenêtre mère à partir de la fenêtre fille ?

→ en utilisant la référence **opener** :
dans la fenêtre fille nous voulons charger une URL sélectionnée dans la seconde :
`window.opener.location.href=...;`

3.3.10 Comment vérifier que tous les champs d'un formulaire ont bien été saisis ?

(Une autre solution plus judicieuse sera présentée dans le second support de cours).

```
function fonctionVerification (formulaire) {
   if (formulaire.champ1 == "" || formulaire.champ2 == "" || ...) {
      alert ("Veuillez remplir tous les champs !");
      return false;
   }
   return true;
}

...
<form onSubmit="javascript:return fonctionVerification(this);"> ... </form>
```

3.4 JavaScript/DOM/DHTML :

La représentation interne de la page web, (initialement déterminée par les balises HTML et les styles CSS associés), est accessible via une bibliothèque de fonctions normalisées, intégrée au sein d'un langage de script.
- Utiliser Javascript pour accéder à cette représentation interne et modifier dynamiquement le contenu de la page web s'appelle le **Dynamic HTML (DHTML)**
- Cette normalisation est appelé le **DOM (Document Object Model)**

3.4.1 Méthodes de l'API DOM de Javascript permettant d'accéder à l'arbre DOM :

Accès à un ou plusieurs éléments par leurs id, noms ou types de balises :
- **document.getElementById()** : l'identifiant id est unique;
- **document.getElementsByName()** : plusieurs éléments peuvent avoir le même nom;
- **document.getElementsByTagName()** : ensemble de balises.

Accès à un ou plusieurs nœuds/attributs à partir de leur nœud père :
- Accès à la valeur d'un attribut : `valeur = noeud.getAttribute("nom_attribut")`
- Test si un nœud a des nœuds fils : `noeud.hasChildNodes()`
- Accès au nombre de nœuds fils : `noeud.childNodes.length`
- Accès à un nœud fils particulier : `noeud.childNodes.item(num)`

Création de nœuds (élément, texte ou attribut) dans l'arbre DOM :
- Création d'un élément : `noeud = document.createElement(balise)`
- Création d'un nœud texte : `feuille = document.createTextNode(chaîne)`
- Création d'un attribut : `noeud.setAttribute(nom, valeur)`

Création de nœuds/attributs dans l'arbre DOM :
- Ajout d'un nœud : `noeud_pere.appendChild(noeud_fils)`

Cela signifie donc que pour faire apparaître un nouveau composant dans la page HTML, il faut :
- **Créer le nœud (la zone mémoire);**
- **Puis l'insérer dans l'arbre DOM (par appendChild(...)).**

Exemple d'un petite explorateur DOM rudimentaire (mais émouvant) :

```
<html>
  <head>
     <script>

       function affichageDOM(noeud) {
          if (noeud == null) {alert("Le noeud est vide !"); return;}
          var type = 0;
          switch (noeud.nodeType) {
              case 1 : type = "Element"; break;
              ...
          }
          alert(noeud.nodeType+":"+noeud.nodeName+"/"+type+"/"+noeud.nodeValue);
          if (noeud.hasChildNodes()) {
              for (var num=0; num < noeud.childNodes.length; num++) {
                  affichageDOM(noeud.childNodes.item(num));
              }
          }
       }
```

```
    </script>
  </head>

  <body onLoad="affichageDOM(window.document)">
    Cela vous aurez rassuré de voir du code HTML ici, avouez-le.
    En fait, c'est la place des balises HTML à explorer...
  </body>
</html>
```

3.4.2 La gestion des divisions :

Le déplacement ou l'apparition/disparition d'objets dans un document html est souvent géré par la création de divisions définies par les balises <div> et </div>.
La manipulation de divisions permet notamment la création de gadgets graphiques (menus déroulants, contextuels...) qui sont standardisés dans les frameworks Javascript.

Voici les principales opérations possibles sur les divisions :
- pour créer une division : <div class="maClasseDeStyle" id="maDivision"> ... </div>
- pour connaître la position d'une division :
  tester l'abscisse et/ou l'ordonnée du coin supérieur gauche de celle-ci :
  `if (document.maDivision.left > ...) {...}`
  (pour tester l'ordonnée, remplacez **left** par **top**)
- pour déplacer une division : incrémenter ou décrémenter les valeurs des propriétés `left` ou `top`
- pour faire apparaître/disparaître une division :
  changer la valeur de la propriété **visibility**

Attention, les divisions n'étaient pas gérées naguère pareillement avec Netscape ou Internet Explorer/Mozilla :
Adressage d'une propriété d'une division de nom "nomDivision" :

Avec Internet Explorer :	window.document.all["nomDivision"].style.propriété
ou Mozilla	ou window.document.nomDivision.style.propriété
Avec Netscape :	window.document.layers["nomDivision"].propriété
	ou window.document.nomDivision.propriété

Voici les principales propriétés des divisions dans les "anciens" navigateurs :

	avec CSS	avec Internet Explorer	avec Netscape	avec Mozilla/Firefox/IE 7
abscisse	left : ...px	style.pixelLeft=...	left=...	style.left
ordonnée	top : ...px	style.pixelTop=...	top=...	style.top=...
visibilité	visibility : visible	style.visibility="visible"	visibility="visible"	style.visibility="visible"
	visibility : hidden	style.visibility="hidden"	visibility="hidden"	style.visibility="hidden"

Pour la portabilité, utilisez maintenant **getElementById** pour référencer une division dans les différents D.O.M.. : `document.getElementById(id_division).style.propriété`

3.5 XML (eXtensible Markup Language) :

Le langage HTML a le grave inconvénient de méler la sémantique des documents web avec leur présentation. Pour remédier à cela, le langage XML ne représentant que la sémantique des documents a été créé. Les documents XML peuvent être alors médiatisées par le client de manière très sophistiquée par des feuilles de style XSL (présentées ultérieurement dans votre cursus), ou simplement par des feuilles de styles CSS.

Un document XML débute par une ligne d'entête qui précise la version et l'encodage employés :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

`ISO-8859-1` appelé également "latin1" permet d'encoder le document avec les caractères romans

Voici un exemple d'un document XML :

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<recette>
   <ingrédient type="viande"> du poulet mal cuit </ingrédient>
   <ingrédient type="légumes"> beaucoup d'oignons très très verts </ingrédient>
   <ingrédient type="condiments"> des épices orientales et intrigantes </ingrédient>
</recette>
```

3.5.1 XML et les CSS :

Voici un exemple de document XML qui sera mis en page par la feuille de style CSS ci-après

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<?xml-stylesheet href="champignons.css" type="text/css" ?>

<révisions_d_automne matière="champignons">

  <famille nom="bolet">
    Les bolets sont des champignons à tubes (certes ils ne sont pas les seuls...)
    <dangerosité>Aucun bolet n'est mortel mais quelques uns peuvent être indigestes</dangerosité
    <espèce class="comestible"> cèpe de bordeaux </espèce>
    <espèce class="comestible"> cèpe des pins </espèce>
    <espèce class="toxique"> bolet satan </espèce>
  </famille>

  <famille nom="amanite">
   Les amanites sont des champignons à lamelles, à anneaux et à bulbes
   <dangerosité> Du meilleur au pire </dangerosité>
     <espèce class="comestible"> amanite des Césars </espèce>
     <espèce class="toxique"> amanite phalloïde </espèce>
  </famille>

</révisions_d_automne>
```

Voici la feuille de style CSS de présentation du document XML ci-dessus :

```
famille
{
    display: block;
    margin-top: 5px; margin-left: 1cm;
    background-color: white; color: black;
    font-family: serif; font-size: 14pt; font-weight: bold; font-style: normal;
```

```
}

famille dangerosité
{
   display: block;
   margin-left: 2cm;
   color: orange;
   font-family: serif; font-size: 12pt; font-style: italic;
   text-decoration: underline;
}

espèce
{
   display: block;
   margin-left: 3cm;
   font-family: times; font-size: 14pt; font-style: oblique;
}

espèce.toxique { text-decoration: underline; color: red; }
espèce.comestible { color: green; }
```

→ pour les èlèments *espèce* dont la valeur de l'attribut *class* est *toxique*
Et le résultat (évidemment sans la couleur c'est moins bien :)

→ **Les sélecteurs de classe avec XML ne fonctionnent pas sous les vieilles versions de Firefox !**

3.5.2 XHTML

Voici les principales règles de syntaxe qu'un document XHTML doit respecter :
- les balises et les noms des attributs doivent uniquement être écrits en minuscules
- à toute balise ouvrante `<balise>` doit correspondre une balise fermante `</balise>`
- une balise unique doit être écrite comme suit : `<balise/>` (exemple `<hr/>`)
- les attributs des balises doivent respecter la syntaxe suivante : `nom_attribut="valeur_attribut"`

Index

AJAX, 3

CGI PERL ENV, 6
CGI PYTHON FieldStorage(), 6
clientHeight, 8
clientWidth, 8
CSS A :, 7
CSS A :active, 7
CSS A :hover, 7
CSS A :link, 7
CSS A :visited, 7
CSS background-color, 7
CSS balise fille, 7
CSS classe de style, 7
CSS color, 7
CSS display, 7
CSS filiation de balises¿, 7
CSS font-family, 7
CSS font-size, 7
CSS font-weight, 7
CSS groupe de balises, 7
CSS link, 7
CSS list-style-image, 7
CSS list-style-type, 7
CSS margin-..., 7
CSS P :, 7
CSS pseudo-élément sur P :, 7
CSS pseudo-classe A :, 7
CSS text-align, 7

DHTML définition, 11
DHTML div, 12
DHTML propriété left, 12
DHTML propriété top, 12
DHTML visibility, 12
DOM appendChild(...), 11
DOM childNodes.item(...), 11
DOM childNodes.length, 11
DOM createElement(...), 11
DOM createTextNode(...), 11
DOM définition, 11
DOM getAttribute(), 11
DOM getElementById(), 11
DOM getElementsByName(), 11
DOM getElementsByTagName(), 11
DOM hasChildNodes(), 11
DOM screen.availHeight, 8
DOM screen.availWidth, 8
DOM screen.height, 8
DOM screen.width, 8
DOM setAttribute(...), 11
DOM window.document.anchors, 8
DOM window.document.applets, 8
DOM window.document.forms, 8
DOM window.document.images, 8
DOM window.document.links, 8
DOM window.frames, 8
DOM window.history, 8
DOM window.location, 8
DOM window.name, 8
DOM window.navigator, 8
DOM window.opener, 8
DOM window.parent, 8

Feuilles de style CSS, 7

html a externe, 4
html a interne, 4
html attribut background, 4
html attribut bgcolor, 4
html attribut color, 4
html b, 4
html body, 4
html button, 6
html couleurs, 4
html font, 4
html form, 6
html h., 4
html head, 4
html i, 4
html images, 4
html img, 4
html input, 6
html li, 5
html listes, 5
html meta, 4
html ol, 5
html table, 5
html tableaux, 5
html td, 5
html title, 4
html tr, 5
html u, 4
html ul, 5

Javascript ¡script¿, 9
Javascript alert(), 10
Javascript Commentaires, 9
Javascript Fonctions, 9
Javascript navigator.appName, 9

Javascript onChange, 9
Javascript onClick, 9
Javascript onLoad, 9
Javascript onMouseover, 10
Javascript onResize, 9
Javascript onSubmit, 9
Javascript onUnLoad, 9
Javascript open(...), 10
Javascript opener.location, 10
Javascript prompt(), 10
Javascript setInterval(...), 10
Javascript setTimeout(...), 10
Javascript Test, 10

PHP GET, 6
PHP POST, 6

XML, 13
XML entête, 13
XML exemple, 13
XML xml-stylesheet, 7