



# Cours Internet





D .Mailliet

Cours Internet ..... 1
Cours Internet ..... 7
I. Introduction ..... 7
A. Logiciels utilisés ..... 7
1. Editeur de Textes ..... 7
2. Navigateur ..... 7
3. Serveur http ..... 7
4. Interprète Php ..... 7
5. Serveur de base de données MySQL ..... 7
6. Installation des logiciels ..... 7
a) Sous Windows ..... 7
b) Sous Linux ..... 7
B. Objectifs du cours : quelques « langages » pour Internet ..... 7
1. Seul le client est utile ..... 8
L' HyperText Markup Language ..... 8
a) Les feuilles de style ..... 8
b) Le javascript ..... 8
c) Les applets java ..... 8
2. serveur(s) nécessaire(s) ..... 8
a) Le Php ..... 8
b) Php - MySql ..... 8
C. But de l'enseignement ..... 8
D. Clients et Serveurs ..... 8
1. Exemple ..... 8
2. Client seul. ..... 9
3. Client et serveur ..... 9
**Bonjour** ..... 10
4. Fonctionnement Résultat ..... 11
E. Javascript ou Php ? ..... 11
1. Exemple 1 ..... 11
2. Exemple 2 ..... 11
3. Exemple 3 ..... 11
II. Attention à la casse ..... 12
III. Html ..... 13
A. Structure d'un document HTML ..... 13
B. Le laxisme du HTML ..... 14
C. Présentation d'un document HTML ..... 14
D. Indentation ou pas ..... 15
E. Liste des balises et des attributs ..... 15
F. Du style dans le texte ..... 33
1. Les titres ..... 33
**Mon titre en H1** ..... 33
**Mon titre en H2** ..... 33
**Mon titre en H3** ..... 33
**Mon titre en H4** ..... 33
2. Les différents styles de caractères ..... 33
**Voici du gras** ..... 33
*Voici de l'italique* ..... 33
`voici du texte en police monospace` ..... 33
voici du texte souligné ..... 33
***Voici un exemple de texte*** **partiellement en italique, en gras** et souligné ..... 33
3. Styles logiques ..... 33
*Mise en évidence simple* ..... 34
**Plus d'emphase** ..... 34
*Pour faire une citation* ..... 34
`Le résultat d'un programme` ..... 34

*ceci est une variable*....34
`Du code`....34
`Une touche du clavier`....34
G. Formater le texte....34
1. Les retours à la ligne : <br> ou <p>....34
avec un retour à la ligne et une ligne vierge....34
2. Aligner le texte....34
Votre paragraphe est au centre....34
3. Mettre du texte en retrait :....34
4. Le texte préformaté :....35
5. Césure ou pas de césure....35
6. Les traits de séparation :....35
7. Les caractères spéciaux....35
8. Indice et exposant....36
H. Afficher des images....36
1. Les formats de fichiers graphiques....36
2. Affichage simple d'une image....36
3. Les attributs de <IMG>....37
4. Une image pour le fond de votre page....37
I. Les Liens HyperTextes....37
1. Créer un lien hypertexte - <A>....37
2. Utiliser les ancres - <A name="..."> ....38
a) Insérer une ancre....38
b) Pointer vers une ancre....38
J. Les Listes....38
1. Listes à puces et listes numérotées....38
a) Les listes à puces - <UL>....38
b) Les listes numérotées - <OL>....38
1. Mon premier élément de liste....39
2. Mon deuxième élément de liste....39
2. Les listes descriptives - <DL>....39
3. Les listes <DIR> et <MENU>....39
4. Mêler les listes....40
K. Couleurs et Polices....40
1. Définition des couleurs en HMTL....40
a) Les couleurs standard....40
b) Les codes hexadécimaux....40
2. Les éléments de la page....40
3. Les polices de caractères : La balise <FONT>....41
a) Changer la taille par défaut....41
L. Les Tableaux....41
1. Définir un tableau - <TABLE>....42
2. Les cellules d'un tableau....42
a) Les titres....42
b) Les cellules....43
3. Exemple de code pour un tableau....43
M. Les frames....44
1. Le fichier <FRAMESET>....44
2. Les attributs pour le fichier <FRAMESET>....44
3. Des fichiers HTML et des frames....45
4. Exemple de code pour un fichier frameset....46
N. Formulaires de saisie....46
IV. Javascript....49
A. Généralités....49
B. Erreur dans un javascript....49
1. Erreur simple....49
2. ErreurGrave....49

C. Commentaires ..... 49
D. Constantes, Variables, types ..... 49
1. Variables ..... 50
**Le nom des variables est sensible à la casse (ie : x != X).** ..... 50
2. Constantes ..... 50
3. La valeur sans valeur ..... 50
4. opérateurs divers ..... 50
a) Opérateurs d'assignement : ..... 50
b) Opérateurs de comparaison : ..... 51
c) Opérateurs 'affectation conditionnelle ..... 51
E. Types ..... 51
1. Généralités ..... 51
a) Syntaxe objet ..... 51
b) Syntaxe Classe ..... 51
c) Syntaxe impérative ..... 51
2. Booléens ..... 51
a) Opérateurs logiques : ..... 52
3. Les nombres ..... 52
a) Les entiers: ..... 52
b) Les nombres décimaux: ..... 52
c) Les opérateurs ..... 52
d) Les constantes mathématiques ..... 52
e) Les fonctions ..... 52
4. Les chaines: ..... 52
a) L'opérateur de Concaténation ..... 53
b) Les fonctions ..... 53
5. Les tableaux: ..... 53
a) Tableaux simples ..... 53
b) Tableaux associatifs ..... 54
c) Tableaux multidimentionnels ..... 54
d) fonctions ..... 54
6. Les dates: ..... 56
a) Fonctions ..... 56
F. Les structures de contrôle ..... 56
1. Conditionnelle ..... 56
a) Le Si ..... 56
b) Le Cas ..... 57
2. Boucles ..... 57
a) Boucle pour ..... 57
b) Pour dans ..... 58
c) Boucle tant que ..... 58
3. Rupture et continuitéde boucle ..... 58
G. Les fonctions ..... 59
1. Passage des arguments par valeur ..... 59
2. Passage des arguments par adresse (par variable) ..... 59
3. Arité variable ..... 59
H. Les procédures ..... 60
I. Portée des variables ..... 60
J. Classes et Objets ..... 61
K. Entrées/Sorties ..... 61
1. sorties ..... 61
2. entrées ..... 62
L. Fichiers ..... 62
M. Inclusion de fichiers ..... 62
N. Evénements : traitement ..... 62
O. Evénements : simulation ..... 63
***Blur() ; focus() ; click(); submit();*** etc... ..... 63

Dans l'exemple suivant on simule l'action de l'utilisateur qui consiste à placer le focus sur le champ mdp ..... 63
Dans les 2 exemples suivants on envoie automatiquement le formulaire ci-dessus ..... 64
P. Objets prédéfinis ..... 64
1. Exemple très important : ..... 64
2. Les tableaux prédéfinis ..... 65
3. Exemple : communication entre frames ..... 66
4. Exercice : ..... 67
V. Feuilles de style ..... 69
A. La création de style ..... 69
B. Les feuilles de style externes ..... 70
1. Le fichier externe ..... 70
2. L'intégration du fichier externe ..... 70
C. Les propriétés de style ..... 70
1. Les propriétés de police ..... 70
2. Les propriétés de texte ..... 71
3. Les propriétés d'arrière-plan ..... 71
4. Les propriétés de positionnement ..... 71
5. Les propriétés d'encadrement ..... 71
6. Exemples ..... 72
VI. Applet java ..... 74
A. Généralités ..... 74
B. Syntaxe ..... 74
C. Passage de paramètres ..... 75
VII. Php ..... 76
A. Généralités ..... 76
B. Commentaires ..... 77
C. Constantes, Variables, types ..... 77
1. variables ..... 77
**Le nom des variables est sensible à la casse (ie : $x != $X).** ..... 77
2. Nommage dynamique des variables ..... 78
3. Constantes ..... 79
4. Références ..... 79
5. La valeur sans valeur ..... 79
6. opérateurs divers ..... 79
a) Opérateurs d'assignement : ..... 79
b) Opérateurs de comparaison : ..... 79
c) Opérateurs 'affectation conditionnelle ..... 79
7. Variables prédéfinies ..... 80
D. Types ..... 80
1. Booléens ..... 80
a) Opérateurs logiques : ..... 80
2. Les nombres ..... 80
a) Les entiers: ..... 80
b) Les nombres décimaux: ..... 81
c) Les opérateurs ..... 81
d) Les constantes mathématiques ..... 81
e) Les fonctions ..... 81
3. Les chaines: ..... 83
a) L'opérateur de Concaténation ..... 83
b) Les fonctions ..... 83
4. Les tableaux: ..... 86
a) Tableaux simples ..... 86
b) Tableaux associatifs ..... 86
c) Tableaux multidimentionnels ..... 86
d) fonctions ..... 87
e) Exemple ..... 88
E. Les structures de contrôle ..... 89

1. Conditionnelle ..... 89
a) Le Si ..... 89
b) Le Cas ..... 90
2. Boucles ..... 90
a) Boucle pour ..... 90
b) Boucle pour chaque ..... 91
c) Boucle tant que ..... 92
d) Boucle Faire tant que ..... 92
3. Rupture et continuité de boucle ..... 93
F. Les fonctions ..... 93
1. Passage des arguments se fait par valeur ..... 93
2. Passage des arguments se fait par adresse (par variable) ..... 93
G. Les procédures ..... 94
H. Portée des variables ..... 94
I. Classes et Objets ..... 94
J. Entrées/Sorties ..... 95
1. sorties ..... 95
2. entrées ..... 95
K. Fichiers ..... 96
Et pour la lecture ..... 96
L. Inclusion de fichiers ..... 97
VIII. PhpMyAdmin ..... 98
On voit ci-dessus les différentes bases que l'on peut développer en cliquant sur le + et découvrir les tables qui la compose ..... 98
CREATE DATABASE `calrep`; ..... 101
Soit par un copier/coller des lignes de ***ess.sql*** ..... 101
IX. Interfaçage PhP/MySql ..... 102

# Cours Internet

# I. Introduction

## A. Logiciels utilisés

### 1. Editeur de Textes

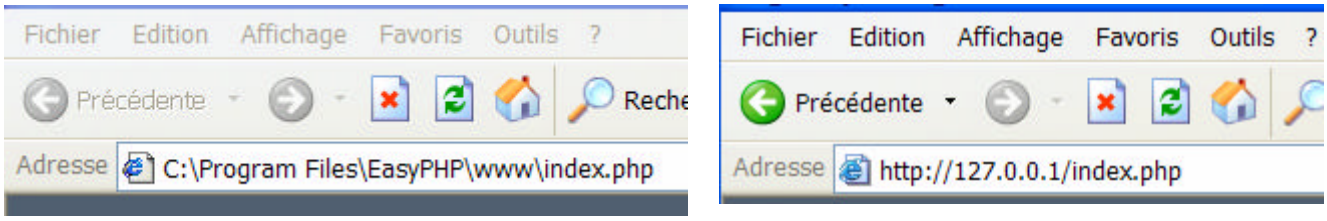
Votre éditeur de textes (ASCII) préféré fera l'affaire. Il est possible d'utiliser Notepad sous Windows, VI ou Kedit sous Linux. Mais les Editeurs plus évolués comme (X)Emacs ou Pfe ou scriptEdit ou encore EditPlus faciliteront la tache. Les éditeurs intégrés aux navigateurs seront aussi utilisés sans en abuser. Il permettra d'écrire les codes dans les différents langages

### 2. Navigateur

Internet Explorer, Netscape, Mozilla sont les plus utilisés. Ils ne réagissent pas tous exactement de la même façon. Internet Explorer n'existe que sous windows. Netscape et mozilla existent sous Windows et linux, Ils ne réagissent pas non plus exactement de la même façon sous ces 2 systèmes d'exploitation. Il conviendra donc de tester les pages créées sous les 2 systèmes d'exploitation et avec les différents navigateurs.

### 3. Serveur http

C'est lui qui va fournir les pages html demandées. Nous utiliserons Apache. Le rôle essentiel du serveur Http va être de fournir une arborescence pour Internet différente de l'arborescence « physique ». Par exemple sous Windows, avec easyphp installé en standard les pages http et php se trouvent en « C:\Program Files\EasyPHP\www »




A gauche : sans utiliser le serveur http. A droite : en utilisant le serveur http sur la machine locale (127.0.0.1 est l'adresse IP de la machine locale, mais cette IP peut être remplacée par celle d'une machine distante équipée d'un serveur http).

On remarque que la racine pour le serveur correspond physiquement à « C:\Program Files\EasyPHP\www » pour la machine. Et qu'il est possible de descendre dans l'arborescence sous WWW, mais qu'il n'est pas possible d'y remonter.

### 4. Interprète Php

Il fonctionne en collaboration avec le serveur http pour traduire le code Php en Html

### 5. Serveur de base de données MySQL

Il travaille en collaboration avec l'interprète Php et fournit les données extraites des tables.

### 6. Installation des logiciels

#### *a) Sous Windows*

Un ensemble easyphp que l'on peut télécharger sur www.easyphp.org s'installe très facilement et contient les 2 serveurs et l'interprète.

#### *b) Sous Linux*

C'est très simple : il suffit de valider les installations des logiciels ci-dessus, ils font partie de toutes les (bonnes) distributions

## B. Objectifs du cours : quelques « langages » pour Internet

Nous distinguerons 2 grandes parties :

### 1. Seul le client est utile

Html, javascript, feuille de style et applet java sont interprétés par le navigateur (récent) il est inutile d'avoir installé apache, php, mysql.

#### *L' HyperText Markup Language*

C'est un langage de mise en forme des pages Html. Il permet de définir l'apparence d'une page : sa présentation. Il permet entre autre de gérer les alignements de textes, le gras, l'italique, les titres, la confection de tableaux et de formulaires etc.… La seule interaction avec l'utilisateur est la gestion de liens hypertexte qui permettent de passer d'une page à une autre en cliquant sur un élément de la page. Le HTML décrit la présentation d'une page à l'aide de balises entre < >

#### *a) Les feuilles de style*

Elles permettent de donner une uniformité de présentation des différentes pages d'un même « site » et de donner de la souplesse à la modification de leur présentation.

Les styles peuvent être définis dans le fichier Html ou mieux dans un fichier séparé (.css). Dans un tel fichier, on définit par exemple qu'un titre de page est en bleu sur fond jaune avec une hauteur de 18 points, que les titres de niveau 1 sont en vert et ont une hauteur de 16 points etc. ... L'application de cette feuille de style aux différentes pages du site lui donnera une certaine homogénéité. De plus la modification dans la feuille de style : titre de page est en rouge sur fond vert avec une hauteur de 20 points provoquera la modification dans toutes les pages sans qu'il soit nécessaire de le faire manuellement dans chaque page.

#### *b) Le javascript*

Il permet de donner une réactivité aux pages, par exemple en animant certaines parties comme le changement d'une image lorsque la souris passe sur elle, contrôle de validité de parties de formulaire comme la vérification que l'utilisateur a bien rempli certaines zones obligatoires.

Le code javascript se trouve directement dans les pages html ou dans un fichier séparé (.js) et est directement lisible. Il se trouve encadré par 2 balises <script> </script>

#### *c) Les applets java*

Il s'agit ici de code java (différent de javascript) compilé (.class) : il a le même objectif que javascript , mais le code est (un peu) protégé.. Il est interprété lui aussi par le navigateur à condition que la JVM (machine virtuelle java) soit installée (sur le client)

### 2. serveur(s) nécessaire(s)

#### *a) Le Php*

C'est du code mélangé au code Html, il est interprété par le serveur et est traduit en code html puis envoyé au navigateur

#### *b) Php - MySql*

Des requêtes SQL sont interprétés par Php, envoyé au serveur de SGBD pour recevoir les données correspondantes, traduites en HTML et envoyées au navigateur.

## C. But de l'enseignement

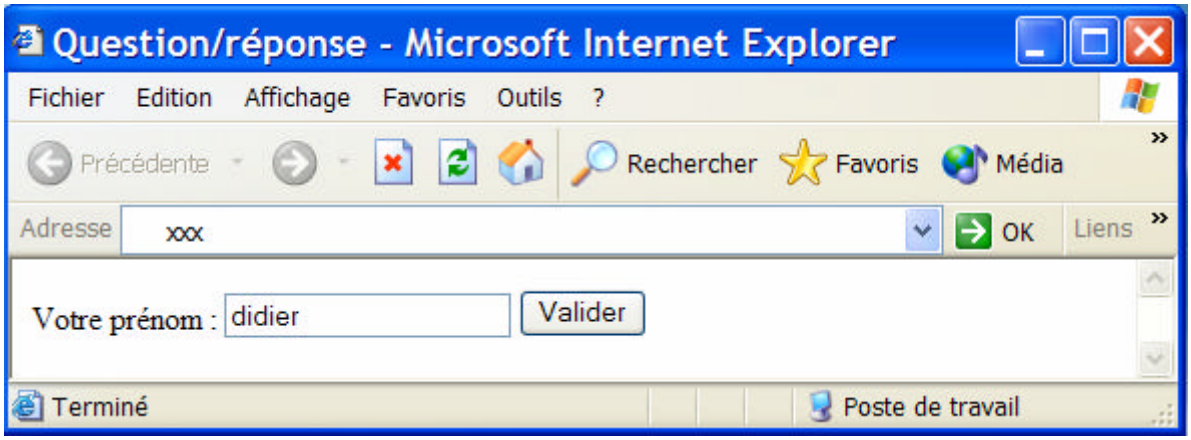
L'objet de cet enseignement n'est pas de faire de vous des experts mais de vous donner les bases pour le devenir. Nous ne verrons que les principales caractéristiques des différents langages et certainement pas toutes leurs finesses. Mais principalement l'utilité et le fonctionnement de chacun d'entre eux. Internet regorge de documentations sur le sujet. Ce polycopié est insuffisant pour traiter tous les TP et projets mais les différents sites Internet préconisés et d'autre que vous pourrez trouver pourront vous y aider.

## D. Clients et Serveurs

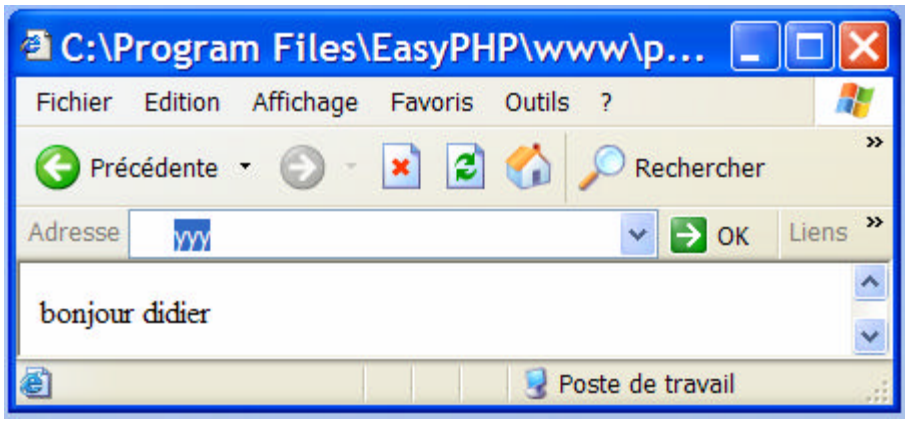
Le fonctionnement normal nécessite ***n+2*** machines : Les *n* clients, le serveur http et le serveur MySql. Bien entendu, les systèmes d'exploitation étant multi-tâches, pour les tests, une seule machine peut faire fonctionner les 2 serveurs et le(s) client(s)

### 1. Exemple

On veut écrire un programme simple qui demande à l'utilisateur son prénom et qui affiche « bonjour » suivi du prénom donné : Il s'agit donc d'un programme interactif, Le Html seul ne suffit pas : nous avons le choix entre javascript et php nous allons voir les différences entre ces 2 possibilités. Voici donc le déroulement du fonctionnement




L'appui sur le bouton ***valider*** provoque l'affichage :




### 2. Client seul.

Seul le navigateur est primordial, le serveur est facultatif

xxx et yyy représentent soit : C:\Program Files\EasyPHP\www\pedago\ess1\tstjs.htm
soit : http://localhost/pedago/ess1/tstjs.htm (sous Windows localhost est équivalent à 127.0.0.1 – A vérifier sous Linux). Le source de « tstjs.htm » est écrit en javascript (texte en gras) et Html (le reste non gras) :

```
<html>
 <head>
  <TITLE>Question/réponse</TITLE>
 </head>
 <body>
  <form name="maforme">
   Votre prénom : <INPUT type="text"  name="leprenom" value="">
   <INPUT type="submit" value="Valider" onclick="javascript:document.write('bonjour'
                                          +document.maforme.leprenom.value);return true;">
  </form>
 </body>
</html>
```

### 3. Client et serveur

Le navigateur et le serveur Http/Php sont nécessaires

xxx représente http://localhost/pedago/ess1/tstjs.php et

yyy : http://localhost/pedago/ess1/suite.php?leprenom=didier

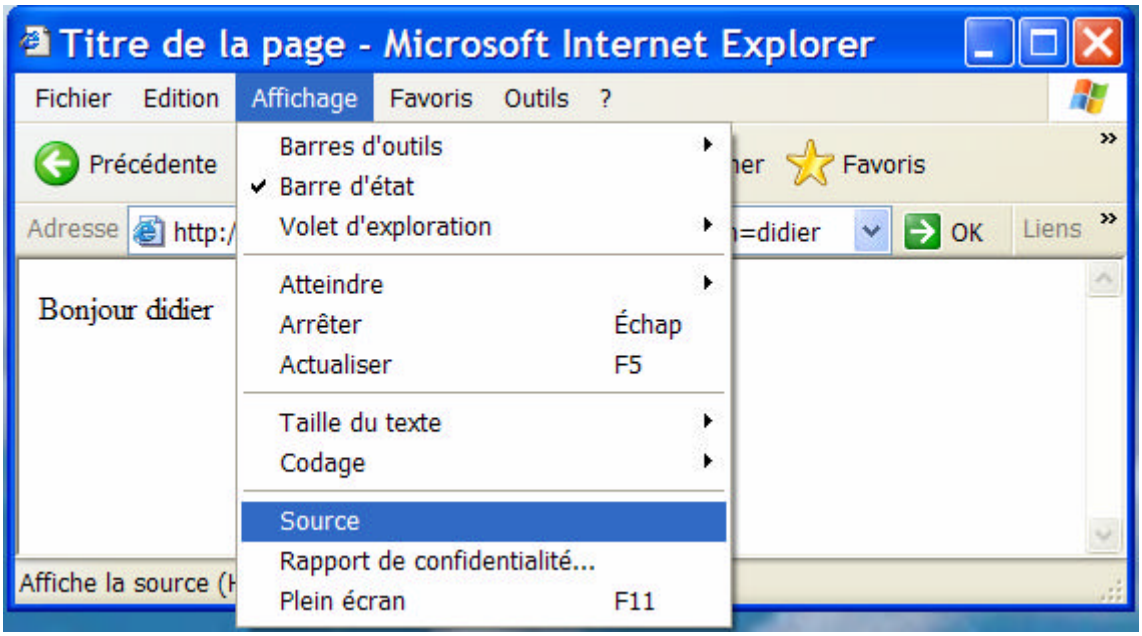
Il y a ici 2 fichiers différents: tstjs.php

```
<html>
 <head>
  <TITLE> Question/réponse </TITLE>
 </head>
 <body>
  <form name="maforme" methode="post" action="suite.php" target="_self">
   Votre prénom : <INPUT type="text" name="leprenom" value="">
   <INPUT type="submit" value="Valider" >
  </form>
 </body>
</html>
```

Et suite.php :

```
<html>
 <head>
  <TITLE> Question/réponse TITLE>
 </head>
 <body>
  Bonjour
  <?php
    echo $leprenom
  ?>
 </body>
</html>
```

Remarquons que si l'on demande l'affichage du source de « suite.php »




Nous ne verrons pas l'original (tel qu'il a été écrit –ci-dessus -) mais son interprétation par php :

```
<html>
 <head>
  <TITLE>Titre de la page</TITLE>
 </head>
 <body>
  Bonjour
  didier
 </body>
</html>
```

où la variable « $leprenom » a été remplacé par sa valeur « didier »

 Remarquons aussi:

- Que ***tstjs.php*** aurait pu s'appeler ***tstjs.htm(l)*** car aucun code php n'y est contenu
- La nécessité de 2 « programmes » : le premier est envoyé au navigateur par le serveur, il est chargé de « poser la question » à l'utilisateur et le second est chargé de mettre en forme la réponse (stockée dans ***leprenom***) de l'utilisateur, et qui transite ici par l'URL.
- Que si l'on met comme adresse ***C:\Program Files\EasyPHP\www\pedago\ess1\tstjs.php,*** c'est-à-dire que l'on ne passe pas par le serveur, les programmes fonctionneront (mal) mais n'interpréteront pas la « réponse » en mettant simplement « Bonjour » non suivi du prénom puisque c'est le serveur qui est chargé de l'interprétation de la variable ***leprenom***.

#### 4. Fonctionnement Résultat

Du point de vue de l'utilisateur, le fonctionnement de ***tstjs.htm*** et de ***tstjs.php/ suite .php*** semble identique et le résultat est le même. Il constatera une différence s'il s'en va examiner les sources. Cependant le fonctionnement est radicalement différent :

- Dans le premier cas : le client demande au serveur la page ***tstjs.htm*** (c'est l'utilisateur qui tape l'URL : ***http://.../tstjs.htm***) lorsque l'utilisateur appuie sur le bouton **valider** , c'est le navigateur qui récupère le texte tapé par l'utilisateur et affiche le résultat traité. Dans ce cas, il nous faut un navigateur évolué (capable de traiter des données et pas seulement de la mise en forme) et un serveur http minimal. Il y a une demande et une réponse.
- Dans le deuxième cas : le client demande au serveur la page ***tstjs.php*** (c'est l'utilisateur qui tape l'URL : ***http://.../tstjs.h***php) lorsque l'utilisateur appuie sur le bouton **valider** , le navigateur demande une nouvelle page au serveur ***suite .php*** en adjoignant à l'URL la donnée tapée par l'utilisateur et c'est le serveur qui la traite pour fournir une page Html. . Dans ce cas, il nous faut un navigateur minimal (capable de traiter de la mise en forme) et un serveur http évolué. Il y a 2 demandes et 2 réponses.

## E. Javascript ou Php ?

Qui fait le travail « intelligent » : le client ou le serveur ?

Si c'est le serveur, il faut en général dissocier les parties puisqu'il y a un aller entre le client (navigateur) qui reçoit les instructions de l'utilisateur et le serveur qui les traite et un retour vers le client qui les affiche et selon la complexité des échanges, cette « navette » peut se reproduire. Alors que si c'est le client, toutes les informations lui sont envoyées et c'est lui qui « choisit » de les afficher (selon les demandes).

Imaginons 3 exemples pour clarifier les choses (on ne parlera que de javascript et php, bien entendu, il comporteront tous du Html qui est l'essence même du fonctionnement http et pourquoi pas des feuilles de style) :

#### 1. Exemple 1

On demande à l'utilisateur de remplir un formulaire (comme celui ci-dessus, mais plus long que d'entrer le seul prénom) et on veut obliger celui-ci à remplir certains champs obligatoires, que la date de naissance est bien valide, que le code postal con tien bien 5 chiffres, etc.

Il est possible de faire ces vérifications avant que ce formulaire soit envoyé ; donc par le navigateur (en javascript par exemple) évitant ainsi d'envoyer un formulaire incorrect au serveur qui ferait la vérification, constaterait l'incorrection et le renverrait au client pour correction et ainsi de suite.

Dans ce cas on choisi javascript pour des questions de rapidité et d'encombrement des échanges

#### 2. Exemple 2

On veut réaliser un programme qui donne accès à certaines informations à l'utilisateur que s'il connaît un certain mot de passe. Si la réalisation est faite en javascript, et que l'utilisateur ne connaît pas ce mot de passe, des tests adaptés permettront de ne pas afficher dans le navigateur, mais il suffira d'examiner le source pour accéder aux informations puisque toutes auront été envoyées par le serveur le choix de les afficher incombant au client (navigateur). Dans ce cas, on choisira Php pour des raisons de sécurité.

#### 3. Exemple 3

On veut réaliser un programme qui permette à l'utilisateur d'entrer un code postal et d'obtenir le n om de la commune correspondante. Comme il y a en France 36 000 communes, si l'on choisit de coder en javascript, le serveur va devoir

envoyer les 36 000 noms et codes postaux afin que le client puisse effectuer la sélection. Dans ce cas, on choisira Php pour des raisons de volumes de données (d'encombrement); il sera même intéressant de faire appel à une requête Sql pour faciliter la recherche.

# II. Attention à la casse

J'attire seulement votre attention sur la notion de casse, c'est-à-dire le respect des majuscules et minuscules. En effet, lorsque vous vous trouvez sous système utilisateur (MacOs ou Windows), le fait que vous fassiez appel à un fichier nommé "Bonjour.html" par "bonjour.html" ne pose a priori pas de problème. Tant que vous êtes sous système utilisateur.

Mais sur Internet, la plupart des serveurs - là où seront stockées vos pages personnelles - fonctionnent sous UNIX ou dérivé. Or Unix est sensible à la casse, c'est-à-dire que si vous appelez le fichier "Bonjour.html" en tapant "bonjour.html", vous n'obtiendrez pas de réponse, ou un message d'erreur. Sous Unix, il s'agit de deux fichiers distincts.

En guise de conseil, essayez de n'utiliser que des minuscules dans vos noms de fichier (ou que des majuscules), ce qui vous permettra d'éviter ce risque d'erreur.Evitez de même les accents et les espaces dans les noms de fichiers

# III. Html

Le langage HTML (Hyper Text Markup Language) est un langage de balisage, c'est à dire qu'il est composé de ce qu'on appelle des balises, balises qui sont ensuite interprétées par le navigateur, celui-ci produit la mise en forme du document. C'est-à-dire, par exemple,Gras, italique, centré, formulaires, images, etc...

Voici un exemple de balise :

<head> xxxxx</ head >

Les balises sont écrites entre < et >. Elles vont généralement par paire, d'abord la balise ouvrante, marquant le début de l'action, et la balise fermante, marquant la fin de l'action. Celle ci est reconnaissable par le / placé juste après le <.

Certaines balises n'ont pas de balise de fermeture par exemple <br> qui permet le passage à la ligne, n'a aucune raison d'avoir une fermeture. Une bonne idée concite à adopter la syntaxe de XML, c'est-à-dire à mettre un antislash à la fin : <br />

Les balises peuvent avoir des attributs qui se place également à l'intérieur des < >, comme ceci :

<balise attribut1=" " attribut2=" "> xxx </ balise >

La valeur des attributs peut être avec ou sans guillemets simple ou double. On mettera toujours les valeurs entre guillemets, même si ça n'et pas obligatoire. De même, il est une habitude de mettre le nom des balises en majuscule. Si l'on veut une compatibilité XML, c'est à éviter. (à vérifier)

## A. Structure d'un document HTML

Dans un document HTML, certaines balises sont indispensables, elles forment la structure du document.

Html est très laxiste, Si certaines balises sont absentes ou non fermée ou même croisées, le document produit risque d'être correct. Mais il vaut mieux respecter les règles suivantes

La première balise est <HTML>. Elle ouvre tous les documents. </HTML> ferme le document.

La deuxième balise est <HEAD>. Théoriquement elle n'est pas obligatoire, une page HTML peut parfaitement fonctionner sans la balise HEAD. Mais en pratique elle se révèle indispensable. Elle contient la plupart des informations sur la page. Ces informations sont défini grâce à des sous balises de HEAD, de la manière suivante :

<HEAD>

< BALISE_de_TETE></ BALISE_de_TETE>

</HEAD>

Voici les principales balises (_de_tête) :

<TITLE> </TITLE> : Permet d'indiquer le titre de la page, celui qui s'affiche dans la page du navigateur .

<SCRIPT> </SCRIPT> : Permet de placer du script qui pourra être exécuter dans la page , Javascript ou VBscript par exemple.

<META> : Donne des indications sur la page qui pourront être récupérées par les robots des moteurs de recherche, et pourront aussi fournir des infos aux personnes qui liront votre code.

La troisième balise formant la structure d'un document HTML est la balise <BODY>.

C'est dans cette balise que va se placer tous le corps de la page html, c'est a dire tous son contenu. Tous le texte inscrit entre les balises <BODY> et </BODY> sera affiché à l'écran. Elle possède les attributs suivants:

| Attribbut | signification |
|---|---|
| BGCOLOR=une couleur | définit la couleur de fond du document, en RGB |
| BACKGROUND = l'URL d'une image | définit l'image de fond du document |
| TEXT = une couleur | définit la couleur du texte du document |
| LINK = une couleur | définit la couleur des liens du document |
| VLINK = une couleur | définit la couleur des liens visités du document |
| ALINK = une couleur | définit la couleur des liens actifs du document |

En résumé,voici un exemple dans lequel, il faut remplacer les … par du texte et les exemples de couleurs (#XXXXXX) par d'autres valeurs. :

```
<html>
        <head>
                <title>...</ title>
                < meta name ="author" content ="..." />
                < meta name ="keywords" content ="..." />
                < meta name ="lang" content ="..." />
                <meta name="description" content="..." />
                        ...
         </head>
         <body bgcolor="#FFFFFF" link="#FF3300" alink="#FFFFFF" vlink="#660099">
                ...
         </body>
</html>
```

## B. Le laxisme du HTML

Le balisage ouvrant/fermant en html correspond au parenthésage mathématique et aux structures de contrôle des langages informatique. Autrement dit , il ne faut pas croiser les balises. Il faut éviter d'oublier de fermer une balise ou de les fermer dans le désordre. Cependant une page mal construite peut très bien s'afficher correctement,


Mais la maintenance (mise à jour) peut s'avérer périlleuse. Il faut éviter à tout prix d'écrire par (contre-)exemple

```
<html>
<head>
<title>Maivais exemple
<body bgcolor="#FFFFFF" link="#FF3300" alink="#FFFFFF" vlink="#660099">
</head></ title>
Il ne faut surtout pas suivre cet exemple
</body>
</html>
```

## C. Présentation d'un document HTML

Comme dans les langages de programmation, les espaces, tabulations et passage à la ligne (même multiples) n'ont aucun effet sur la présentation. Que l'on mette 1 ou 20 espaces entre 2 mots, ils apparaîtront toujours séparés d'un seul, il en va de même si l'on passe 1 ligne entre 2 mots, ils apparaîtront séparés d'un espace.

Pour obtenir plusieurs espaces, il faut mettre plusieurs ***  ;*** pour obtenir plusieurs sauts de ligne, il faut mettre plusieurs ***<br />***

Il convient donc , comme en programmation, pour améliorer la clareté, de bien indenter les lignes de code.

## D. Indentation ou pas

 Remarque : Si l'indentation procure incontestablement une facilité de lecture du code, les multiples espaces « inutiles » prennent du temps de transmission sur le réseau.

Il n'est donc pas inutile de disposer de 2 versions des pages html une pour le développement et une pour l'utilisation du site. Il est facile d'écrire une « moulinette » dans un langage de programmation quelconque qui suprime les espaces, tabulation et sauts de ligne non obligatoires.

## E. Liste des balises et des attributs

Le tableau ci-dessous représente la liste des balises classée par ordre alphabétique. La balise se situe dans la colonne la plus à gauche. Et la description correspond à la première ligne d'entête du tableau. Lorsque la cellule la plus à gauche est vide, il s'agit d'une description des attributs de la balise située au dessus. Et la description correspond à la deuxième ligne d'entête du tableau

| Balise | Attribut | Fermeture | Description | Origine / |
|---|---|---|---|---|
| | Attributs | Valeur | Description | Exemple |
| on…() | représente dans la suite: | | Balise de script simple | HTML 2.0 |
| | onKeyDown | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est enfoncée) | "fonction()" |
| | onKeyPress | Appel une fonction | Balise de script (se produit lorsqu'on tape une touche du clavier) | "fonction()" |
| | onKeyUp | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est levée) | "fonction()" |
| | onClick | Appel une fonction | Balise de script (se produit lorsqu'on click) | "fonction()" |
| | onDblClick | Appel une fonction | Balise de script (se produit lorsqu'on double-click) | "fonction()" |
| | onHelp | Appel une fonction | Balise de script (se produit lorsqu'on appel l'aide F1) | "fonction()" |
| | onMouseDown | Appel une fonction | Balise de script (se produit lorsqu'un bouton de la souris est pressé) | "fonction()" |
| | onMouseMove | Appel une fonction | Balise de script (se produit lorsqu'on bouge la souris) | "fonction()" |
| | onMouseOut | Appel une fonction | Balise de script (se produit lorsqu'on déplace la souris sur le texte) | "fonction()" |
| | onMouseOver | Appel une fonction | Balise de script (se produit lorsqu'on déplace la souris hors le texte) | "fonction()" |
| | onMouseUp | Appel une fonction | Balise de script (se produit lorsqu'un bouton de | "fonction()" |
| <!DOCTYPE> | N | N | Version du HTML utilisé | HTML 2.0 |
| <!-- --> | N | N | Commentaire | HTML 2.0 |
| <ABBR> | | O | Abréviation | HTML 2.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction() |
| <ACRONYM> | | O | Acronyme (remplace <ABBR>) | HTML 4.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |

| | | | | |
|---|---|---|---|---|
| | style<br>on,,,() | Un style<br>une fonction | Attache un style spécifique à la balise<br>voir Balise de script simple | "font-size: 12 pt"<br>onClick="fonction()" |
| <ADDRESS> | N | O | URL sous forme textuel | HTML 2.0 |
| <A HREF> | | O | Lien hypertexte | HTML 2.0 |
| | Accesskey | Texte | C'est la touche de raccourci permettant d'accéder au lien (par la combinaison Alt + nom de la touche). | "b" |
| | href | URL | C'est l'URL vers où pointe le lien hypertexte | "page.html" |
| | Name | Nom | Nom (marque une position à l'intérieur du document) | "Nom" |
| | Target | Target (respectivement : même cadre, cadre père, nouvelle fenêtre, page entière) ou le nom du cadre de destination | Cadre de destination du lien | "_self" \| "_parent" \| "_blank" \|"top" \| "cadre" |
| | Tabindex | Nombre | C'est l'ordre de tabulation. 2 signifie que lorsque vous appuierez sur Tab, ce sera le deuxième élément sélectionné | "2" |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style<br>on,,,() | Un style<br>une fonction | Attache un style spécifique à la balise<br>voir Balise de script simple et complexes | "font-size: 12 pt"<br>onClick="fonction()" |
| <APPLET> | O | O | Insertion d'applet Java | HTML 3.0 |
| <AREA> | O | N | Zone cliquable | N.N 2.0 |
| <B> | N | O | Texte en gras | HTML 2.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style<br>on,,,() | Un style<br>une fonction | Attache un style spécifique à la balise<br>voir Balise de script simple | "font-size: 12 pt"<br>onClick="fonction()" |
| <BASE> | O | N | Déf. sup. de l'URL de la page HTML | HTML 2.0 |
| | href | URL | C'est l'URL vers où pointe le lien hypertexte | "page.html" |
| | Target | Target (respectivement : même cadre, cadre père, nouvelle fenêtre, page entière) ou le nom du cadre de destination | Cadre de destination du lien | "_self" \| "_parent" \| "_blank" \|"top" \| "cadre" |
| <BASEFONT> | O | N | Taille par défaut d'une police | N.N 1.0 |
| <BDO> | N | O | Changement de direction impossible | HTML 3.0 |
| <BGSOUND> | O | N | Insertion de fichier audio | * I.E 3.0 |
| <BIG> | N | O | Police de taille plus importante | HTML 3.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |

| | | | | |
|---|---|---|---|---|
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction() |
| <BLINK> | N | O | Clignotement du texte | * N.N 1.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction() |
| <BLOCKQUOTE> | N | O | Légère marge à gauche | HTML 2.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction() |
| <BODY> | O | F | Corps du document | HTML 2.0 |
| | Text | Couleur hexa RRVVBB ou nom de la couleur | Couleur du texte | "#000000" |
| | Link | Couleur | Couleur d'un lien | "#0000FF" |
| | VLink | Couleur | Couleur d'un lien visité | "#0099FF" |
| | ALink | Couleur | Couleur d'un lien actif | "#FF0000" |
| | background | Image | Image d'arrière plan | "fond.jpg" |
| | bgcolor | Couleur | Couleur de la page | "#FFFFFF" |
| | bgproperties | Valeur fixed | Arrière plan statique (qui ne défile pas avec le texte) | "fixed" |
| | BottomMargin | Nombre en pixels | Marge en bas de la page | "10" |
| | Scroll | Valeurs yes ou no | Permet ou non de faire défiler la page | "yes" \| "no" |
| | TopMargin | Nombre en pixels | Marge en haut de la page | "10" |
| | LeftMargin | Nombre en pixels | Marge à gauche de la page | "10" |
| | RightMargin | Nombre en pixels | Marge à droite de la page | "10" |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | oncontextmenu | Appel une fonction | Balise de script (se produit lors de l'apparition du menu (click bouton droit)) | "fonction()" |
| | ondragstart | Appel une fonction | Balise de script (se produit lorsqu'on effectue un glisser-déposer) | "fonction()" |
| | onselectstart | Appel une | Balise de script (se produit lorsqu'on | "fonction()" |

| | | | | |
|---|---|---|---|---|
| | | fonction | entame une sélection) | |
| | onUnload | Appel une fonction | Balise de script (se produit lorsque la feuille est fermée) | "fonction()" |
| | onScroll | Appel une fonction | Balise de script (se produit lorsqu'on scroll sur la feuille) | "fonction()" |
| | onResize | Appel une fonction | Balise de script (se produit lorsqu'on redimensionne la feuille) | "fonction()" |
| | onLoad | Appel une fonction | Balise de script (se produit lorsque la feuille est chargée) | "fonction()" |
| | onFocus | Appel une fonction | Balise de script (se produit lorsque la feuille reçoit le focus) | "fonction()" |
| | onBlur | Appel une fonction | Balise de script (se produit lorsque la feuille perd le focus). | "fonction()" |
| | onAfterUpdate | Appel une fonction | Balise de script | "fonction()" |
| | onBeforeUpdate | Appel une fonction | Balise de script | "fonction()" |
| | onKeyDown | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est enfoncée) | "fonction()" |
| | onKeyPress | Appel une fonction | Balise de script (se produit lorsqu'on tape une touche du clavier) | "fonction()" |
| | onKeyUp | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est levée) | "fonction()" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <BR> | O | N | Saut de ligne | HTML 2.0 |
| | Clear | Marges (gauche, droite, toutes) | Supprime les marges | "left" \| "right" \| "all" |
| <BUTTON> | N | O | Bouton dans un formulaire | * I.E 4.0 |
| <CAPTION> | O | O | Titre de tableau | HTML 3.0 |
| | Align | Alignement (respectivement : haut, bas, à gauche, centré, à droite) | Défini l'alignement de la légende | "top" \| "bottom" \| "left" \| "center" \| "right" |
| | VAlign | Alignement vertical (haut ou bas) | Définit l'alignement vertical de la légende | "top" \| "bottom" |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| <CENTER> | N | O | Centrage | HTML 3.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <CITE> | N | O | Texte sous forme de citation | HTML 2.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |

| | | | | |
|---|---|---|---|---|
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction() |
| <CODE> | N | O | Texte sous forme de code source | HTML 2.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction() |
| <COL> | O | N | Déf d'une colonne d'un tableau | * I.E 3.0 |
| | Width | Largeur | Largeur des colonnes définie pas la balise col | "100" \| "10%" |
| | Span | Nombre | Indique le nombre de colonnes sur lesquelles s'appliqueront les propriétés de la balise col | "2" |
| | Align | Alignement (respectivement : à gauche, centré, à droite, justifié, aligné sur un caractère spécifique) | Définit l'alignement de la colonne | "left" \| "center" \| "right" \| "justify" \| "char" |
| | char | Texte | Caractère sur lequel le texte s'alignera (si char est utilisé dans align) | "a" |
| | CharOff | Nombre | Nombre de pixels de retrait du texte | "10" |
| | VAlign | Alignement vertical (respectivement en haut, au milieu, en bas, ligne de base) | Définit l'alignement vertical de la colonne | "top" \| "middle" \| "bottom" \| "baselin |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| <COLGROUP> | N | F | Regroupement de colonnes (tableau) | * I.E 3.0 |
| | Width | Largeur | Largeur des colonnes définie pas la balise col | "100" \| "10%" |
| | Span | Nombre | Indique le nombre de colonnes sur lesquelles s'appliqueront les propriétés de la balise col | "2" |
| | Align | Alignement (respectivement : à gauche, centré, à droite, justifié, aligné sur un caractère spécifique) | Définit l'alignement de la colonne | "left" \| "center" \| "right" \| "justify" \| "char" |
| | char | Texte | Caractère sur lequel le texte s'alignera (si char est utilisé dans align) | "a" |
| | CharOff | Nombre | Nombre de pixels de retrait du texte | "10" |

| | | | | |
|---|---|---|---|---|
| | VAlign | Alignement vertical (respectivement en haut, au milieu, en bas, ligne de base) | Définit l'alignement vertical de la colonne | "top" \| "middle" \| "bottom" \| "baseline" |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| <COMMENT> | N | O | Commentaire | * I.E 3.0 |
| <DD> | N | F | Définition d'un terme | HTML 2.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <DEL> | N | O | Fourni des infos sur un document | HTML 2.0 |
| <DFN> | N | O | Texte sous forme logique | HTML 2.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <DIR> | N | O | Liste à plusieurs dimensions | HTML 2.0 |
| | name | Texte | Nom de la division | "Division" |
| | align | Alignement (gauche, centré, droite) | Alignement du texte | "left" \| "center" \| "right" |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | onAfterUpdate | Appel une fonction | Balise de script | "fonction()" |
| | onBeforeUpdate | Appel une fonction | Balise de script | "fonction()" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <DIV> | O | O | Structuration d'objets quelconques | HTML 3.0 |
| <DL> | O | O | Structure d'un glossaire | HTML 2.0 |
| | compact | | Indique qu'une liste doit être légèrement espacée pour être plus compacte | |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une | Attache une classe à la balise (un style | "titre1" |

| | | | | |
|---|---|---|---|---|
| | | classe | défini dans une feuille de style) | |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction() |
| <DT> | N | F | Élément d'un glossaire | HTML 2.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction() |
| <EM> | N | O | Emphase (italique) | HTML 2.0 |
| <EMBED> | O | N | Insertion d'applications multimédia | N.N 2.0 |
| <FIELDSET> | N | O | Regroupement de zones dans les formulaires | HTML 2.0 |
| <FONT> | O | O | Définition du texte | HTML 2.0 |
| | face | Police | Police de caractère utilisée pour le texte | "Arial" |
| | size | Chiffre de 1 à 7 pouvant être précédé de - ou + | Taille du texte. Plus la valeur sera grande plus la taille du texte sera élevée. Si vous utilisez un - ou un +, cela voudra dire que le texte devra être tant et tant plus grand ou plus petit par rapport à la taille de texte par défaut du navigateur (3 sur PC et 2 sur Macintosh) ; 7 est la taille maximale | "2" \| "-1" |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction() |
| <FORM> | O | O | Début du formulaire | HTML 2.0 |
| | onBlur | | Lorsque le contrôle perd le focus (vous cliquez à côté après l'avoir sélectionné). | |
| | onFocus | | Lorsque le contrôle reçoit le focus | |
| | onScroll | | Lorsque vous scrollez dans le contrôle | |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction() |
| <FRAME> | O | N | Structure d'un cadre (frame) | N.N 2.0 |
| <FRAMESET> | O | O | Fichier d'index pour les cadres (frame) | N.N 2.0 |
| <H1> | O | O | Titre de niveau 1 | HTML 2.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction() |
| <H2> | O | O | Titre de niveau 2 | HTML 2.0 |
| <H3> | O | O | Titre de niveau 3 | HTML 2.0 |
| <H4> | O | O | Titre de niveau 4 | HTML 2.0 |
| <H5> | O | O | Titre de niveau 5 | HTML 2.0 |
| <H6> | O | O | Titre de niveau 6 | HTML 2.0 |

| | | | | |
|---|---|---|---|---|
| | idem pour tous les Hi | | | |
| <HEAD> | N | F | En-tête d'un fichier HTML | HTML 2.0 |
| <HR> | O | N | Filet horizontal (séparation) | HTML 2.0 |
| | align | Alignement (gauche, centré, droite) | Alignement de la ligne | "left" \| "center" \| "right" |
| | color | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de la ligne | "#FF0000" \| "red" |
| | noshade | - | Pas d'ombrage (ligne pleine) | - |
| | size | Hauteur (pixels) | Hauteur de la ligne | "2" |
| | width | Largeur (pixels ou %) | Largeur de la ligne (par défaut 100%) | "100%" |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <HTML> | N | F | Structure externe d'une page HTML | HTML 2.0 |
| <I> | N | O | Texte en italique | HTML 2.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <IFRAME> | O | O | Définition d'un cadre local | * I.E 3.0 |
| | name | Nom | Le nom de la frame. | name="nom de la frame" |
| | target | Nom | la destination | target="target" |
| | scrolling | Dans l'ordre : automatique (les barres de défilement apparaissent si nécessaire), jamais de défilement, toujours une barre de défilement présente. | | scrolling="auto, no, yes" |
| | src | Le nom de la page qui se charge dans cette frame | | src="fichier" |
| | Heigh | Hauteur (pixels) | | Heigh=250 |
| | Width | Largeur | | width="100%" |
| <IMG> | O | N | Insertion d'image | HTML 2.0 |
| <INPUT> | O | N | Zone de saisie dans un formulaire | HTML 2.0 |
| <INS> | N | O | Fourni des infos sur un document | HTML 2.0 |
| <ISINDEX> | O | N | Déf. d'un fichier de recherche (script CGI) | HTML 2.0 |
| <KBD> | N | O | Texte de type machine à écrire | HTML 2.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <LABEL> | N | O | Extension de la balise FORM | * I.E 3.0 |

| | | | | |
|---|---|---|---|---|
| <LAYER> | O | O | Définition des couches (layers) sous Netscape | * N.N 4.0 |
| <LEGEND> | N | O | Légende de tableau | HTML 3.0 |
| <LI> | O | O | Élément d'une liste | HTML 2.0 |
| | type | Type de liste (Respectivement puce sous forme de rond, carré ou cercle) | Type de puce dans une liste de puces | "disc" \| "square" \| "circle" |
| | | Type de liste (Respectivement sous forme 1, a, A, i, I) | Type d'étiquette dans une liste d'étiquettes (numérique) | "1" \| "a" \| "A" \| "i" \| " |
| | value | Nombre | Attribut valable uniquement dans une liste numérique (change l'ordre de la liste) | "2" |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction() |
| <LINK> | O | N | Lien indépendant du média | HTML 2.0 |
| | href | URL | Définit la base de l'URL. Tous les liens de la page auront la même base (par exemple "http://www.serveur.com/". | "http://www.serveu com/" |
| | disabled | - | - | - |
| | media | Valeur, respectivement : à l'écran, pour l'imprimante, projection, braille, speech, tout | Identifie la méthode de présentation la mieux adaptée à la feuille de style | "screen" \| "print" \| "projection" \| "braille" \| "speech" "all" |
| | Rel | Texte | Indique la fin de la source d'un lien et en identifie le type | - |
| | Rev | Texte | Définit la fin de la destination d'un lien et en identifie le type | - |
| | Type | Texte | Définit le langage qui sera utilisé par la feuille de style pour créer les règles de styles | "text/css" |
| <MAP> | N | O | Structure externe de la zone cliquable | N.N 2.0 |
| <MARQUEE> | O | O | Texte défilant | *I.E 3.0 |
| | Align | Alignement (haut, milieu, bas) | Alignement du texte | "top" \| " middle" \| " bottom" |
| | Behavior | Comportement (défiler, glisser ou alterner) | Comportement du défilement | " scroll" \| " slide" \| " alternate" |
| | BgColor | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de fond (arrière plan du texte) | "#FF00FF" \| "red" |
| | Direction | Sens de défilement (gauche ou droite) | Sens de défilement | "left" \| "right" |
| | Height | Hauteur (chiffre ou %) | Hauteur du texte défilant | "100" \| "10%" |
| | HSpace | Marge (chiffre) | Spécifie une marge en largeur entre la | "10" |

| | | | | |
|---|---|---|---|---|
| | | | zone de texte défilant et les éléments autours | |
| | Loop | Nombre | Nombre de défilement (si non précisé, à l'infini) | "1" |
| | onAfterUpdate | Appel une fonction | Balise de script | "fonction()" |
| | onBeforeUpdate | Appel une fonction | Balise de script | "fonction()" |
| | onBounce | Appel une fonction | Balise de script | "fonction()" |
| | onFinish | Appel une fonction | Balise de script (se produit lorsque le défilement est finit) | "fonction()" |
| | onKeyDown | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est enfoncée) | "fonction()" |
| | onKeyPress | Appel une fonction | Balise de script (se produit lorsqu'on tape une touche du clavier) | "fonction()" |
| | onKeyUp | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est levée) | "fonction()" |
| | onStart | Appel une fonction | Balise de script (se produit lorsque le défilement commence) | "fonction()" |
| | ScrollAmount | Valeur défilement (chiffre) | Valeur du défilement : plus le nombre est grand, plus l'écart de défilement (distance effectuée en un déplacement du texte) sera grand (par défaut 6) | "7" |
| | ScrollDelay | Vitesse défilement (chiffre) | Vitesse de défilement : plus le nombre est grand, plus le temps entre chaque déplacement de texte sera important (par défaut 90) | "100" |
| | Truespeed | ? | ? | ? |
| | VSpace | Marge (chiffre) | Spécifie une marge en largeur entre la zone de texte défilant et les éléments autours | "10" |
| | Width | Largeur (chiffre ou %) | Largeur du texte défilant | "100" \| "10%" |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <MENU> | O | O | Liste de type menu | HTML 2.0 |
| | compact | - | Indique qu'une liste doit être légèrement espacée pour être plus compacte | - |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <META> | O | N | Informations sur la page HTML | HTML 2.0 |
| | name | Nom | Le nom détermine le type de balise méta (description, langage...) | "keywords" |
| | content | Texte | Ce que contient la balise (dépend du nom de celle-ci) | "meta, name, keywords, mots, clefs" |
| | HTTP-Equiv | Nom | Détermine le type de balise méta | "Content-Language" |
| <NEXTID> | O | N | Indique le document suivant | HTML 2.0 |

| | | | | |
|---|---|---|---|---|
| <NOBR> | N | O | Pas de passage à la ligne | N.N 1.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction() |
| <NOEMBED> | N | O | Contenu alternatif aux applications multimédia | N.N 2.0 |
| <NOFRAMES> | N | O | Contenu alternatif aux cadres (frame) | N.N 2.0 |
| <NOSCRIPT> | N | O | Contenu alternatif aux scripts | N.N 2.0 |
| <OBJECT> | O | O | Insertion d'objet (multimédia, applet, ...) | HTML 4.0 |
| <OL> | O | O | Liste ordonnée | HTML 2.0 |
| | compact | - | Indique qu'une liste doit être légèrement espacée pour être plus compacte | - |
| | start | Valeur | Indique la valeur à laquelle le numérotage de la liste doit commencer | "2" |
| | type | Type de liste (Respectivement sous forme 1, a, A, i, I) | Type d'étiquette dans une liste d'étiquettes (numérique) | "1" \| "a" \| "A" \| "i" \| " |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction() |
| <P> | N | F | Début d'un paragraphe (saut de 2 lignes) | HTML 2.0 |
| | align | Alignement (gauche, centré, droite) | Alignement du texte | "left" \| "center" \| "right" |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction() |
| <PARAM> | O | N | Paramètres d'un objet inséré | HTML 2.0 |
| <PRE> | N | O | Texte préformaté | HTML 2.0 |
| <Q> | O | O | Citations longues | HTML 4.0 |
| <S> | N | O | Texte barré | HTML 3.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction() |
| <SAMP> | N | O | Texte sous forme d'exemple | HTML 2.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |

| | | | | |
|---|---|---|---|---|
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <SCRIPT> | N | O | Insertion de script (JavaScript) | N.N 2.0 |
| <SELECT> | N | O | Définition d'une liste de formulaire | HTML 2.0 |
| <SMALL> | N | O | Police de taille moins importante | HTML 3.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <SPAN> | N | O | Structuration d'éléments dans une ligne | HTML 3.0 |
| <STRIKE> | N | O | Texte barré | HTML 2.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <STRONG> | N | O | Texte en gras | HTML 2.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <STYLE> | O | O | Définie les feuilles de style | HTML 3.0 |
| <SUB> | N | O | Texte sous forme d'indice | HTML 3.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <SUP> | N | O | Texte sous forme d'exposant | HTML 3.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <TABLE> | O | O | Structure d'un tableau | HTML 3.0 |
| | Rules | Respectivement : Pas de bordure, | Spécifie quel côté de la bordure intérieur du tableau doit être affiché | "none" \| "groups" \| "rows" \| "cols" \| "all" |

| | | | |
|---|---|---|---|
| | bordures horizontale entre tous les groupes de tableaux, bordures horizontale entre toutes les rangées du tableau, bordures partout | | |
| Name | Nom | Nom du tableau | "nom" |
| Frame | Respectivement : Pas de bordure externe, en haut, en bas, en haut et en bas, du côté gauche, du côté droit, côtés gauche et droit, encadrant le tableau, autour du tableau | Spécifie quel côté de la bordure extérieur du tableau doit être affiché | "above" \| "below" \| "hsides" \| "vsides" "LHS" \| "RHS" \| "box" \| "border" |
| CellSpacing | Nombre (en pixels) | Marge entre le bord de la cellule et son contenu | "2" |
| CellPadding | Nombre (en pixels) | Marge entre les bords extérieurs du tableau et les cellules qu'il contient | "2" |
| Border | Nombre (en pixels) | Définit la largeur de la bordure du tableau | "2" |
| BorderColor | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de la bordure du tableau | "#FFFFFF" |
| BorderColorDark | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de la bordure foncée du tableau | "#FFFFFF" |
| BorderColorLight | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de la bordure claire du tableau | "#FFFFFF" |
| background | Image | Image d'arrière plan du tableau | "fond.jpg" |
| bgcolor | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de fond du tableau | "#FFFFFF" |
| Cols | Nombre | Définit le nombre de colonnes du tableau | "3" |
| Height | Hauteur (en pixel ou %) | Hauteur du tableau | "100" \| "10%" |
| Width | Largeur (en pixel ou %) | Largeur du tableau | "100" \| "10%" |
| Align | Alignement (respectivement : à gauche, centré, à droite) | Définit l'alignement du tableau | "left" \| "center" \| "right" |
| onAfterUpdate | Appel une fonction | Balise de script | "fonction()" |
| onBeforeUpdate | Appel une fonction | Balise de script | "fonction()" |
| onKeyDown | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est enfoncée) | "fonction()" |
| onKeyPress | Appel une fonction | Balise de script (se produit lorsqu'on tape une touche du clavier) | "fonction()" |

| | | | | |
|---|---|---|---|---|
| | onKeyUp | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est levée) | "fonction()" |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <TBODY> | N | F | Regroupement de tableaux | I.E 3.0 |
| | BGColor | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de fond du tableau | "#FF0000" |
| | Align | Alignement (respectivement : haut, bas, à gauche, centré, à droite) | Définit l'alignement du tableau | "top" \| "bottom" \| "left" \| "center" \| "right" |
| | VAlign | Alignement vertical (haut ou bas) | Définit l'alignement vertical du tableau | "top" \| "bottom" |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | onKeyDown | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est enfoncée) | "fonction()" |
| | onKeyPress | Appel une fonction | Balise de script (se produit lorsqu'on tape une touche du clavier) | "fonction()" |
| | onKeyUp | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est levée) | "fonction()" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <TD> | N | F | Cellule d'un tableau | HTML 3.0 |
| | ColSpan | Nombre | Spécifie le nombre de colonnes recouvertes par une cellule | "2" |
| | NoWrap | Nombre | | "2" |
| | RowSpan | Nombre | Spécifie le nombre de lignes recouvertes par une cellule | "2" |
| | VAlign | Alignement (respectivement : en haut, au milieu, en bas, sur la ligne de base) | Alignement vertical du texte dans la cellule | "top" \| "middle" \| "bottom" \| "baseline" |
| | BorderColor | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de la bordure du tableau | "#FFFFFF" |
| | BorderColorDark | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de la bordure foncée du tableau | "#FFFFFF" |
| | BorderColorLight | Couleur hexadécimale RRVVBB ou nom | Couleur de la bordure claire du tableau | "#FFFFFF" |

| | | | | |
|---|---|---|---|---|
| | | de la couleur | | |
| | background | Image | Image d'arrière plan du tableau | "fond.jpg" |
| | bgcolor | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de fond du tableau | "#FFFFFF" |
| | Height | Hauteur (en pixel ou %) | Hauteur du tableau | "100" \| "10%" |
| | Width | Largeur (en pixel ou %) | Largeur du tableau | "100" \| "10%" |
| | Align | Alignement (respectivement : à gauche, centré, à droite) | Définit l'alignement du tableau | "left" \| "center" \| "right" |
| | onAfterUpdate | Appel une fonction | Balise de script | "fonction()" |
| | onBeforeUpdate | Appel une fonction | Balise de script | "fonction()" |
| | onKeyDown | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est enfoncée) | "fonction()" |
| | onKeyPress | Appel une fonction | Balise de script (se produit lorsqu'on tape une touche du clavier) | "fonction()" |
| | onKeyUp | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est levée) | "fonction()" |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction() |
| <TEXTAREA> | N | O | Zone de saisie à plusieurs lignes | HTML 3.0 |
| <TFOOT> | N | F | Regroupement de tableaux | I.E 3.0 |
| | BGColor | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de fond du tableau | "#FF0000" |
| | Align | Alignement (respectivement : haut, bas, à gauche, centré, à droite) | Définit l'alignement du tableau | "top" \| "bottom" \| "left" \| "center" \| "right" |
| | VAlign | Alignement vertical (haut ou bas) | Définit l'alignement vertical du tableau | "top" \| "bottom" |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | onKeyDown | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est enfoncée) | "fonction()" |
| | onKeyPress | Appel une fonction | Balise de script (se produit lorsqu'on tape une touche du clavier) | "fonction()" |
| | onKeyUp | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est levée) | "fonction()" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction() |

| | | | | |
|---|---|---|---|---|
| <TH> | N | F | Cellule d'en-tête d'un tableau | HTML 3.0 |
| | ColSpan | Nombre | Spécifie le nombre de colonnes recouvertes par une cellule | "2" |
| | NoWrap | Nombre | | "2" |
| | RowSpan | Nombre | Spécifie le nombre de lignes recouvertes par une cellule | "2" |
| | VAlign | Alignement (respectivement : en haut, au milieu, en bas, sur la ligne de base) | Alignement vertical du texte dans la cellule | "top" \| "middle" \| "bottom" \| "baseline" |
| | BorderColor | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de la bordure du tableau | "#FFFFFF" |
| | BorderColorDark | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de la bordure foncée du tableau | "#FFFFFF" |
| | BorderColorLight | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de la bordure claire du tableau | "#FFFFFF" |
| | background | Image | Image d'arrière plan du tableau | "fond.jpg" |
| | bgcolor | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de fond du tableau | "#FFFFFF" |
| | Height | Hauteur (en pixel ou %) | Hauteur du tableau | "100" \| "10%" |
| | Width | Largeur (en pixel ou %) | Largeur du tableau | "100" \| "10%" |
| | Align | Alignement (respectivement : à gauche, centré, à droite) | Définit l'alignement du tableau | "left" \| "center" \| "right" |
| | onAfterUpdate | Appel une fonction | Balise de script | "fonction()" |
| | onBeforeUpdate | Appel une fonction | Balise de script | "fonction()" |
| | onKeyDown | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est enfoncée) | "fonction()" |
| | onKeyPress | Appel une fonction | Balise de script (se produit lorsqu'on tape une touche du clavier) | "fonction()" |
| | onKeyUp | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est levée) | "fonction()" |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <THEAD> | N | F | Regroupement de tableaux | HTML 3.0 |
| | BGColor | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de fond du tableau | "#FF0000" |
| | Align | Alignement | Définit l'alignement du tableau | "top" \| "bottom" \| |

| | | | | |
|---|---|---|---|---|
| | | (respectivement : haut, bas, à gauche, centré, à droite) | | "left" \| "center" \| "right" |
| | VAlign | Alignement vertical (haut ou bas) | Définit l'alignement vertical du tableau | "top" \| "bottom" |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | onKeyDown | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est enfoncée) | "fonction()" |
| | onKeyPress | Appel une fonction | Balise de script (se produit lorsqu'on tape une touche du clavier) | "fonction()" |
| | onKeyUp | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est levée) | "fonction()" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction() |
| <TITLE> | N | O | Titre de votre page | HTML 2.0 |
| <TR> | N | F | Ligne d'un tableau | HTML 3.0 |
| | VAlign | Alignement (respectivement : en haut, au milieu, en bas, sur la ligne de base) | Alignement vertical du texte dans la cellule | "top" \| "middle" \| "bottom" \| "baselin |
| | BorderColor | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de la bordure du tableau | "#FFFFFF" |
| | BorderColorDark | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de la bordure foncée du tableau | "#FFFFFF" |
| | BorderColorLight | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de la bordure claire du tableau | "#FFFFFF" |
| | bgcolor | Couleur hexadécimale RRVVBB ou nom de la couleur | Couleur de fond du tableau | "#FFFFFF" |
| | Align | Alignement (respectivement : à gauche, centré, à droite) | Définit l'alignement du tableau | "left" \| "center" \| "right" |
| | onKeyDown | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est enfoncée) | "fonction()" |
| | onKeyPress | Appel une fonction | Balise de script (se produit lorsqu'on tape une touche du clavier) | "fonction()" |
| | onKeyUp | Appel une fonction | Balise de script (se produit lorsqu'une touche du clavier est levée) | "fonction()" |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |

| | | | | |
|---|---|---|---|---|
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <TT> | N | O | Texte de type machine à écrire | HTML 2.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <U> | N | O | Texte souligné | HTML 3.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <UL> | N | O | Liste à puces | HTML 2.0 |
| | type | Type de liste (Respectivement sous forme 1, a, A, i, I) | Type d'étiquette dans une liste d'étiquettes (numérique) | "1" \| "a" \| "A" \| "i" \| "I" |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <VAR> | N | O | Texte sous forme de variable | HTML 2.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |
| <WBR> | N | N | Désactive la balise <NOBR> | N.N 1.0 |
| <XMP> | O | O | Texte sous forme d'exemple | HTML 2.0 |
| | title | Texte | Une description du texte lorsque l'on pointe le curseur dessus | "Description" |
| | id | Nom d'une ID | Attache une ID à la balise (un style défini dans une feuille de style) | "pargr1" |
| | class | Nom d'une classe | Attache une classe à la balise (un style défini dans une feuille de style) | "titre1" |
| | style | Un style | Attache un style spécifique à la balise | "font-size: 12 pt" |
| | on,,,() | une fonction | voir Balise de script simple | onClick="fonction()" |

# F. Du style dans le texte

## 1. Les titres

Il existe 6 niveaux de titre en HTML, allant de <H1> à <H6>, <H1> étant le titre le plus important. Leur affichage dépend directement du navigateur que vous utilisez et des polices de caractère choisies.

Dans tous les cas, le texte entre ces balises est placé sur une nouvelle ligne.

| Voici une exemple de code : | s'affiche ainsi (dans le navigateur) |
|---|---|
| <H1>Mon titre en H1</H1> | **Mon titre en H1** |
| <H2>Mon titre en H2</H2> | **Mon titre en H2** |
| <H3>Mon titre en H3</H3> | **Mon titre en H3** |
| <H4>Mon titre en H4</H4> | **Mon titre en H4** |
| <H5>Mon titre en H5</H5> | **Mon titre en H5** |
| <H6>Mon titre en H6</H6> | **Mon titre en H6** |

## 2. Les différents styles de caractères

Pour appliquer un style à un mot ou à un morceau de phrase, il suffit d'encadrer le ou les mots des balises qui correspondent au style désiré. On peut aussi combiner les styles.

| Voici une exemple de code : | s'affiche ainsi (dans le navigateur) |
|---|---|
| <B> Voici du gras </B> | **Voici du gras** |
|  Voici de l'italique  - | *Voici de l'italique* |
| <TT> voici du texte en police monospace </TT> - | `voici du texte en police monospace` |
| <U> voici du texte souligné </U> | voici du texte souligné |
| STRIKE> voici du texte barré </STRIKE> ou<br><S> voici du texte barré </S> - | ~~voici du texte barré~~ |

Pour mêler les style, il vous suffit d'ajouter les balises les unes à la suite des autres, sans oublier qu'il faudra toutes les fermer :

```
<B><I><U>Voici un exemple de texte</I> partiellement en italique, en gras </B>et souligné</U>
```

s'affiche :

***Voici un exemple de texte*** **partiellement en italique, en gras** et souligné

## 3. Styles logiques

Au lieu de définir physiquement le style de caractères, il est souvent préférable de lui donner une définition logique plus facilement modifiable grace aux feuilles de styles

| Voici une exemple de code : | s'affiche ainsi (dans le navigateur) |
|---|---|
| <EM> Mise en évidence simple </EM> | *Mise en évidence simple* |
| <STRONG>Plus d'emphase</STRONG>< | **Plus d'emphase** |
| <CITE> Pour faire une citation </CITE> | *Pour faire une citation* |
| <SAMP>Le résultat d'un programme</SAMP> | `Le résultat d'un programme` |
| <VAR> ceci est une variable </VAR> | *ceci est une variable* |
| <CODE>Du code</CODE> | `Du code` |
| <KBD>Une touche du clavier</KBD> | `Une touche du clavier` |
| <DFN>Une définition</DFN> | *Une définition* |
| <ADDRESS>Une adresse</ADDRESS> | *Une adresse* |

## G. Formater le texte

### 1. Les retours à la ligne : <br> ou <p>

Le HTML est insensible aux espaces multiples ainsi qu'aux passages à la ligne.

| Voici une exemple de code : | s'affiche ainsi (dans le navigateur) |
|---|---|
| Votre texte<BR>avec un retour à la ligne simple | Votre texte<br>avec un retour à la ligne simple. |
| Votre texte<P>avec un retour à la ligne et ajout d'une ligne vierge. | Votre texte<br><br>avec un retour à la ligne et une ligne vierge |

### 2. Aligner le texte

En HTML, vous pouvez choisir d'aligner vos paragraphes de texte à gauche (par défaut), au centre ou à droite. Il existe trois solutions, aucune ne permettant cependant de justifier le texte.

- <P align=left / center / right>Texte</P> : Oui, il s'agit bien de la même balise que pour l'ajout d'un retour à la ligne, à laquelle on adjoint un attribut, celui de l'alignement (respectivement gauche / centre / droite). A noter aussi, dans cette utilisation de la balise <P>, il vous faut utiliser la balise de fermeture pour signaler la fin de l'alignement.

Ex. : <P align=center>Votre paragraphe est au centre</P>

s'affiche ainsi :

Votre paragraphe est au centre

- <CENTER>Texte</CENTER> : C'est une balise Nestcape, aujourd'hui reconnue par la majorité des navigateurs Internet. Elle centre le paragraphe entre ces balises.

Ex. : <CENTER>Votree paragraphe est au centre</CENTER>

s'affiche :

Votre paragraphe est au centre

- <DIV align=left / center / right>Texte, image, tableau</DIV> : cette balise permet d'aligner en même temps plusieurs type de données, qu'il s'agisse de texte, d'images ou encore de tableaux. Son fonctionnement est similaire a celui de <P align =...>

### 3. Mettre du texte en retrait :

<BLOCKQUOTE>...</BLOCKQUOTE>

A l'origine, cet élément devait permettre de différencier un paragraphe de citation du reste du texte, en affichant un retrait à gauche. Mais la majorité des gens utilisent simplement cette balise pour le retrait, sans que le contenu soit forcément une citation. Pour augmenter le retrait, multipliez les blockquote.

Ex. : <BLOCKQUOTE><BLOCKQUOTE>Texte avec deux retraits</BLOCKQUOTE></BLOCKQUOTE>

s'affiche ainsi :

Texte avec deux retraits

### 4. Le texte préformaté :

<PRE>.....</PRE>

Le texte que vous saisissez entre ces balises sera affiché tel que vous l'avez entré, c'est-à-dire avec les espaces et les retours à la lignes.

Ex. : <PRE> Voici un texte pre -

formate</PRE>

s'affiche ainsi :

Voici un texte pre-

formate

### 5. Césure ou pas de césure

- Interdire la césure : <NOBR>Phrase de texte</NOBR> - Le navigateur ne pourra alors pas faire de césure ou de retour automatique à la ligne dans le texte encadré par ces balises. Si besoin est, il ajoutera des ascenceurs horizontaux pour que le lecteur puisse afficher le reste de la ligne.

- Programmer une césure : <WBR>Texte</WBR> - Si la portion de texte pour laquelle vous avez interdit les retours automatiques est longue, il est judicieux de placer cet élément pour permettre au navigateur, si besoin est, d'utiliser la césure programmée pour un retour à la ligne.

### 6. Les traits de séparation :

<HR>

Pour aérer vos textes, il peut être judicieux de séparer les parties par des traits horizontaux. Cet élément HTML ne possède pas de balise de fin, puisque le trait s'affiche sur une nouvelle ligne.

| **Attribbut** | **signification** |
|---|---|
| align = left / center (défaut) / right | stipule l'alignement du trait. |
| width = pixels / pourcentage | spécifie la largeur du trait, soit en pixels, soit en pourcentage de la taille de la fenêtre. |
| height = pixels | hauteur ou épaisseur du trait en pixels. |
| noshade | efface l'effet d'ombrage affiché par défaut dans le navigateur. |

### 7. Les caractères spéciaux

Dans un fichier Html, on ne peut mettre que du code ascii standard, certains caractères doivent être remplacé par un code spécial : celui-ci commence par un « **&** » et se termine par un « **;** ».

| Lettre | Code | Lettre | Code | Lettre | Code | Lettre | Code | Lettre | Code | Lettre | Code |
|---|---|---|---|---|---|---|---|---|---|---|---|
| à | à | ñ | ñ | À | À | Ñ | Ñ | < | < | | |
| á | á | ò | ò | Á | Á | Ò | Ò | > | > | | |
| â | â | ó | ó | Â | Â | Ó | Ó | " | " | | |
| ä | ä | ô | ô | Ä | Ä | Ô | Ô | „ | „ | ™ | ™ |
| ã | ã | õ | õ | Ã | Ã | Õ | Õ | « | « | ° | ° |
| å | å | ö | ö | Å | Å | Ö | Ö | » | » | § | § |
| æ | æ | ø | ø | Æ | &Aelig; | Ø | Ø | © | © | ¶ | ¶ |
| ç | ç | œ | œ | Ç | Ç | Œ | Œ | ® | ® | ± | ± |
| è | è | ù | ù | È | È | Ù | Ù | & | & | ¹ | ¹ |
| é | é | ú | ú | É | É | Ú | Ú | ¥ | ¥ | ² | ² |
| ê | ê | û | û | Ê | Ê | Û | Û | ¢ | ¢ | ³ | ³ |
| ë | ë | ü | ü | Ë | Ë | Ü | Ü | £ | £ | ½ | ½ |
| ì | ì | ý | ý | Ì | Ì | Ý | Ý | ß | ß | ¼ | ¼ |
| í | í | Ÿ | ÿ | Í | Í | Ÿ | Ÿ | € | € | ¾ | ¾ |
| î | î | Ð | ð | Î | Î | Ð | Ð | *f* | ƒ | · | · |
| ï | ï | Þ | þ | Ï | Ï | Þ | Þ | ‰ | ‰ | | |

### 8. Indice et exposant

Dans vos formules mathématiques ou chimiques, il se peut que vous ayez besoin de mettre des caractères en indice ou en exposant.

+ indice : H<SUB>2</SUB>O : $H_2O$

+ Exposant : 10<SUP>3</SUP> : $10^3$

## H. Afficher des images

### 1. Les formats de fichiers graphiques

Il n'y a que deux formats qui soient reconnues par la plupart des navigateurs : le GIF et le JPG.

- le format GIF : particulièrement à l'aise avec les aplats de couleurs (dessins), ce format permet des fichiers de faible encombrement et permet l'entrelacement (affichage progressif par bandes) et la transparence ainsi que les annimations.
- le format JPG permet de varier le taux de compression et donc de mieux adapter le rapport taille / qualité de l'image. Ce format est recommandé pour les photos et en général les images comportant des dégradés de couleurs.

Il existe d'autres formats, mais il faut alors que votre visiteur ait ajouté à son navigateur les pluggins nécessaires (petits programmes additionnels).

### 2. Affichage simple d'une image

Le code de base est des plus simples : <IMG SRC="xxx" />

Cet élément ne comporte pas de balise de fin.

xxx représente l'URL du fichier image. Cet URL peut évidemment être un chemin relatif local. Une bonne idée consiste à créer un répertoire pour stocker les images.

### 3. Les attributs de <IMG>

| Attribbut | signification |
|---|---|
| width="pixels" | indique la largeur de l'image à afficher |
| height="pixels" | indique la hauteur de l'image à afficher |
| alt="texte descriptif" | quelques mots qui s'affichent avant le chargement de l'image, ou en cas d'impossibilité de charger le fichier |
| align="left / center / right" | alignement de l'image sur la page |
| border="pixel" | par défaut, le navigateur affiche un cadre de 1 pixel d'épaisseur autour de l'image. 0 signifie qu'il n'y a pas de cadre. |
| vspace="pixels" | spécifie un espace vierge en pixels de chaque côté de l'image |
| hspace="pixels" | spécifie un espace vierge en pixels en haut et en bas de l'image |
| lowsrc="fichierbis.jpg" | spécifie un fichier en basse résolution de l'image à charger. Si votre fichier graphique est lourd, utiliser cet attribut pour faire patienter vos visiteurs avec une image en basse résolution. Cette image doit avoir les mêmes dimensions que l'image finale. |

Voici un exemple de code reprenant tous ces attributs :

```
<IMG SRC="../../images/fic1.gif" lowsrc="../../images/fic2.jpg" height="125" width="125" alt="exemple" border="0" align=center vspace=10 hspace=10>
```

Pendant son chargement, il affiche le texte "exemple", puis l'image en basse résolution "fic2.jpg". Sa taille est de 125 x 125 et elle est alignée au centre. Elle ne possède pas de bordure et un espace de 10 pixels l'entoure complètement.

### 4. Une image pour le fond de votre page

Nous le verrons plus tard, vous pouvez spécifier la couleur du fond de votre page, celle du texte, et des liens visités. Mais nous allons voir maintenant comment charger une image comme fond de page. Le navigateur répète cette image pour "remplir" entièrement le fond de la fenêtre ouverte.

Un conseil, prenez un fichier très léger et dont les motifs ne nuiront pas à la lecture de vos documents.

Il s'agit en fait d'un attribut de la balise de début

```
<BODY background="fichier.jpg">
```

N'oubliez pas de fournir le chemin du fichier si celui-ci ne se trouve pas dans le même dossier/répertoire que le fichier HTML.

## I. Les Liens HyperTextes

Toute la puissance d'Internet réside sans doute dans sa faculté à relier les sites et les documents entre eux, grâce aux liens hypertextes.

### 1. Créer un lien hypertexte - <A>

Un lien HTML possède toujours deux extrémités : le début (le pointeur, ou le lien) et la fin (la cible).

Le début correspond au texte (ou image) qui pointe vers un autre élément. Souvent, il s'agit du texte souligné et de couleur différente au-dessus duquel le curseur de la souris se transforme. Pour une image, si celle-ci ne possède pas de cadre, seule le changement de forme du curseur indique qu'il s'agit d'un lien.

La fin d'un lien est le fichier qui s'affiche lorsque le lien a été cliqué. Il peut s'agir d'un fichier HTML, d'une partie du même fichier (voir la section suivante sur les ancres), d'une image, de vidéo ou de son. Ce document peut se trouver sur le même site (on utilise alors une adresse relative. Ex : ../images/fichier.html) ou sur un autre site (on utilise alors une adresse absolue. Ex. : http://www.monsite.com/fichier.html).

La mise en oeuvre d'un lien hypertexte est très simple :

```
<A HREF="adresse_document_visé.html">Lien hypertexte</A>
```

Il vous suffit d'indiquer l'adresse (relative ou absolue) du fichier qui devra se charger quand le texte sera cliqué.

### 2. Utiliser les ancres - <A name="...">

Grâce aux ancres, il est possible de spécifier une section de texte, à l'intérieur d'un document HTML, qui devra être affichée. Pour être plus clair, imaginons que nous ayons un document contenant des poésies, dont l'une des sections, disons la partie 2, concerne les oeuvres de Baudelaire. Si dans un sommaire nous faisons référence à cette partie, au milieu de beaucoup d'autres. Nos visiteurs, intéressés particulièrement par Baudelaire et non par les autres, pourrait grâce à un lien aller directement à cette partie. Il nous faut pour cela construire dans le document contenant les poésies, un marqueur invisible identifiant la partie concernant Baudelaire :

#### *a) Insérer une ancre*

A l'endroit que vous souhaitez identifier, placez la balise suivante : <A name="partie2">II- La poésie de Baudelaire</A>

Le nom que vous donnez à une ancre doit être unique à l'intérieur d'un même document HTML.

dans le sommaire, un lien pointant vers le fichier, et plus précisement, à l'intérieur de ce fichier, vers l'ancre identifiant cette partie.

#### *b) Pointer vers une ancre*

Lorsque vous construisez votre lien vers le fichier poesie.html (c'est un exemple!), il vous suffit d'ajouter le nom de l'ancre à la suite du nom du fichier. Comme suit : <A HREF="poesie.html#partie2">Baudelaire et le Mal de Vivre</A>

N'oubliez surtout pas le signe dièse (#) entre le nom de fichier et celui de l'ancre, car c'est lui qui indique au navigateur de chercher l'ancre "partie2" dans le fichier "poesie.html".

Vous pouvez aussi faire référence à une ancre située dans un même fichier : <A HREF="#partie2">Baudelaire</A>

Voilà. Vous pouvez désormais relier vos documents entre eux. Faites des essais, le plus dur étant d'être très rigoureux dans le chemin que l'on fournit.

## J. Les Listes

Il est parfois utile de faire appel à une liste dont la présentation sera plus parlante qu'une longue énumération. Le HTML a prévu un certain nombre de type de listes.

### 1. Listes à puces et listes numérotées

Chaque liste est encadrée des balises indiquant le style de la liste : <UL> pour les listes à puces et <OL> pour les listes numérotées. Chaque élément de la liste est précédé de la balise <LI> (pas de balise de fin) et sera présenté avec un retour à la ligne simple et une puce le précédant.

#### *a) Les listes à puces - <UL>*

| Code HTML | Présentation dans un navigateur |
|---|---|
| <UL><br><LI>Mon premier élément de liste<br><LI>Mon deuxième élément de liste<br></UL> | • Mon premier élément de liste<br>• Mon deuxième élément de liste |

Nous pouvons néanmoins agir sur la forme de la puce en ajoutant l'attribut type=

- type="disc (défaut) / circle /square" - circle affiche des puces en forme de cercle évidé et square des carrés.

Voici un exemple :

```
<UL type="square">
        <LI>Mon premier élément de liste
        <LI>Mon deuxième élément de liste
</UL>
```

#### *b) Les listes numérotées - <OL>*

Elles se construisent de la même manière que les listes à puces, si ce n'est que les puces sont remplacées par de nombres.

| Code HTML | Présentation dans un navigateur |
|---|---|
| <OL><br><LI>Mon premier élément de liste<br><LI>Mon deuxième élément de liste<br></OL> | 1. Mon premier élément de liste<br>2. Mon deuxième élément de liste |

Il éxiste deux attributs pour les listes numérotées : le premier permet de choisir le type de numérotation utilisé, et le second de spécifier un nombre de départ de la liste.

- type="A / a / I / i / 1 (défaut)

A - chaque élément de la liste sera précédé d'une lettre majuscule en guise de numérotation

a - numérotation par des lettres minuscules

I - numérotation par capitales romaines

i - numérotation par minuscules romaines

1 - numérotation par chiffres arabes, option par défaut si l'attribut "type" n'est pas spécifié

- start=nombre

Cette option peut s'avérer utile si vous décidez de partager une liste en plusieurs colonnes d'un tableau. Le nombre spécifié pour start correspond au numéro (ou la lettre) qui sera attribué au premier élément de la liste.

Exemple de code :

```
<OL type="i" start=6>
        <LI>Mon premier élément de liste
        <LI>Mon deuxième élément de liste
</OL>
```

### 2. Les listes descriptives - <DL>

Les listes descriptives se différencient des listes à puces et des listes numérotées par le fait que les éléments ne sont pas numérotées mais regroupés dans des rubriques ou parties.

La liste complète est encadrée par les balises <DL> et </DL>. Chaque titre de rubrique est précédé par la balise <DT> (pas de balise de fin) et chaque élément de rubrique et précédé de <DD> (pas de balise de fin).

| Code HTML | Présentation dans un navigateur |
|---|---|
| <DL><br><DT>Ma première rubrique<br><DD>Mon premier élément de première rubrique<br><DD>Mon deuxième élément de première rubrique<br><DT>Ma deuxième rubrique<br><DD>Mon premier élément de première rubrique<br><DD>Mon deuxième élément de première rubrique<br></DL> | Ma première rubrique<br>Mon premier élément de première rubrique<br>Mon deuxième élément de première rubrique<br>Ma deuxième rubrique<br>Mon premier élément de deuxième rubrique<br>Mon deuxième élément de deuxième rubriqu |

### 3. Les listes <DIR> et <MENU>

Ces deux types de listes (qui disparaissent dans le HTML 4) sont très similaires à la liste à puces. Seul le rendu de ces listes est différent (ce rendu diffère même selon le navigateur que vous utilisez). Je me contenterai donc de vous donner deux exemples de codes :

| Liste Menu | Liste Dir (répertoire ou listing) |
|---|---|
| <MENU><br><LI>Mon premier élément de liste<br><LI>Mon deuxième élément de liste<br></MENU> | <DIR><br><LI>Mon premier élément de liste<br><LI>Mon deuxième élément de liste<br></DIR> |

### 4. Mêler les listes

Pour bien marquer les différentes parties de ce sommaire, vous pouvez "imbriquer" différentes listes, et même des listes de types différents. Voici un exemple :

```
<OL type="I">
<LI>Les débuts de la poésie
   <UL>
   <LI>Le Moyen-Age
   <LI>La Renaissance
   </UL>
<LI>Les poètes les plus connus
   <UL>
   <LI>Baudelaire
   <LI>Verlaine
   </UL>
</OL>
```

## K. Couleurs et Polices

### 1. Définition des couleurs en HMTL

Il y a deux possibilités en HTML pour définir une couleur :

- l'utilisation des 16 couleurs standards (héritage des premiers écrans VGA sous Windows)
- l'utilisation de codes hexadécimaux

#### *a) Les couleurs standard*

Elles sont donc au nombre de 16, et sont reconnues par tous les systèmes. En voici la liste :

| | | | |
|---|---|---|---|
| Aqua | black | blue | fuschia |
| gray | green | lime | maroon |
| navy | olive | purple | red |
| silver | teal | white | yellow |

Leur emploi est simple, comme nous le verrons ultérieurement.

#### *b) Les codes hexadécimaux*

Pour utiliser ces couleurs converties en code hexadécimaux, il vous faut précéder la valeur par un signe dièse (#).

Voici quelques exemples de codes hexadécimaux de couleurs :

- code hexadécimal pour le blanc : FFFFFF
- code hexadécimal pour le noir : 000000

### 2. Les éléments de la page

Nous avons vu dans la partie "Afficher des images" qu'il était possible de spécifier un fichier graphique qui sert de fond de page. Il est cependant aussi possible d'intervenir sur la couleur d'autres éléments de la page, tel que le fond de la page, le texte, les liens, les liens visités ou encore le lien cliqué. Tout ceci se modifie en ajoutant des attributs à la balise <BODY>.

Les attributs de <BODY> :

| attribut | signification |
|---|---|
| bgcolor= "couleur / code hexadécimal" | spécifie la couleur du fond de page. |
| text="couleur / code hexadécimal" | spécifie la couleur par défaut pour tout le texte du document. |
| link= "couleur / code hexadécimal" | spécifie la couleur des liens hypertextes et du cadre des images agissant comme des liens hypertextes. |
| vlink= "couleur / code hexadécimal" | spécifie la couleur des liens visités dans le document. |
| alink= "couleur / code hexadécimal" | spécifie la couleur du lien au moment où il est cliqué (bouton de la souris en bas). |

Voici un exemple de code :

```
<BODY bgcolor="#000000" text="white" link="blue" vlink="yellow">
```

Le fond de page sera noir, le texte blanc, les liens en bleu et les liens visités seront jaunes.

 Note : Lorsque vous spécifiez une couleur par son code hexadécimal, n'oubliez pas le signe dièse (#), sans lequel votre code ne sera pas reconnu.

### 3. Les polices de caractères : La balise <FONT>

| attribut | signification |
|---|---|
| face="nom de la police1, nom de la police2, etc" | spécifie une liste de polices de caractère par ordre de préférence. Si la police n'est pas présente dans le système de votre visiteur, le navigateur cherche la suivante et ainsi de suite. Si aucune n'est présente, il affiche le texte dans la police de caractère par défaut. |
| size=valeur absolue / valeur relative | spécifie la taille des caractères. Deux méthodes coéxistent : - la valeur absolue : les tailles vont de 1 à 7, 3 étant la taille normale par défaut. On spécifie alors la taille par size=5, ce qui a pour effet d'agrandir de deux tailles les caractères ainsi encodés. - la valeur relative : 3 étant la taille standard, on spécifie alors size=+2, ce qui a le même effet que dans l'exemple précédent. |
| color="couleur standard / code hexadécimal" | spécifie la couleur du texte encodés. |

Voici un exemple de code :

```
<FONT face="Helvetica, Times" size=+2 color="#AE3F1A">Mon texte special</FONT>
```

Ce qui s'affiche ainsi : Mon texte special

#### *a) Changer la taille par défaut*

Il est possible de changer la taille par défaut du texte. On utilise la balise <BASEFONT> (sans balise de fin).

Placée à un endroit quelconque, elle stipule une taille de caractère par une valeur absolue ou relative (voir plus haut) à tout le texte qui la suit. Si vous faites appel à la balise <FONT> pour modifier la taille d'une portion de texte, la taille normale sera celle spécifiée dans <BASEFONT>. Ceci est particulièrement important pour utiliser des valeurs relatives.

Ne possédant pas de balise de fin, si vous désirez retrouver la taille par défaut du HTML (c'est-à-dire 3 en valeur absolue), il vous faudra disposer une nouvelle balise <BASEFONT> pour l'indiquer.

Exemple de code :

```
<BASEFONT size=+2>
```

Le texte qui suit cette balise verra sa taille augmentée de deux niveaux.

## L. Les Tableaux

.

### 1. Définir un tableau - <TABLE>

Les balises <TABLE> et </TABLE> marquent le début et la fin d'un tableau.

Il est possible d'ajouter un certain nombre d'attribut à la balise de début pour modifier l'aspect du tableau.

Les attributs de <TABLE> :

| Attributs | signification |
|---|---|
| bgcolor="couleur standard / code hexadécimal" | spécifie la couleur de fond du tableau. |
| background="fichier graphique" | spécifie un fichier image en guise de fond pour le tableau. Les formats GIF et JPG sont reconnus universellement. |
| border="pixel" | spécifie l'épaisseur en pixels du cadre et des bordures intérieures du tableau ("0" signifie qu'il n'y a pas de bordure, et 1 est la valeur par défaut). |
| bordercolor="couleur standard / code hexadécimal" | spécifie la couleur du cadre et des bordures intérieures du tableau. |
| height="pixels" | spécifie la hauteur en pixels du tableau. |
| width="pixels / %" | spécifie la largeur du tableau, soit en pixels, soit en pourcentage par rapport à la taille de la fenêtre. |
| cellpadding="pixels" | spécifie l'épaisseur des bordures intérieures du tableau |
| cellspacing="pixels" | spécifie l'espace en pixels entre le contenu des cellules et le bord de la cellule. |

Exemple de code :

```
<TABLE border="0" align="center" bgcolor="gray" width="50%" cellpadding="0" cellspacing="2">
```

Ici, le tableau n'aura pas de bordure, aura un fond gris, une largeur égale à la moitié de celle de la fenêtre du navigateur. L'épaisseur des bordures internes sera nulle et l'espace entre le bord des cellules et leur contenu sera de 2 pixels.

### 2. Les cellules d'un tableau

#### *a) Les titres*

Un tableau se lit et se construit de haut en bas, en suivant chaque ligne de gauche à droite.

- <CAPTION>Titre du tableau</CAPTION> : fournit un titre au tableau.

- <TH>Colonne 1</TH> : utilisées à la place des balises de cellule (voir plus bas), elles permettent de donner un titre à une colonne ou à une rangée d'un tableau.

Attributs de <TH> :

| *Attributs* | *signification* |
|---|---|
| align="left (défaut) / center / right" | spécifie l'alignement des données à l'intérieur de la cellule (gauche, centre ou droite). |
| valign="top / middle (défaut) /bottom" | spécifie l'alignement vertical des données à l'intérieur de la cellule (en haut, au milieu ou en bas). |
| bgcolor="couleur standard / code hexadécimal" | spécifie la couleur de fond de la cellule. |
| background="fichier graphique" | spécifie un fichier image en guise de fond pour la cellule. Les formats GIF et JPG sont reconnus universellement. |
| height="pixels" | spécifie la hauteur de la cellule de titre. |
| width="pixels / %" | spécifie la largeur de la cellule de titre (% par rapport à la largeur totale du tableau). |

### b) Les cellules

+ Pour créer une nouvelle ligne dans le tableau (y compris pour les titres de colonnes utilisant les balises <TH>) :
<TR>vos cellules</TR>

Attributs de <TR> :

| ***Attributs*** | ***signification*** |
|---|---|
| align="left (défaut) / center / right" | spécifie l'alignement des données à l'intérieur des cellules de la rangée (gauche, centre ou droite). |
| valign="top / middle (défaut) /bottom" | spécifie l'alignement vertical des données à l'intérieur des cellules de la ligne (en haut, au milieu ou en bas). |
| bgcolor="couleur standard / code hexadécimal" | spécifie la couleur de fond des cellules de la ligne. |
| background="fichier graphique" | spécifie un fichier image en guise de fond pour les cellules de la ligne. Les formats GIF et JPG sont reconnus universellement. |
| height="pixels" | spécifie la hauteur de la ligne. |

+ Pour créer une cellule : <TD>Le contenu de la cellule (texte, images,...)</TD>

Attributs de <TD> :

| ***Attributs*** | ***signification*** |
|---|---|
| align="left (défaut) / center / right" | spécifie l'alignement des données à l'intérieur de la cellule (gauche, centre ou droite). |
| valign="top / middle (défaut) /bottom" | spécifie l'alignement vertical des données à l'intérieur de la cellule (en haut, au milieu ou en bas). |
| bgcolor="couleur standard / code hexadécimal" | spécifie la couleur de fond de la cellule. |
| background="fichier graphique" | spécifie un fichier image en guise de fond pour la cellule. Les formats GIF et JPG sont reconnus universellement. |
| height="pixels" | spécifie la hauteur de la cellule. |
| width="pixels" | spécifie la largeur de la cellule. |
| colspan="nombre de colonnes" | spécifie que la cellule occupe plusieurs colonnes (vers la droite). |
| rowspan="nombre de lignes" | spécifie le nombre de lignes occupées par la cellule (vers le bas). |

### 3. Exemple de code pour un tableau

```
<TABLE width=400 border=1 bordercolor="#000000" cellpadding=2 cellspacing=2 bgcolor="gray">
<TR>
<TD align=center width=150>cellule 1 de la ligne 1</TD>
<TD align=center width=250>cellule 2 de la ligne 1</TD>
</TR>
<TR>
<TD align=center colspan=2 bgcolor="yellow">cellule unique de la ligne 2</TD>
</TR>
<TR>
<TD>cellule 1 de la ligne 3</TD>
<TD rowspan=2 bgcolor="red">cellule 2 des lignes 3 et 4</TD>
</TR>
<TR>
<TD>cellule 1 de la ligne 4</TD>
</TR>
</TABLE>
```

Ce qui devrait apparaitre comme suit :

<table>
<tr><td>cellule 1 de la ligne 1</td><td>cellule 2 de la ligne 1</td></tr>
<tr><td colspan="2">cellule unique de la ligne 2</td></tr>
<tr><td>cellule 1 de la ligne 3</td><td rowspan="2">cellule 2 des lignes 3 et 4</td></tr>
<tr><td>cellule 1 de la ligne 4</td></tr>
</table>

## M. Les frames

Les frames sont les cadres créés dans la fenêtre du navigateur. Chaque cadre reçoit un fichier HTML (différent). Ce principe est surtout utilisé pour des barres de navigations, des hauts de

La tendance actuelle veut qu'on évite de mettre des frames, car la maintenance du site est plus délicate et les moteurs de recherche accorde des pénalités à ces sites.

Dans le cas où nous construisons une page contenant ***n*** frames, il nous faut au moins ***n+1*** fichiers

- le fichier <FRAMESET>, celui qui répartit les cadres dans la fenêtre
- 1 fichiers HTML pour le contenu de chaque cadre

### 1. Le fichier <FRAMESET>

Comme un tableau, un fichier frameset se construit en lisant les cadres de haut en bas, et de gauche à droite. Mais commençons par un exemple de code simple :

```
<HTML>
<HEAD>
<TITLE>Mon site</TITLE>
</HEAD>
<FRAMESET cols="20, *">
<FRAME SRC="fichier1.html" name="colGauche">
<FRAME SRC="fichier2.html" name="colDroite">
<NOFRAMES>
Ce site a ete construit avec des frames. Il semble que votre navigateur ne supporte pas ces balises.
Voici un lien pour visiter notre site sans frames : <A HREF="fichier2.html">Site sans frames</A>
</NOFRAMES>
</FRAMESET>
</HTML>
```

Le fichier <FRAMESET> est le fichier qui crée et répartit dans la fenêtre les différents cadres (frames) de notre page. Les balises <FRAMESET> et </FRAMESET> remplacent les balises traditionnelles <BODY> et </BODY>.

Ici, nous avons créé deux colonnes, la première fait 20 pixels de large et la seconde occupe le reste disponible de la fenêtre du navigateur. Dans le colonne de gauche de 20 pixels de large - son nom étant "colGauche" -, nous chargeons le fichier "fichier1.html" et dans la colonne de gauche - "colGauche" -, nous chargeons le fichier "fichier2.html".

Au cas où le navigateur de notre visiteur ne supporterait pas les balises de frames, nous fournissons un contenu alternatif via les balises <NOFRAMES>. Ici, un lien vers fichier2.html.

### 2. Les attributs pour le fichier <FRAMESET>

Les attributs de la balise <FRAMESET> :

| *Attributs* | *signification* |
|---|---|
| cols="pixels / %, pixels / %" | crée des colonnes dont la largeur peut être spécifiée en pixels ou en pourcentage de la fenêtre totale. Les dimensions pour chaque colonne sont séparées par des virgules. (on note aussi qu'il est possible d'indiquer le signe "*" pour indiquer que la colonne occupe l'espace restant). |
| rows="pixels / %, pixels / %" | crée des cadres horizontaux (lignes). Il n'est pas possible d'utiliser cols et rows dans le même frameset (voir dans l'exemple en fin de leçon). |
| framespacing="pixels" | indique l'espace entre les deux cadres. |
| frameborder="pixels" | épaisseur de la bordure des cadres. |
| bordercolor="code hexadécimal" | couleur de la bordure des cadres (ici, les noms des 16 couleurs standard ne sont pas possibles). |
| border="pixels" | épaisseur de la bordure entre les cadres. |

 NOTE IMPORTANTE : si vous désirez que vos cadres n'aient pas de bordure, il vous faut fixer les trois attributs suivants à la valeur de "0" pour que cela fonctionne sous Netscape et Internet Explorer : framespacing, frameborder et border.

Certains de ces attributs sont redondants, mais c'est parce qu'ils ne fonctionnent soit que sous Internet Explorer soit sous Netscape.

Les attributs des balises <FRAME> :

| *Attributs* | | *signification* |
|---|---|---|
| src="fichier" | | indique le fichier initialement chargé dans la fenêtre. |
| name="texte" | | spécifie un nom pour la fenêtre créée. C'est particulièrement important car vous ferez appel à ce nom lorsque vous voudrez charger un fichier dans ladite fenêtre.: |
| -Bien que le nom que vous donnez à vos fenêtres soit libre, il éxiste certains noms qui sont réservés en HTML | "parent" | le fichier est chargé dans la fenêtre entière (efface les frames) |
| | "blank" | une nouvelle fenêtre est ouverte dans votre navigateur pour y charger le fichier. L'ancienne fenêtre reste ouverte en arrière plan. |
| | "top" | ouvre le fichier dans une fenêtre supérieure hiérarchiquement (dans le cas où plusieurs fenêtres du navigateur sont ouvertes). |
| scrolling="yes / no / auto" | | autorise ou non la présence d'ascenseurs de défilement pour le cadre. "auto" laisse au navigateur le soin d'afficher des ascenseurs si besoin est (c'est l'option par défaut si rien n'est spécifié). |
| marginheight="pixels" | | marges disposées en haut et en bas du cadre. |
| marginwidth="pixels" | | marges disposées de chaque côté du cadre. |

### 3. Des fichiers HTML et des frames

En fait, rien ne différencie à première vue les fichiers HTML pour un site avec des frames des autres fichiers HTML. La seule différence s'observe dans les liens hypertextes : il faut préciser la frame de destination pour l'affichage du fichier à charger. Par défaut, si rien n'est indiqué, le fichier se charge dans le cadre où se trouvait le lien hypertexte.

Voici un exemple de code :

```
<A HREF="monExemple.html" target="colDroite">Un exemple de chargement dans une frame</A>
```

En guise de rappel, trois noms réservés vous permettent de sortir des frames : "_top", "_parent", "_blank" (à utiliser avec l'attribut target).

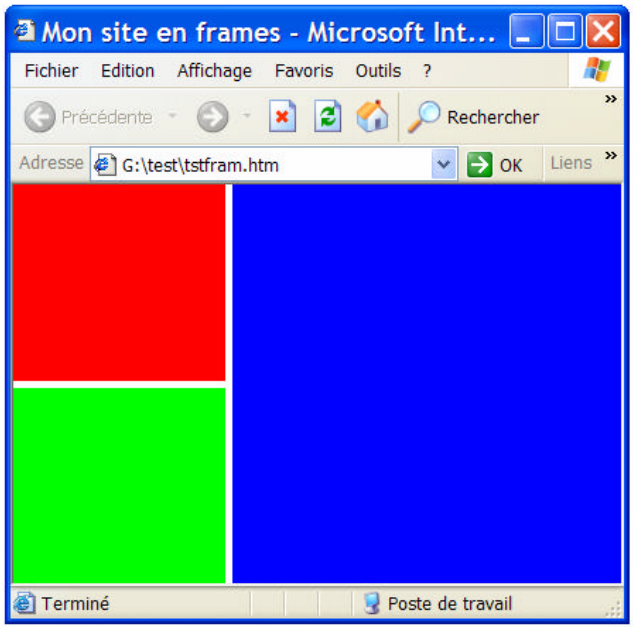
### 4. Exemple de code pour un fichier frameset

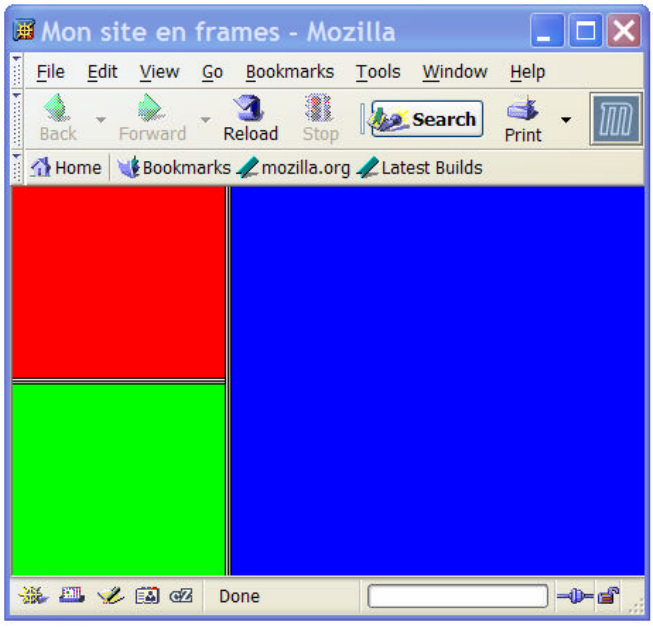
Nous allons construire trois frames : deux colonnes, dont celle de gauche contenant deux rangées.

```
<HTML>
 <HEAD>
  <TITLE>Mon site en frames</TITLE>
 </HEAD>
 <FRAMESET cols="150, *" border="5" frameborder="5" bordercolor="#000000">
  <FRAMESET rows="50%, 50%">
   <FRAME SRC="fichier1.html" name="hautGauche">
   <FRAME SRC="fichier2.html" name="basGauche">
  </FRAMESET>
  <FRAME SRC="fichier3.html" name="colDroite">
 <NOFRAMES>
  Votre navigateur ne supporte pas les frames. Voici un acces a <A HREF="fichier3.html">notre site
sans frames</A>
 </NOFRAMES>
  </FRAMESET>
</HTML>
```

Voici deux captures d'écran qui vous permettront de visualiser le résultat. On note que sous Explorer 3, la couleur et l'épaisseur des bordures de cadre ne fonctionnent pas.

Capture du résultat sour Internet Explorer 6 et Mozilla 1.4




## N. Formulaires de saisie

Il permet la saisie d'informations, qui sont retournées au serveur pour traitement.

<FORM METHOD=.. ACTION=..>...</FORM>

Encadre une section "form".

METHOD= [GET | POST] : indique au serveur la méthode a utiliser pour traiter la requète. par defaut, la méthode GET est employée. Cette méthode transcode les reponses du formulaire au format MIME "x-www-form-urlencoded". c'est la plus rapide et la plus pratique pour des réponses courtes.

Pour des formulaires de grande taille ou pour assurer une discression, la méthode POST seule permet le transfert, car GET impose une longueur limite des valeurs renvoyées et les informations apparaissent.

ACTION="url script" : indique au serveur le script a lancer lors du retour du formulaire.

<INPUT NAME="." TYPE="." VALUE="." >

| **Attribbut** | **signification** |
|---|---|
| NAME=nom | donne un nom a l'objet, utilisé par le serveur. |
| TYPE=nom_de_type | identifie le type d'objet a afficher: |
| TEXT : champ de saisie de texte | C'est le type par defaut. |
| SUBMIT | bouton qui déclenche le renvoi du formulaire au serveur |
| IMAGE: | comme submit, mais affiche une image dont l'url est donné par SRC="url" . De plus, les coordonnées du clic souris sur l'image sont transmises au serveur. Voir a ce sujet les server-side map |
| RESET | bouton qui reinitialise tous les objets de la FORM a leurs valeurs par défaut. |
| CHECKBOX | case à cocher |
| PASSWORD | : permet de saisir un mot de passe ou un code secret; les caractères saisis sont remplacés par des "*" à l'écran. |
| HIDDEN | n 'apparait pas à l 'utilisateur. Sa valeur par defaut est souvent utilisée par le script pour identifier le formulaire (un script peut ainsi traiter plusieurs formulaires) |
| RADIO | Remarque: Cle type est "declassé"; il n'a pas été repris par la norme HTML 3. |
| VALUE=valeur | : a une signification différente suivant le TYPE de l'INPUT, soit: |
| submit, reset | : Type du bouton |
| checkbox | definit le texte renvoyé au serveur si la case est cochée (ex: true, checked, on...) |
| <SELECT NAME="." [SIZE=n [MULTIPLE]]>...</SELECT> | Encadre une liste de choix prédéfinis, les OPTIONs SI l'attribut SIZE=n est présent, la liste aura l'apparence d'une liste à ascenseur de n lignes. Sinon, la liste sera une liste "pop-up", dont les options s'affichent lors d'un clic souris. |
| MULTIPLE | le choix de plusieurs options est posssible (liste ascenseur seulement) |
| <OPTION [SELECTED]> | Definit une entrée dans la liste de choix |
| SELECTED | indique le choix par défaut |
| <TEXTAREA NAME="." ROWS=n COLS=n >...</TEXTAREA> | définit une fenetre de saisie de texte à plusieurs lignes. Le texte situé entre les balises de début et fin est le texte par défaut. |
| ROWS=n | nombre de ligne de la fenetre |
| COLS=n | nombre de colonnes de la fenetre |

Exemple :

```
<form method="GET" action="" >
  <input name="idform" type="hidden" value=100>
  <p>votre Nom : <input name="nom" type=text value="toto" >
  <p>Statut:<select name="statut">
        <option value=1 >celibataire
        <option value=2 >marie
  </select>
  <p />une checkbox: <input name="cbox1" type="checkbox" value="true">
  <br />une autre: <input name="cbox2" type="checkbox" value="false">
```

```
  <br />une troisième: <input name="cbox3" type="checkbox" value="false"><p />
  <input type="radio" name="monaie" checked  value="ff"> franc français <br />
  <input type="radio" name="monaie" value="fb"> franc belge <br />
  <input type="radio" name="monaie" value="dm"> Deutch Mark <br />
  <input type="radio" name="monaie" value="pe"> pesetas espagnole <br />
  <input type="radio" name="monaie" value="li"> Lire Italienne <br />
  vos remarques :<br /><textarea name="rem" value="" rows="3" cols="56">facultatif</textarea><br />
<p> <input name="send" type="submit" value="Envoyer" >
<p> <input name="reset" type="reset" value="defaut">
</form>
```

donne :

votre Nom : toto

Statut: celibataire

une checkbox: ☐
une autre: ☐
une troisième: ☐

- ◉ franc français
- ○ franc belge
- ○ Deutch Mark
- ○ pesetas espagnole
- ○ Lire Italienne

vos remarques :

facultatif

Envoyer

defaut

# IV. Javascript

## A. Généralités

:

```
<SCRIPT LANGAGE="JavaScript">
...
</SCRIPT>
```

Exemple:

```
<html>
<body>
<SCRIPT LANGAGE="JavaScript">
<!--
document.write('bonjour');
//-->
</SCRIPT>
</body>
</html>
```

est équivalent à

```
<SCRIPT LANGAGE="JavaScript">
<!--
document.write('<html> /n');
document.write('<body> /n');
document.write('bonjour /n');
document.write('</body>/n');
document.write('</html>
//-->
</SCRIPT>
```

mais ne présente aucun interêt

## B. Erreur dans un javascript

En Javascript, il y a 2 types d'erreurs :

### 1. Erreur simple

Le script (programme) s'exécute de <SCRIPT LANGAGE="JavaScript"> jusqu'à l'apparition de la première erreur puis rien ne se passe entre la première erreur et </SCRIPT>

### 2. ErreurGrave

Rien ne s'exécute entre <SCRIPT LANGAGE="JavaScript> et </SCRIPT>

## C. Commentaires

Un script javascript se commente comme en C:

Exemple :

```
<SCRIPT LANGAGE="JavaScript">
<!--
// commentaire de fin de ligne
/* commentaire
sur plusieurs
lignes */
//-->
</SCRIPT>
```

Tout ce qui se trouve dans un commentaire est ignoré. Il est conseillé, comme toujours, de commenter largement ses scripts.

## D. Constantes, Variables, types

Le code javascript s'écrit dans une page Html entre les balises :

```
<SCRIPT LANGAGE="JavaScript">
<!--
…
//-->
</SCRIPT>
```

Ce code est interprété par le navigateur.

Les balises
<!-- //-->
sont là au cas ou le navigateur (trop ancien) n'interpréterait pas le javascript et serait interprété par le navigateur comme un commentaire html

### 1. Variables

Le typage des variables est implicite en javascript. Il n'est donc pas nécessaire de déclarer leur type au préalable ni même de les déclarer avant leur utilisation.

**Le nom des variables est sensible à la casse (ie : x != X).**

Les noms de variables suivent les mêmes règles de nommage que les autres entités javascript. Un nom de variable valide doit commencer par une lettre ou un souligné (_), suivi de lettres, chiffres ou soulignés.

Une lettre peut être une des lettres minuscules (a à z) ou majuscules (A à Z), et les caractères ASCII de 127 à 255 (0x7f-0xff).

Exemple . Validité des noms de variables

```
SCRIPT LANGAGE="JavaScript">
  var MaVar ;    // Déclaration seule
   var mavar = "Jean";  // déclaration avec affectation initiale
 MaVar = "Paul";  // affectation
  document.write(mavar+', '+MaVar);        // affiche "Jean, Paul"
// 4site = 'pas encore';     // invalide : commence par un nombre
  _4site = 'pas encore';    // valide : commence par un soulign&eacute;
  maïs = 'jaune';      // valide :  le code ASCII de ï est 239.
  document.write( maïs );
  document.write( _4site );
</SCRIPT>
```

Les variables peuvent être de type entier (integer), réel (double), chaîne de caractères (string) , booléen (boolean)., tableau (array), objet (object)

Il est possible de convertir une variable en un type primitif grâce au cast (comme en C, java ou delphi).

Le cast est une conversion de type. L'action de caster (transtyper) consiste à convertir une variable d'un type en un autre.

Exemple :

```
chn=  "12";                // chn vaut la chaîne « 12 »
nbr = parseInt(chn);              // nbr vaut le nombre entier 12
```

Quelques fonctions :

### 2. Constantes

L'utilisateur ne peut définir des constantes dont la valeur est fixée une fois pour toute !

### 3. La valeur sans valeur

La valeur spéciale null représente l'absence de valeur. Une variable avec la valeur NULL n'a pas de valeur. La constante null est **sensible** à la casse.

```
 var mavar= null;
```

Par exemple prompt() renvoie null quand l'utilisateur appuie sur annuler.

### 4. opérateurs divers

#### *a) Opérateurs d'assignement :*

Pour l'affectation on utilise = (pour la comparaison = =)

, *= /=, +=, -=, %= ; .=

```
a=3 ;
x*=y //  équivalent à x=x*y,
```

#### b) Opérateurs de comparaison :

,= = (égalité), < (inférieur strict), <= (inférieur large), >, >=, != (différence)

#### c) Opérateurs 'affectation conditionnelle

Signalons un opérateur très spécial qui équivaut à une structure conditionnelle complexe if then else à la différence qu'il renvoie un résultat de valeur pouvant ne pas être un booléen : l'opérateur ternaire.

Syntaxe :

(condition)?(expression1):(expression2);

Si la condition est vrai alors évalue et renvoie l'expression1 sinon évalue et renvoie l'expression2.

Exemple :

```
nbr = (toto>10)?(toto):(toto%10);
```

Dans cet exemple, la variable ***nbr*** prend ***toto*** pour valeur si ***toto*** est strictement supérieur à 10, sinon vaut le reste de la division entière de ***toto*** par 10.

## E. Types

### 1. Généralités

Même si les variables n'ont pas à être explicitement déclarées, javascript n'en conserve pas moins la notion de type, plus exactement la notion de classe et d'objet.

Une connaissance de la POO (programmation orientée objet) facilite la compréhension de ce qui va suivre. Néanmoins, en première approche, mais vraiment en toute première approche (les puriste vont pousser les haut cris !!), on peut dire à voix basse (et de façon à peine audible) que, au niveau syntaxe, le paramètre principal de la fonction, au lieu de le mettre derrière le nom de la fonction, on l'écrit devant et on met un point comme séparateur.

Selon les fonctions, on utilise » l'une des 3 formes :

#### a) Syntaxe objet

Dans un langage de programmation impératif pour obtenir une sous-chaine extraite de la chaine ***chn*** entre les rangs ***deb*** et ***fin*** , on écrirait : substring(chn,deb,fin), en javascript, on écrit :

```
chn.substring(deb,fin),
```

pour la longueur du mot « bonjour », on écrirait :length("bonjour"), ici

```
"bonjour".length
```

#### b) Syntaxe Classe

Certaines fonctions ont comme syntaxe celle de la méthode de classe : le nom de la classe (attention à la **casse**) est suivi du nom de la méthode ou de l'attribut.

```
a = Math.sqrt(Math.sin(Math.PI / 3))
```

Que l'on peut écrire en mettant Math en « facteur »

```
With (Math){ a=sqrt(sin(PI / 3)) }
```

0.9306048591020996

#### c) Syntaxe impérative

Contrairement à ce qui est écrit plus haut, on n'écrira pas "12".parseInt();

```
nbr = parseInt("12");          // nbr vaut le nombre entier 12
```

### 2. Booléens

C'est le type le plus simple. Un booléen prend  les valeurs de TRUE ou FALSE.

#### a) Opérateurs logiques :

, && (et), , || (ou), ! (non)

### 3. Les nombres

#### a) Les entiers:

Les entiers sont de nombres dont la plage dépend des plate-formes, mais reste équivalente à la portée du type long en C.

On pourra exprimer un entier en décimal, hexadécimal ou encore en octal.

```
decimal = 10;
hexa = 0x0F;
octal = 020;
```

#### b) Les nombres décimaux:

. On peut exprimer ce type sous forme de nombre normal avec un point décimal, ou en notation scientifique.

```
normal = 0.017;
scientifique = 17.0E-3;
```

#### c) Les opérateurs

+ (addition), - (soustraction), * (multiplié), / (divisé), % (modulo), ++ (incrément), --(décrément). Ces deux derniers peuvent être pré ou post fixés

```
i = 5 ;
i++ ;    // i vaut 6 equivalent à i+=1 ou encore à i=i+1
document.write( i++) ; // affiche 6 et ensuite donne à i la valeur 7
document.write( ++i) ; // donne à i la valeur 8 et ensuite affiche 8
```

#### d) Les constantes mathématiques

Math.PI (avec des majuscules à M P I) la valeur de π soit 3.141592653589793

#### e) Les fonctions

| syntaxe | description |
|---|---|
| Math.abs(*nombre*) | la valeur absolue d'un nombre |
| Math.acos(*nombre*) | l'arc cosinus en radian d'un nombre entre -1 et 1 (0 sinon) |
| Math.asin(*nombre*) | l'arc-sinus en radian d'un nombre compris entre -1 et 1 (0 sinon) |
| Math.atan(*nombre*) | l'arc tangente en radian d'un nombre entre -pi/2 et pi/2 |
| Math.ceil(*nombre*) | l'entier le plus proche par valeur supérieure |
| Math.cos(*nombre*) | le cosinus d'un angle en radian |
| Math.exp(*nombre*) | la valeur exponentielle d'une valeur |
| floor(*nombre*) | l'entier le plus proche par valeur inférieure |
| Math.log(*nombre*) | le log de *nombre* |
| max(*nombre1*, *nombre2*) | le maximum entre deux nombres |
| min(*nombre1*, *nombre2*) | le minimum entre deux nombres |
| pow(*base*, *exposant*) | *base* puissance *exposant* |
| Math.random() | une valeur aléatoire entre 0 et 1 |
| round(*nombre*) | l'entier le plus proche de la valeur donnée en argument |
| Math.sin(*angle*) | le sinus de *l'angle* donné en radian |
| Math.sqrt(nombre) | la racine carrée de Math.sqrt(number) |
| Math.tan(nombre) | la tangente du nombre donné en radian |

### 4. Les chaines:

Une chaine est une suite de caractères. Elle peut être délimitée par des guillements simples ou doubles.

```
titi = "toto";
titi = 'toto';
```

```
document.write( "Je m'appelle " + titi);
document.write( 'Je m \'appelle ' + 'titi ');
```

Le premier appel à document.write( va afficher « je m'appelle toto » tandis que le deuxième va afficher « je m'appelle titi ».(indépendamment des guillemets ou apostrophes)

Les séquences d'échappement de javascript:

- \n: Nouvelle ligne
- \t: Tabulation
- \r: Retour chariot
- \\: Anti slash
- \: Signe dollar
- \' : l'apostrophe
- \" : le guillemet

#### *a) L'opérateur de Concaténation*

le + (plus) .

```
document.write( 'Je m'+ "appelle "+titi);  // va afficher « je m'appelle toto » si la variable titi contient
a = "Bonjour " ;
a .+= "monde" ; // équivalent à a = a +  "monde" ; a contient « Bonjour monde »
```

#### *b) Les fonctions*

| syntaxe | description |
|---|---|
| *text.anchor(nom)* | *Crée un signet dans la page HTML* |
| *chaine*.big() | positionne le couple de balises BIG (gros caractères) autour du texte |
| *chaine*.blink() | positionne le couple de balises BLINK (clignotant) autour du texte |
| *chaine*.bold() | positione le couple de balises BOLD (gras) |
| *chaîne*.charAt(*index*) | le caractère *index* de la *chaîne* (commençant à 0) |
| escape(*chaine*) | en caractères ASCII des caractères ISO Latin-1 (& devient %26) |
| *chaine*.fixed() | positionne le couple de balises TT (fixe) autour du texte |
| *chaine*.fontcolor(*couleur*) | la couleur donnée à la chaîne de caractères concernée (équivalent à &tl;FONT COLOR=*couleur*>) |
| *chaine*.fontsize(*taille*) | la taille de police donnée à la chaîne de caractères concernée (équivalent à &tl;FONT size=*taille*>) |
| *chaine1*.indexOf(*chaine2*,[*depuis*]) | la position de chaîne2 recherchée dans chaine1 à partir de la position *depuis*(-1 si pas trouvé) |
| *chaine*.italics() | positionne le couple de balises I (italique) autour du texte |
| *chaine1*.lastIndexOf(*chaine2*, [*depuis*]) | retourne la dernière occurrence de *chaine2* dans *chaine1* à partir de la position *depuis* |
| *texte*.link(*URL*) | crée un lien hypertexte vers URL à partir de *texte* |
| *chaine*.small() | positionne le couple de balises SMALL (petites lettres) autour du texte |
| *chaine*.strike() | positionne le couple de balises STRIKE (biffé) autour du texte |
| *chaine*.sub() | positionne le couple de balises SUB(indice) autour du texte |
| *chaine*.substring(position1, position2) | la chaine de caractères commençant à la position1 et finissant à la position2 |
| *chaine*.sup() | positionne le couple de balises SUP (exposant) autour du texte |
| *chaine*.toLowerCase() | convertit la chaine concernée en minuscule |
| *chaine*.toUpperCase() | convertit la chaine concernée en majuscule |

## 5. Les tableaux:

#### *a) Tableaux simples*

Un tableau est un type de données pouvant contenir plusieurs valeurs, indexées numériquement à partir de ***ZERO***.

Il est possible de préciser ou non le nombre d'éléments

```
var tableau = new  Array(4) ;
```
tout comme :

```
var tableau = new  Array() ;
```
puis :

```
tableau[0] = "titi";
tableau[1] = "toto!";
tableau[3]="tata";
```


Remarquons

- le A majuscule à Array
- Une chaîne de caractères n'est pas un tableau de caractères.

```
jours = new array("Lundi","Mardi","Mercredi","Jeudi", "Vendredi","Samedi","Dimanche");
```

### *b) Tableaux associatifs*

La clé (index) est forcément numérique commençant à 0. L'indexation ne peut se faire autrement

### *c) Tableaux multidimentionnels*

Il s'agit de tableaux de tableaux. Exemple :

```
var toto = new Array( new Array(1,2,3), new Array(4,5,6)) ;
document.write( toto[1][2]);   //  affiche 6  ; les indices (clé) commencent à 0 !

var tata = new Array();
tata [1]=new Array(1,2,3);
tata [2]=new Array(4,5,6) ;
//
document.write( tata[2][2]);   //  affiche 6  ;  ici les indices (clé) commencent toujours à 0 mais la
                               //  ligne 0 n'est pas définie  ni déclarée

//document.write( tata[0][0]);   //  provoque une erreur
//var tata[0]=new Array();     // provoque une erreur grave
tata[0]=new Array();           // maintenant, déclaré mais non défini
document.write( tata[0][0]);   //  affiche undefined
```

### *d) fonctions*

| Méthode | Description | Exemplse |
|---|---|---|
| concat(valeur,...) | concatène des tableaux. | var tableau = new Array('un','deux','trois','quatre','cinq');<br>var tableau2 = new Array('six','sept','huit','neuf','dix');<br>tableau1.concat('tableau2') |
| join(séparateur) | forme une chaîne de caractère à partir du tableau. | var tableau = new Array('push','pop','reverse','shift');<br>tableau.join(', '); |
| s | supprime et retourne le dernier élément du tableau. | var tableau = new Array('o','ko','Mo','Go','To');<br>tableau.pop(); |
| push(valeur,...) | ajoute des éléments à un tableau (NE 4). | var tableau = new Array('0');<br>tableau.push('1','2','3','4'); |
| reverse() | permute les éléments du tableau. | var tableau = new Array('0','1','2','3');<br>tableau.reverse(); |
| shift() | décale les éléments du tableau. | var tableau = new Array('février','mars','avril','mai','juin');<br>tableau.shift(); |
| slice(début, fin) | retourne une partie du tableau. | var tableau = new Array(45,77,20,87,10,32);<br>tableau.slice(1, 2); |
| sort(fonction_tri) | trie les éléments du tableau. | var tableau = new Array('Marc','Angélique','Edith','Annick','Isabelle');<br>tableau.sort(); |
| splice (début, nb_élt_à_effacer, valeur, ...) | insére, supprime ou remplace des éléments du tableau. | var tableau = new Array('a','1','2','&','\?','f','g');<br>tableau.splice(1, 4, 'b', 'c', 'd', 'e'); |
| toString() | convertit un tableau en une chaîne de caractères. | var tableau = new Array('Ceci','est','un','programme','javascript','.')<br>tableau.toString() |
| unshift() | ajoute des éléments au début d'un tableau. | var tableau = new Array('Mardi','Mercredi','Jeudi','Vendredi','Samedi','Dimanche')<br>tableau.unshift('lundi') |

### 6. Les dates:

#### a) *Fonctions*

| *syntaxe* | *description* |
|---|---|
| *date*.getDate() | le jour du mois de la date concernée (entier de 1 à 31) |
| *date*.getDay() | le jour de la semaine de la date concernée (0=dimanche) |
| *date*.getHours() | l'heure de la journée de la date concernée (0 à 23) |
| *date*.getMinutes() | la minute de l'heure de la date concernée (0 à 59) |
| *date*.getMonth() | le mois de la date concernée (0 à 11) |
| *date*.getSeconds() | la seconde de la minute de la date concernée (0 à 59) |
| *date*.getTime() | l'heure dans la valeur numérique correspondante |
| *date*.getTimezoneOffset() | la différence (en minutes) entre l'heure courante et l'heure GMT |
| *date*.getYear() | le nombre d'années écoulées depuis 1900 |
| Date.parse(*chainedate*) | le nombre de millisecondes depuis le 01/01/1970 |
| *date*.setDate(*valeur*) | positionne le jour du mois de la date concernée (entier de 1 à 31) |
| *date*.setDay(*valeur*) | positionne le jour de la semaine de la date concernée (0=dimanche) |
| *date*.setHours(*valeur*) | positionne l'heure de la journée de la date concernée (0 à 23) |
| *date*.setMinutes(*valeur*) | positionne la minute de l'heure de la date concernée (0 à 59) |
| *date*.setMonth(*valeur*) | positionne le mois de la date concernée (0 à 11) |
| *date*.setSeconds(*valeur*) | positionne la seconde de la minute de la date concernée (0 à 59) |
| *date*.setTime(*valeur*) | positionne l'heure dans la valeur numérique correspondante |
| *date*.setYear(*valeur*) | positionne le nombre d'années écoulées depuis 1900 |
| *chainedate*.toGMTString() | convertit une date en chaîne de caractères en suivant les conventions GMT (Mar, 01 Mar 1996 20:00:00 GMT) |
| *chainedate*.toLocaleString() | convertit une date en chaîne de caractères en suivant les conventions locales (03/01/96 20:00:00) |
| Date.UTC(*an*, *mois*, *jour* [, *h*] [, *min*] [, *sec*]) | le nombre de secondes écoulées depuis le 01/01/1970 0h0mn |

## F. Les structures de contrôle

Elles ressemblent fortement à celle de C, Java, Javascript. On se contente donc de donner ici une série d'exemples

### 1. Conditionnelle

#### a) *Le Si*

Syntaxes : ***if (***test1***)*** instructions1 ***else*** instructions2

Elseif n'existe pas en javascript, il faut imbriquer les if. else est facultatif les parenthèses autour des tests sont obligatoires. Si les instructions sont multiples, il faut les mettre entre accolades {}.

 Attention : coller else et if provoque une erreur grave.

```
a = 3;
b = 5;
if (a > b)
   document.write( "a est plus grand que b");
 else if (a == b)
   document.write( "a est &eacute;gal &agrave; b");
 else
   document.write( "a est plus petit que b");
```

dans l'exemple ci-dessous, 2 instructions sont exécutées si le test est valide, on les met donc entre accolades {} et il n'y a pas de clause elseif ni else.

```
if (a > b) {
```

```
    document.write( "a est plus grand que b");
    b = a;
}
```

```
if (a == 5){
    document.write( "a &eacute;gale 5");
    document.write( "...");
 } else if (a == 6){
    document.write( "a &eacute;gale 6");
    document.write( "!!!");
} else
    document.write( "a ne vaut ni 5 ni 6");
```

### *b) Le Cas*

```
switch (i) {
    case 0:
       document.write( "i &eacute;gale 0");
       break;
    case 1:
       document.write( "i &eacute;gale 1");
       break;
    case 2:
       document.write( "i &eacute;gale 2");
       break;
}
```

est équivalent à

```
if (i == 0) {
    document.write( "i &eacute;gale 0");
}
if (i == 1) {
    document.write( "i &eacute;gale 1");
}
if (i == 2) {
    document.write( "i &eacute;gale 2");
}
```

 le break est obligatoire,par exemple :

```
i=1;
switch (i) {
    case 0:
       document.write( "i &eacute;gale 0");
    case 1:
       document.write( "i &eacute;gale 1");
    case 2:
       document.write( "i &eacute;gale 2");
}
```

affiche

i égale 1i égale 2

## 2. Boucles

### *a) Boucle pour*

Syntaxe : ***for*** **(**expr1**;** expr2**;** expr3**)** instruction

La première expression (expr1) est évaluée (exécutée), quoi qu'il arrive au début de la boucle.

Au début de chaque itération, l'expression expr2 est évaluée. Si l'évaluation vaut TRUE, la boucle continue et l'instruction est exécutée. Si l'évaluation vaut FALSE, l'exécution de la boucle s'arrête.

A la fin de chaque itération, l'expression expr3 est évaluée (exécutée).

expr1; expr2; expr3 sont facultatifs

```
// exemple 1
for (i = 1; i <= 10; i++) {
   document.write( i);
}
// exemple 2
for (i = 1;;i++) {
   if (i > 10) {
      break;
   }
   document.write( i);
}
// exemple 3
i = 1;
for (;;) {    // boucle infinie
   if (i > 10) {
      break;    // rupture de boucle
   }
   document.write( i);
   i++;
}
/* exemple 4 : le même que le 3 mais avec
contraction de 2 lignes */
i = 1;
for (;;) {
   if (i > 10) {
      break;
   }
   document.write( i++);
}
// exemple 5
for (i = 1; i <= 10; document.write(i++)) ;
// exemple 6
for (i = 0; i <= 9; document.write( ++i)) ;
```

Les 5 exemples affichent tous 12345678910

 Attention : l'exemple 1 est de loin le meilleur ! Il ne faut surtout pas abuser des contractions comme dans les exemples 5 et 6 qui rendent les programmes illisibles et dificilement maintenables

#### *b) Pour dans*

```
var toto = new Array(1,7,2,6,4,5,8,9);
for (var i in toto) document.write(i+" : "+ toto[i]+" , ");
```

Affiche : 0 : 1 , 1 : 7 , 2 : 2 , 3 : 6 , 4 : 4 , 5 : 5 , 6 : 8 , 7 : 9 ,

#### *c) Boucle tant que*

Syntaxe :

While (test) instruction ;

La boucle n'exécute qu'une seule instruction, pour en exécuter plusieurs, il faut les mettre entre accolades { }

```
i = 1;
while (i <= 10)   document.write( i++);
```

Affiche 12345678910

### 3. Rupture et continuité de boucle

Break permet de quiter immédiatement une boucle :

```
for (;;) {
   if (i > 10) {
      break;
   }
   document.write( i++);
}
```

continue permet de passer à l'itération suivante sans exécuter le reste des instructions :

```
var toto = new Array(1,7,2,6,4,5,8,9);
for (var i in toto) {
  if ( (toto[i] % 2) !=0 ) continue; // évite les valeurs paires
   document.write( toto[i]+" , ");
}
```

Affiche : 2 , 6 , 4 , 8 ,

On obtient le même résultat avec :

```
var toto = new Array(1,7,2,6,4,5,8,9);
for (var i in toto) {
  if ( (toto[i] % 2) = 0 )   document.write( toto[i]+" , ");
}
```

## G. Les fonctions

Syntaxe :

***function*** nom **(**arg_1, arg_2, ..., arg_n**) {**
  corps de fonction
  ***return*** retval**;**
**}**


On peut déclarer les fonctions n'importe où dans le programme. Cependant une bonne règle consiste à les déclarer avant de les utiliser (Ne pas suivre l'exemple ci-dessous)

### 1. Passage des arguments par valeur

La valeur des paramètres n'est pas modifié :

```
<html>
<body>
<SCRIPT LANGAGE="JavaScript">
<!--

c=2;
document.write( incr(c,3));
document.write(" , ");
document.write( c);
```

```
function incr(a,b){
  a += b;  // a = a + b
  return a ;
}

// -->
</SCRIPT>

</body>
</html>
```

Affiche: 5 , 2

Il es préférable de déclarer les fonctions dans la partie HEAD :

```
<html>
<head>
<SCRIPT LANGAGE="JavaScript">
<!--
function incr(a,b){
  a += b;  // a = a + b
  return a ;
}
// -->
</SCRIPT>
</head>
<body>
<SCRIPT LANGAGE="JavaScript">
```

```
<!--

c=2;
document.write( incr(c,3));
document.write(" , ");
document.write( c);

// -->
</SCRIPT>

</body>
</html>
```

Remarque On aurait pu se contenter de mettre  function incr(a,b){ return a+b ;}

### 2. Passage des arguments par adresse (par variable)


Le passage par adresse n'existe pas en javascript

### 3. Arité variable

Javascript définit la classe Fonction et les paramètres des fonctions sont rangés dans un tableau ***arguments*** et le nombre d'éléments dans ***length***

```
<html>
<head>
<SCRIPT LANGAGE="JavaScript">
<!--

function inc(){
  if (inc.arguments.length==1)
    return inc.arguments[0] +1 ;
  else
    return inc.arguments[1] + inc.arguments[0];
}// -->
</SCRIPT>
</head>
<body>
<SCRIPT LANGAGE="JavaScript">
<!--

c=2;
 document.write( inc(c,3)+"<br />");
 document.write( inc(c)+"<br />");
 document.write( c +"<br />");
// -->
</SCRIPT>

</body>
</html>
```


Remarquons que le transtypage entier → chaîne est automatique.

Autre exemple :

```
function somme(){
var res = 0;
  for (var i = 0; i < somme.arguments.length ; i++) res +=somme.arguments[i] ;
     return res;
}// -->
```

## H. Les procédures

C'est la même syntaxe que pour les fonctions on ne met pas de ligne ***return***

## I. Portée des variables

La portée d'une variable dépend du contexte dans lequel la variable est définie. Pour la majorité des variables, la portée concerne la totalité d'un script javascript. Mais, lorsque vous définissez une fonction, la portée d'une variable définie dans cette fonction est locale à la fonction. Par exemple:

```
var a = 1; // portée globale avec ou sans var
function test1() {
    document.write( "test1, a vaut :"+a+"<br />");
}
function test2() {
  a = 2;  // portée globale
    document.write( "test2, a vaut :"+a+"<br />");
}
function test3() {
 var a = 3;  // portée locale
    document.write( "test3, a vaut :"+a+"<br />");
}
function test4() {
  b = 4; // portée globale
    document.write( "test4, b vaut :"+b+"<br />");
}
function test5() {
 var c = 5;  // portée locale
    document.write( "test5, c vaut :"+c+"<br />");
}

document.write( "programme, a vaut :"+a+"<br
/>");
test1();
document.write( "programme, a vaut :"+a+"<br
/>");
test2();
document.write( "programme, a vaut :"+a+"<br
/>");
test3();
document.write( "programme, a vaut :"+a+"<br
/>");
test4();
document.write( "programme, b vaut :"+b+"<br
/>");
test5();
document.write( "programme, c vaut :"+c+"<br
/>"); // pas d'affichage jusqu'à la fin ==> erreur
```

affiche le message :
programme, a vaut :1
test1, a vaut :1
programme, a vaut :1
test2, a vaut :2
programme, a vaut :2

```
test3, a vaut :3
programme, a vaut :2
test4, b vaut :4
programme, b vaut :4
test5, c vaut :5
```

## J. Classes et Objets

Javascript permet la poo (programmation orientée objet) comme java C++ php ou Delphi Il est possible de définir des classes et d'instancier les objets correspondants

Cette partie suppose une connaissance des langages orientés objet. On ne donnera que la syntaxe sur l'exemple très classique des carrés et rectangles

- Le constructeur porte le nom de la classe (***function*** et non ***class***)
- Le mécanisme d'héritage n'existe pas en javascript
- ***This*** fait référence, dans une classe, à l'objet correspondant à l'instance de cette classe

```
<html>
<head>
SCRIPT LANGAGE="JavaScript">
  function rectangle(Long,larg){  // on définit ici
la classe
// définitions des fonctions et procédure
   function perimetre(){
    return 2*(this.Lng + this.larg) ;
   }
   function surface (){
    return this.Lng * this.larg ;
   }
   function double(){
    this.Lng *= 2;
    this.larg *= 2;
   }
// affectation des attributs
   this.Lng = Long;
   this.larg = larg;
// affectations des méthodes
   this.surf = surface;
   this.dble =double;
   this.perimetre=perimetre;
  }
</SCRIPT>
</head>
<body>
```

```
<SCRIPT LANGAGE="JavaScript">

var rect=new rectangle (3,5);

document.write(rect.Lng);  // 3
document.write("<br>");
document.write(rect.larg);  // 5
document.write("<br>");
document.write(rect.surf());  //  15
document.write("<br>");
document.write(rect.perimetre()); //  16
document.write("<br>");

rect.dble();

document.write(rect.Lng);  // 6
document.write("<br>");
document.write(rect.larg);  // 10
document.write("<br>");
document.write(rect.surf());  // 60
document.write("<br>");
document.write(rect.perimetre());  // 32
document.write("<br>");

</SCRIPT>
</body>
</html>
```

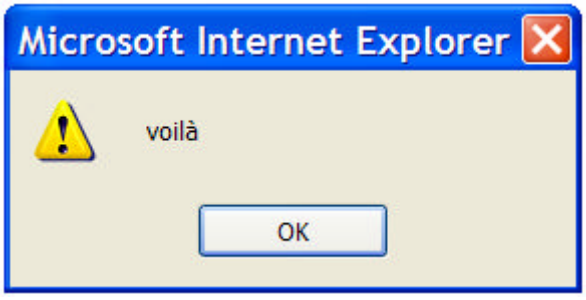
## K. Entrées/Sorties

### 1. sorties

Pour obtenir un affichage dans le navigateur , on utilise ***document.write()*** avec parenthèses obligatoires.

On peut aussi faire ouvrir une fenêtre :

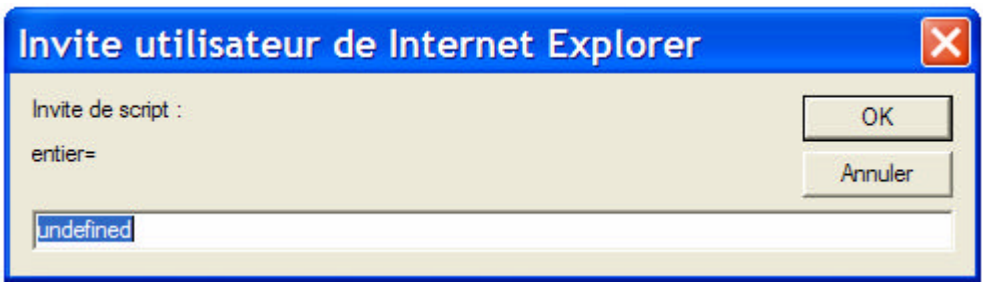
```
alert('voilà');
```



### 2. entrées

Pour que le JavaScript obtienne des informations de l'utilisateur, on peut :

- Soit récupérer un élément quelquonque se trouvant dans la page y compris les éléments d'un formulaire (voir plus loin)
- Soit faire ouvrir une fenêtre :




```
x=prompt('entier=',0);
alert(x);
```

dans ce cas, si l'utilisateur clique sur :

- ok, x reçoit la valeur contenue dans la ligne, y compris ***undefined***
- Annuler, x reçoit la valeur ***null***

Remarque ***prompt*** admet 1 ou 2 paramètres, le deuxième est facultatif et représente la valeur affichée à l'ouverture de la fenêtre. En cas d'absence de ce paramètre le mot ***undefined*** apparaît

- Soit pour une réponse binaire :

```
alert(confirm("d'accord ?"));
```

***confirm*** renvoie vrai ou faux selon que l'utilisateur clique sur ***ok*** ou ***annuler***

## L. Fichiers

La lecture et l'écriture dans un fichier local ou distant poserait un sérieux problème de sécurité !!!!!

La seule chose qu'il est possible d'enregistrer ou de lire sur le disque de l'utilisateur, ce sont des Cookies

## M. Inclusion de fichiers

Pour inclure un fichier javascript dans une page html, on ajoute l'attribut src suivi de l'URL (relative ou absolue) du fichier à inclure. Ce fichier ne doit contenir que du code javascript.

```
<Script Language="JavaScript" src="xxx.js">
// description du fichier xxx
</script>
```

## N. Evénements : traitement

L'utilisateur du navigateur déclanche un Evenement en opérant soit un clic sur un bouton, un lien, une option d'une liste déroulante ou à cocher un bouton radio etc.. soit en déplaçant la souris ,etc..

Voici les principaux événements et quelques exemples de balises :

| *Evénement* | *Se produit lorsque ...* | *Exemples de balises HTML supportant l'événement* |
|---|---|---|
| onBlur | un champ de saisie d'un formulaire perd le focus | <INPUT> <SELECT> <TEXTAREA><TEXT> |
| onClick | l'utilisateur clique avec la souris l'objet HTML considéré | <INPUT> <SELECT> <A> |
| onChange | l'utilisateur change le contenu d'un objet d'un formulaire HTML | <INPUT> <SELECT> <TEXTAREA> |

| | | |
|---|---|---|
| onFocus | un champ de saisie d'un formulaire prend le focus | <INPUT> <SELECT> <TEXTAREA> |
| onLoad | le document HTML est complètement chargé | <BODY> |
| onMouseOut | la souris quitte la zone d'un lien hypertexte | <A ... > |
| onMouseOver | la souris rentre dans la zone d'un lien hypertexte | <A ... > |
| onReset | Appui sur le bouton reset | <INPUT> |
| onSelect | un choix dans un liste est sélectionné | <INPUT> <SELECT > |
| onSubmit | un formulaire est soumis au serveur HTTP | <FORM> |
| onUnload | on s'apprête à changer de document HTML | <BODY> |

L'événement est capté par l'objet sur lequel il s'est produit et si le programme indique ce qu'il faut faire au cas où celui-ci se présente, le code prévu à cet effet se'exécute.

```
<html>
  <head>
    <TITLE>Question/réponse</TITLE>
<SCRIPT LANGAGE="JavaScript">
function reponse() {
  alert('bonjour ' + document.maforme.leprenom.value);
  return true
}
function tst() {
  alert('bonjour ' + document.maforme.lenom.value);
  return true
}
</SCRIPT>
  </head>
  <body>
    <form name="maforme">
      Votre nom : <INPUT type="text"  name="lenom" value="" onBlur="tst();">
      Votre prénom : <INPUT type="text"  name="leprenom" value="">
      <INPUT type="submit" value="Valider" onclick="reponse();">
    </form>
 </body>
</html>
```

## O. Evénements : simulation

Il est possible de simuler la génération d'un événement.

En principe la méthode qui génère l'événement porte le même nom que celle qui la gère mais sans le « on », c'est à dire :

***Blur() ; focus() ; click(); submit();*** etc...

Dans l'exemple suivant on simule l'action de l'utilisateur qui consiste à placer le focus sur le champ mdp

```
<html>
  <head>
    <TITLE>Vérification d'identité</TITLE>
  </head>
  <body onLoad="window.document.maforme.mdp.focus();">
    <form name="maforme">
      Votre nom : <INPUT type="text"  name="lenom" value="">
      Votre mot de passe : <INPUT type="password"  name="mdp" value="">
      <INPUT type="submit" value="Valider" >
```

```
  </form>
 </body>
</html>
```

Dans les 2 exemples suivants on envoie automatiquement le formulaire ci-dessus

On simule l'appui sur valider

```
<html>
 <head>
  <TITLE>Vérification d'identité</TITLE>
 </head>
 <body onLoad="window.document.maforme.ok.click();">
  <form name="maforme">
   Votre nom : <INPUT type="text"  name="lenom" value="toto">
   Votre mot de passe : <INPUT type="password"  name="mdp" value="titi">
   <INPUT name="ok" type="submit" value="Valider" >
  </form>
 </body>
</html>
```

ou on simule l'envoi du formulaire, les champs n'apparaissent plus sur la page

```
<html>
 <head>
  <TITLE>Vérification d'identité</TITLE>
 </head>
 <body onLoad="window.document.maforme.submit();">
  <form name="maforme">
   Votre nom : <INPUT type="hidden"  name="lenom" value="toto">
   Votre mot de passe : <INPUT type="hidden"  name="mdp" value="titi">
     </form>
 </body>
</html>
```

***Remarque*** : D'un point de vue sécurité, ce n'est pas une bonne idée que de mettre le mot de passe en clair.

## P. Objets prédéfinis

Le schéma suivant, que l'on retrouve souvent dans la littérature, illustre la hiérarchie des classes JS pour le Navigateur :

```
window
  |
  + --parent, frames, self, top
  |
  + --location
  |
  + --historique
  |
  + --document
       |
       + --formulaire (FORM)
       |     |
       |     + --elements (text fields, textarea, checkbox, password
       |              radio, select, button, submit, reset)
       + --links
       |
       + -- URL
```

### 1. Exemple très important :

Le programme ci-dessous est une mine de renseignements, il dresse la liste des propriétés des objets de la feuille :

```
<html>
 <body>
  <form name="maforme">
    Votre nom : <INPUT type="text"  name="lenom" value="" onBlur="tst();">
    <INPUT type="submit" value="Valider" onclick="reponse();">
  </form>
  <SCRIPT LANGAGE="JavaScript">
  <!--
    function show_props(obj,obj_name){
        var result="";
        for(var i in eval(obj)) result+= obj_name+"."+i+" = "+eval(obj)[i]+"<br />\n";
        return result;
     }
   x=prompt("propriétés de l'objet : ","window.document");
   alert(  show_props(x,x));
   document.write(  show_props(x,x));
  // -->
  </SCRIPT>
 </body>
</html>
```

à essayer avec la proposition du prompt ***window.document*** , repérer un object et réessayer avec lui par exemple ***window.document.maforme*** puis ***window.document.maforme.lenom*** puis etc..

Remarque

- le formulaire n'est là que pour observer ses propriétés
- la fonction eval qui comme son nom l'indique évalue l'objet passé en paramètre exemple :

```
eval("5 +  3")
```

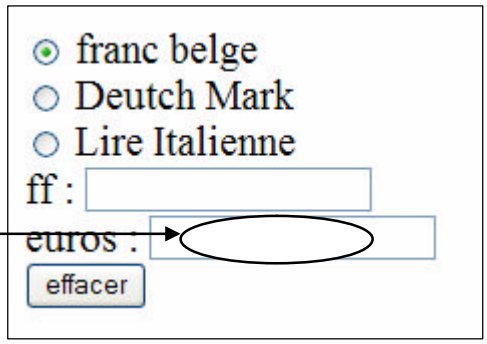
### 2. Les tableaux prédéfinis

Les principaux objets d'une page HTML peuvent être nommés mais ils sont aussi définis par des tableaux que l'on retrouve dans la liste suivante :

| Array | Description |
|---|---|
| anchors | comprend toutes les balises <A> qui contiennent un argument NAME |
| applets | comprend toutes les balises <APPLET> du document |
| arguments | comprend tous les arguments d'une fonction |
| elements | comprend toutes les balises <FORM> dans l'ordre de leur définition |
| embeds | comprend toutes les balises <EMBED> dans l'ordre de leur définition |
| forms | comprend toutes les balises <FORM> du document |
| frames | comprend toutes les balises <FRAME> contenant un FRAMESET dans l'ordre de leur définition |
| history | comprend tous l'historique du navigateur |
| images | comprend toutes les balises <IMG> dans l'ordre de leur définition |
| links | comprend tous les liens <AREA HREF="...">, <A HREF=""> et les objets Link créés par la méthode linkwith the link |
| mimeTypes | Comprend tous les types MIME supportés par le navigateur (helpers ou plug-ins) |
| options | comprend tous les éléments OPTION d'une balise SELECT |
| plugins | comprend tous les plug-ins installés sur votre navigateur |

Ainsi dans le programme suivant :

```
<html>
 <head> </head>
 <body>
  <form name="calc">
   <input type="radio" name="monaie" > franc belge <br />
   <input type="radio" name="monaie""> Deutch Mark <br />
   <input type="radio" name="monaie" > Lire Italienne <br />
   ff :   <input type="text" name="franc""> <br />
   euros : <input type="text" name="euro"> <br />
   <input type="button" value="effacer">
  </form>
 </body>
</html>
```




pour accéder à
on a le choix : soit par indexation (la numérotation commence à **zéro** !!) :

```
window.document.forms[0].elements[4].value
```

soit par nommage :

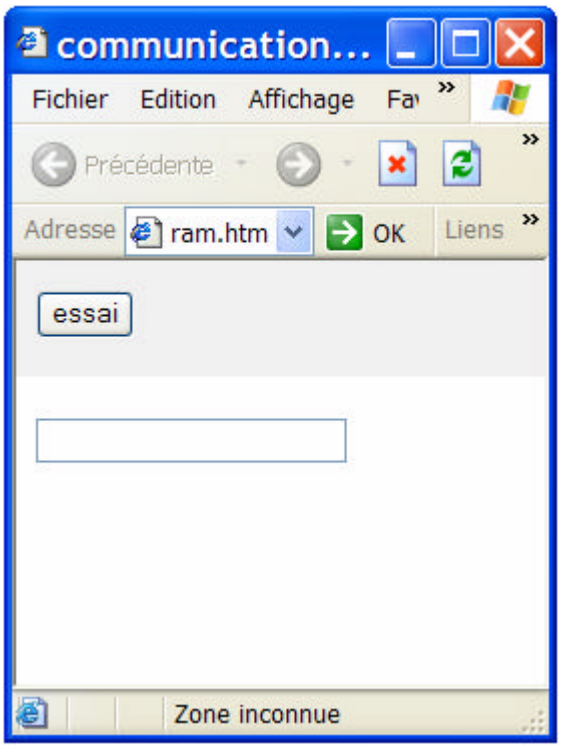
```
window.document.calc.euro.value
```

soit par combinaison

```
window.document.calc.elements[4].value
```

### 3. Exemple : communication entre frames

L'exemple ci dessois montre la hiérarchie entre les différents documents.




Le fichier suivant est le fichier principal et définit les frames :

```
<HTML>
 <HEAD>
  <TITLE> communication entre frames </TITLE>
 </HEAD>
 <FRAMESET rows="20%, 50%" border="5" frameborder="5"
bordercolor="#000000">
    <FRAME SRC="haut.html" name="haut">
    <FRAME SRC="" name="bas">
 <NOFRAMES>
  Votre navigateur ne supporte pas les frames. Voici un acces a
       <A HREF="fichier3.html">notre site sans frames</A>
 </NOFRAMES>
  </FRAMESET>
</HTML>
```

ci-dessous le fichier haut.html et ci contre le résultat :

```
<HTML>
 <HEAD>
  <TITLE>Mon site en frames</TITLE>
      <SCRIPT LANGUAGE="JavaScript">
      <!--
            parent.bas.document.writeln('<form>');
            parent.bas.document.writeln('<INPUT TYPE="text" >');
            parent.bas.document.writeln('');
            parent.bas.document.writeln('</form>');
            parent.bas.document.close() ;
            function recupere(){
                  alert(parent.bas.document.forms[0].elements[0].value)
            }
```

```
        //-->
        </SCRIPT>
 </HEAD>
 <body bgcolor="#f0f0f0">
 <INPUT TYPE="button" name="essai" onClick="recupere()" value="essai">
 </body>
</HTML>
```

 Lorsqu'on définit 2 cadres, il faut logiquement 3 fichiers (voir plus haut). Ici, le cadre du bas ne contenent rien initialement, n'est défini par aucun fichier.

Ce qui apparaît dans le cadre inférieur (nommé ***bas*** ) est créé par le script contenu dans le fichier du cadre supérieur (nommé ***haut*** ) le bouton situé dans ce cadre récupère le contenu de l'élément « zone texte » du cadre ***bas***

### 4. Exercice :

Voici un convertisseur de monnaies nationales européenne :

```
<html>
 <head>
  <SCRIPT LANGAGE="JavaScript">
  <!--
   var taux = 6.55957;
   function eurofr(){
    window.document.calc.franc.value =window.document.calc.euro.value * taux;
   }
   function freuro(){
    window.document.calc.euro.value =window.document.calc.franc.value / taux;
   }
   function modif(tx,ch){ // ch est une chaine de 2 caractères
    taux = tx;
    var chn =window.document.calc.outerHTML;
    var pos = (chn.indexOf(window.document.calc.franc.outerHTML));
    window.document.calc.outerHTML=chn.substring(1,pos - 5) +ch + chn.substring(pos-
3,chn.length);
     eurofr();
   }
  // -->
  </SCRIPT>
 </head>
 <body>
  <form name="calc">
   <input type="radio" name="monaie" checked  onClick="modif(6.55957,'FF');"> franc français <br />
   <input type="radio" name="monaie" onClick="modif(40.3399,'FB');"> franc belge <br />
   <input type="radio" name="monaie" onClick="modif(1.95583,'DM');"> Deutch Mark <br />
   <input type="radio" name="monaie" onClick="modif(166.386,'Pt');"> pesetas espagnole <br />
   <input type="radio" name="monaie" onClick="modif(1936.27,'Ll');"> Lire Italienne <br />
   ff :   <input type="text" name="franc" onkeyup ="freuro()"> <br />
   euros : <input type="text" name="euro" onkeyup ="eurofr()" )"> <br />
   <input type="button" value="effacer"
onClick="window.document.calc.franc.value='';window.document.calc.euro.value='';">
  </form>
 </body>
</html>
```

Le tester, l'étudier et le modifier de façon à :

- Le rendre plus esthétique

- Admettre 3 caractères au lieu de 2 pour les monaies
- augmenter le nombres de monaies convertibles :
  - DEMParEuros = 1.95583;  allemagne
  - ESPParEuros = 166.386;  espagne
  - FRFParEuros = 6.55957;  france
  - IEPParEuros =  0.787564; Ireland
  - ITLParEuros =  1936.27;  Italie
  - BEFParEuros =  40.3399;  Belgique
  - NLGParEuros =  2.20371;  Hollande
  - ATSParEuros =  13.7603;  Autriche
  - PTEParEuros =  200.482;  Portugal
  - FIMParEuros =  5.94573;  Finlanded
  - GRDParEuros = 340.750;  Grece
  - LUFParEuros = 40.3399;Luxambourg

# V. Feuilles de style

## A. La création de style

La déclaration de style se fait dans l'entête de la page, encadrée par les balises <STYLE> et </STYLE>.
Il y a 3 possibilités pour affecter un style

- modifier le style d'une balise existante
- créer une nouvelle classe de style
- affecter l'attribut style

Les balises existantes
Voici la syntaxe pour affecter la police arial en 12 points à toutes les cellules des tableaux d'une page.
Toutes les propriétés de style sont détaillées dans le dernier paragraphe.

```
<HTML><HEAD>
<STYLE>
  TD {font-family:arial;font-size:12pt}
</STYLE>
</HEAD>
<BODY>
  Ici le corps de la page
</BODY></HTML>
```

Dans l'exemple ci-dessus, toutes les cellules TD auront comme style par défaut Arial 12pt.
On peut modifier ainsi toutes les balises existantes.

Le cas particulier de la balise de lien A
La balise <A> est particulière, car elle peut être dérivée selon l'action du visiteur. On obtient ainsi 3 balises A.

```
A:hover {text-decoration:underline}
A:active {color:red}
A:visited {color:green}
```

Avec cette déclaration, le texte de lien :
- est souligné au passage de la souris,
- devient rouge sur un clic,
- reste vert pour indiquer que ce lien a été visité.

 ATTENTION : Netscape 4.X ne supporte pas cette fonctionnalité. Plus généralement, il n'est pas possible de changer le style d'un objet HTML une fois la page chargée dans Netscape.

Création de classes
Modifier les balises existantes n'est pas suffisant quand il s'agit d'affecter un style à une partie de texte. Par exemple dans cette page, les parties de code sont en courier new vert, grâce à la classe de style "code" qui est déclarée ainsi :

```
<HTML><HEAD>
<STYLE>
  .code {font-family:courrier;color:green}
</STYLE>
</HEAD>
```

Le point devant code indique la création d'une nouvelle classe appelée code.

Pour affecter ce style à une portion de texte, on écrit :

```
<SPAN class="code">texte mis en forme</SPAN>
```

La balise <SPAN> permet d'affecter à un groupe de mot une classe de style.

Il est aussi possible d'affecter ce style à une balise existante. Pour donner le style courier new vert à une cellule de tableau, on écrit :

```
<TD class="code">cellule en police courrier vert</TD>
```

Affecter l'attribut style
Il n'est pas indispensable de créer une classe de style pour affecter un style à un objet. Par exemple :

```
<SPAN style="font-family:courier;color:green">Texte mis en forme</SPAN>
```

L'attribut style d'une balise reçoit les propriétés de style.
Il est toutefois préférable de déclarer une classe pour regrouper la mise en forme.

## B. Les feuilles de style externes

Nous avons vu jusqu'ici comment affecter un style à une page, ce qui implique de recopier la déclaration dans toutes les pages et rend les mises à jour longues et périlleuses.
Il est possible de regrouper la déclaration des styles dans un fichier externe et de l'appeler dans toutes les pages du site. La maintenance est accélérée et toutes les pages du site sont homogènes.

### 1. Le fichier externe

L'extension d'un fichier de feuille de style est toujours .css
Le fichier css contient uniquement l'intérieur des balises <STYLE>.
Voici un exemple de fichier de style appelé style.css :

```
/* Ceci est un commentaire */
TD {font-family:arial;font-size:12pt}
.code {font-family:courrier;color:green}
```

Le fichier style.css ne contient pas les balises <STYLE>.
Comme en javascript, les commentaires sont marqués par // ou /* */.

### 2. L'intégration du fichier externe

Une ligne dans l'entête suffit pour appeler le fichier de style dans une page :

```
<HTML><HEAD>
  <LINK rel="StyleSheet" type="text/css" href="style.css">
</STYLE>
</HEAD>
```

La balise <LINK> fait un lien entre la page en cours et le fichier de style externe.

## C. Les propriétés de style

Nous avons vu comment déclarer les styles dans une page. La norme CSS a prévu de nombreuses propriétés pour les différents objets HTML. Toutes ces propriétés ne sont pas prises en compte dans tous les navigateurs. Netscape 4.X en particulier est assez en retard par rapport à Internet Explorer 4 et 5, qui lui-même ne respecte pas toutes les normes.

### 1. Les propriétés de police

Normes respectées par Internet Explorer et Netscape.

| Propriété | Signification | Valeurs possibles |
|---|---|---|
| font-size | Hauteur de police en points ou pixels | font-size:12px<br>font-size:10pt |
| font-family | Nom de police | font-family:arial,verdana<br>font-family:sans serif |
| font-weight | Epaisseur de la police (gras) | font-weight:bold<br>font-weight:normal |
| font-style | Style de police (italic) | font-style:italic<br>font-style:normal |
| color | Couleur de police | color:green<br>color:#FFCC00 |

### 2. Les propriétés de texte

Normes respectées par Internet Explorer et Netscape.

| *Propriété* | *Signification* | *Valeurs possibles* |
|---|---|---|
| text-height | Hauteur de ligne en points ou pixels | text-height:12px |
| text-decoration | Attribut de soulignement | text-decoration:overline \| line-through \| underline \| none |
| text-align | Alignement de paragraphe (balise P) | text-align:left \| :right \| :center |
| text-indent | Indentation de paragraphe (balise P) | text-indent:25px \| -15px |
| text-transform | Transformation des majuscules | text-transform:capitalize\|:uppercase \| lowercase |

### 3. Les propriétés d'arrière-plan

Seul Internet Explorer 4 et 5 permet de choisir une image d'arrière plan.

| *Propriété* | *Signification* | *Valeurs possibles* |
|---|---|---|
| background | Image d'arrière-plan | background:image.gif |
| background-color | Couleur d'arrière-plan | background-color:black |

### 4. Les propriétés de positionnement

Netscape 4.X a un gestion particulière des propriétés de positionnement.

| *Propriété* | *Signification* | *Valeurs possibles* |
|---|---|---|
| clip | Clipping | clip:auto |
| width | Largeur de la zone | width:auto width:150 |
| height | Hauteur de la zone | height:auto height:100 |
| top | Ordonnée du coin supérieur gauche | top:10 |
| left | Abscisse du coin supérieur gauche | left:400 |
| overflow | Recouvrement | overflow:clip overflow:scroll overflow:none |
| position | Type de positionnement | position:absolute position:relative position:static |
| visibility | Visibilité | visibility:hidden visibility:visible |

### 5. Les propriétés d'encadrement

Aucune de ces propriétés n'est reconnue par Netscape 4.X
Internet Explorer respecte presque complètement l'affichage de ces propriétés.

| *Propriété* | *Signification* | *Valeurs possibles* |
|---|---|---|
| margin-top | Marge du haut en pixels | margin-top:10 |
| margin-right | Marge du bas en pixels | margin-right:15 |
| margin-bottom | Marge du haut en pixels | margin-bottom:15 |
| margin-left | Marge de gauche en pixels | margin-left:10 |
| margin | Groupement des 4 marges haute, droite, bas et gauche | margin:10,15,15,10 |
| padding-top | Espacement en haut | padding-top:10 |

| | | |
|---|---|---|
| padding-right | Espacement à droite | padding-right:10 |
| padding-bottom | Espacement en bas | padding-bottom:10 |
| padding-left | Espacement à gauche | padding-left:10 |
| padding | Groupement des espacements en haut, à droite, en bas et à gauche | padding:10,10,10,10 |
| border-style | Style de contour | none, solid, double, groove, ridge, inset, outset |
| border-color | Couleur de contour | border-color:red |
| border-width | Largeur de contour | border-width:3px<br>thin, medium, thick |
| border-left-width | Epaisseur du trait de contour gauche | border-left-width:thin |
| border-right-width | Epaisseur du trait de contour droit | border-right-width:medium |
| border-top-width | Epaisseur du trait de contour haut | border-top-width:thick |
| border-bottom-width | Epaisseur du trait de contour bas | border-bottom-width:0px |
| float | Flottaison du bloc<br>A éviter | float:right<br>float:left |

## 6. Exemples

```
<html>
 <body>
  <STYLE>
   TH {font-size:large; color: #3B3DE3; background-color: #EEEEEE}
   /* alternance de couleurs pour les lignes des tables*/
   TR.A0 {background-color: #CCCCCC}
   TR.A1 {background-color: #DDDDDD}
  </STYLE>
  <table>
   <TR>
    <TH>Titre colonne 1</TH>
    <TH>Titre colonne 2</TH>
   </TR>
   <TR CLASS='A0' >
    <TD align=center width=150>cellule 1 de la ligne 1</TD>
    <TD align=center width=250>cellule 2 de la ligne 1</TD>
   </TR>
   <TR CLASS='A1' >
    <TD align=center width=150>cellule 1 de la ligne 2</TD>
    <TD align=center width=250>cellule 2 de la ligne 2</TD>
   </TR>
   <TR CLASS='A0' >
    <TD align=center width=150>cellule 1 de la ligne 3</TD>
    <TD align=center width=250>cellule 2 de la ligne 3</TD>
   </TR>
   <TR CLASS='A1' >
    <TD align=center width=150>cellule 1 de la ligne 4</TD>
    <TD align=center width=250>cellule 2 de la ligne 4</TD>
   </TR>
  </table>
```

```
  </body>
</html>
```

donne le tableau suivant :

| Titre colonne 1 | Titre colonne 2 |
|---|---|
| cellule 1 de la ligne 1 | cellule 2 de la ligne 1 |
| cellule 1 de la ligne 2 | cellule 2 de la ligne 2 |
| cellule 1 de la ligne 3 | cellule 2 de la ligne 3 |
| cellule 1 de la ligne 4 | cellule 2 de la ligne 4 |

Dans l'exemple suivant, le style supprime les ascenseurs de la page et du textarea et le javascript fait occuper tout l'écran à la page :

```
<HTML>
  <HEAD>
    <TITLE> Exemple 9</TITLE>
    <style>
      .noScrollBar{
         background-color : #FFFFFF;
         scrollbar-track-color : #FFFFFF;
         scrollbar-arrow-color : #FFFFFF;
         scrollbar-base-color : #FFFFFF;
         scrollbar-face-color : #FFFFFF;
         scrollbar-shadow-color : #FFFFFF;
         scrollbar-darkshadow-color : #FFFFFF;
         scrollbar-highlight-color : #FFFFFF;
      }
    </style>
  </HEAD>

  <BODY CLASS="noScrollBar" onLoad ="window.resizeTo(screen.availWidth,screen.availHeight);
                                     moveTo(0,0);browser.statusBar=false;">
    Il suffit ensuite d'appeller la balise TEXTAREA ainsi :
    <TEXTAREA CLASS="noScrollBar" rows="7" cols="50" >
       Vous avez vu, il n'y a pas de barre de défilement  !!!
     </TEXTAREA>
  </BODY>
</HTML>
```

# VI. Applet java

## A. Généralités

On donne ici à titre d'exemple, la façon d'intégrer du code java compilé (pcode et non du JavaScript interprété)

Cette façon de faire permet d'accélérer le fonctionnement et aussi de protéger son code

## B. Syntaxe

On utilise les balises applet

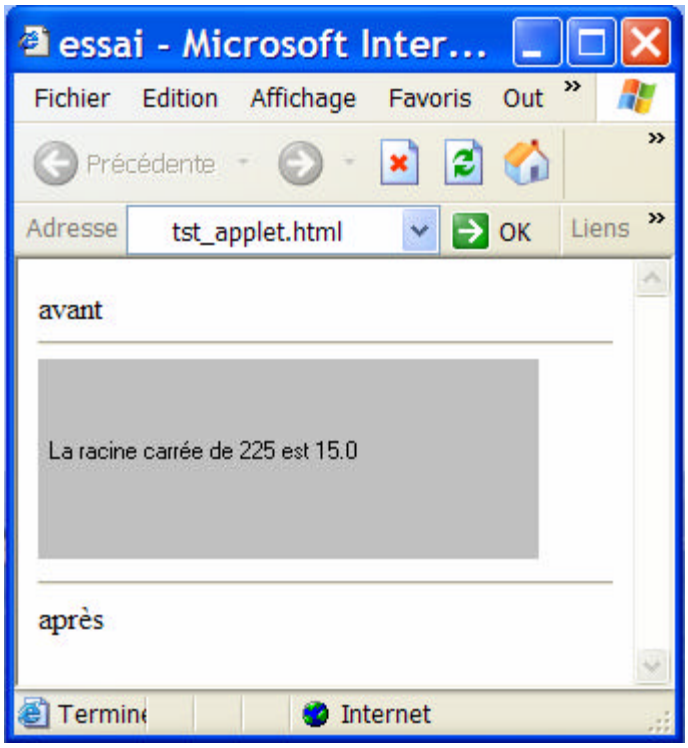
```
<applet code="xxx.class"
height=100 width =250>
<param name="yyy" value="zzz">
</applet>
```

exemple :

le code :

```
<html>
  <head>
    <title>essai</title>
  </head>
  <body>
    avant
    <hr>
    <applet code="Essai2.class"
        height=100 width =250>
        <param name="nbr" value="100">
    </applet>
   <hr>
  après
  </body>
</html>
```

donne :




ou "***Essai2.class***" résulte de la compilation de "***Essai2.java***" :

```
import java.awt.*;
public class Essai2 extends java.applet.Applet {
   int nombre;
   public void init( ) {
         super.init();
      String s = getParameter("nbr");
      if (s!=null)
            nombre = Integer.parseInt(s);
      else
        nombre=0;
   }
   public void start() {
         super.start();
   }
   public void paint(Graphics screen) {
         screen.drawString("La racine carré de " +
                           nombre +
                           " est " +
                           Math.sqrt(nombre), 5, 50);
   }
}
```

## C. Passage de paramètres

On on utilise les balises <***param***> et les attributs ***name*** et ***value***. Il faut bien entendu connaître le nom des paramètres utilisés dans le code ***Java***

# VII. Php

## A. Généralités

Le PHP est né en 1994, Rasmus Lerdorf possède une page web avec son CV. Celui-ci développe un programme pour garder une trace de ses visiteurs: PHP.
Rapidement, des internautes curieux lui demandent une copie de son programme, et c'est ainsi que Rasmus met en ligne la version 1.0 de PHP. Très vite, sa création connait un grand succès, et de nombreuses suggestions lui parviennent.

C'est un langage incrusté au HTML et interprété (PHP3) ou compilé (PHP4) côté serveur. Il dérive du C et du Perl dont il reprend la syntaxe. Il est extensible grâce à de nombreux modules et son code source est ouvert. Comme il supporte tous les standards du web et qu'il est gratuit, il s'est rapidement répandu sur la toile.

En 1997, PHP devient un projet collectif et son interpréteur est réécrit par Zeev Suraski et Andi Gutmans pour donner la version 3 qui s'appelle désormais PHP : Hypertext Preprocessor (acronyme récursif à l'exemple du système Open Source Linux : Is Not UniX).

Il existe par ailleurs des applications web prêtes à l'emploi (PHPNuke, PHP SPIP, PHPSlash…) permettant de monter facilement et gratuitement son portail. En juillet 2000 plus de 300.000 sites tournaient déjà sous PHP !

Intégration d'un script dans une page

Les pages web sont au format html. Les pages web dynamiques générées avec PHP4 sont au format php. Le code source php est directement inséré dans le fichier html grâce au conteneur de la norme XML :

**<?php … ?>**
Exemple:

```
<html>
<body>
<?php
        echo ''Bonjour'';
?>
</body>
</html>
```

Exemple commenté : ( le célèbre « hello world »)

*Exemple de script, code source (**côté serveur**) :*

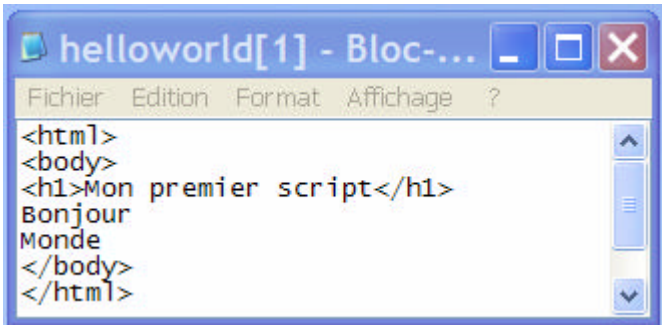
```
<html>
<body>
<h1>Mon premier script</h1>
<?php echo"Bonjour\nMonde\n";?>
```

*Autre écriture du même script :*

```
<?php
echo "<html>\n<body>\n";
echo "<h1>Mon premier script</h1>\n";
echo "Bonjour\nMonde\n";
```

*Résultat affiché par le navigateur :*


*Code source (**côté client**) de la page essai.ph3 résultant du script*




L'exemple ci-dessus montre :

- Que le code source vu par l'utilisateur n'est pas celui écrit par le programmeur : les 2 codes coté serveur donne le même résultat coté client
- Que l'on peut écrire du Php dans du html et réciproquement
- Que **\n** provoque un passage à la ligne dans le code source client mais pas dans le navigateur (il faudrait utiliser <br />

## B. Commentaires

Un script php se commente comme en C ou comme en Shell:

Exemple :

```
<?php
// commentaire de fin de ligne
/* commentaire
sur plusieurs
lignes */

# commentaire de fin de ligne
?>
```

Tout ce qui se trouve dans un commentaire est ignoré. Il est conseillé, comme toujours, de commenter largement ses scripts.

## C. Constantes, Variables, types

Le code php s'écrit dans une page Html entre les balises :

```
< ?php
?>
```

Ce code est traduit par le serveur en code html et est envoyé au client pour être interprété par le navigateur.

### 1. variables

Le typage des variables est implicite en php. Il n'est donc pas nécessaire de déclarer leur type au préalable ni même de les déclarer avant leur utilisation.

**Le nom des variables est sensible à la casse (ie : $x != $X).**

Les noms de variables suivent les mêmes règles de nommage que les autres entités PHP. Un nom de variable valide doit commencer par une lettre ou un souligné (_), suivi de lettres, chiffres ou soulignés.

Une lettre peut être une des lettres minuscules (a à z) ou majuscules (A à Z), et les caractères ASCII de 127 à 255 (0x7f-0xff).

Les identificateurs de variable sont précédés du symbole « $ » (dollars). Exemple : $toto.

Exemple . Validité des noms de variables

```
<?php
   $var = "Jean";
 $Var = "Paul";
 echo "$var, $Var";       // affiche "Jean, Paul"
// $4site = 'pas encore';    // invalide : commence par un nombre
 $_4site = 'pas encore';   // valide : commence par un soulign&eacute;
 $maïs = 'jaune';     // valide :  le code ASCII de ï est 239.
 echo $maïs ;
 echo $_4site ;
?>
```

Les variables peuvent être de type entier (integer), réel (double), chaîne de caractères (string) , booléen (boolean)., tableau (array), objet (object)

Il est possible de convertir une variable en un type primitif grâce au cast (comme en C, java ou delphi).

Le cast est une conversion de type. L'action de caster (transtyper) consiste à convertir une variable d'un type en un autre.

Exemple :

```
$chn=  "12";              // $chn vaut la chaîne ''12''
$nbr = (int)$chn;                  // $nbr vaut le nombre entier 12
```

Quelques fonctions :

- empty($var) : renvoie vrai si la variable est vide
- isset($var) : renvoie vrai si la variable existe
- unset($var) : détruit une variable
- gettype($var) : retourne le type de la variable
- settype($var, "type ") : convertit la variable en type type (cast)
- is_long(), is_double(), is_string(), is_array(), is_object(), is_bool(), is_float(), is_numeric(), is_integer(), is_int()…

### 2. Nommage dynamique des variables

Une variable peut avoir pour identificateur la valeur d'une autre variable.

Syntaxe : ${$var} = valeur;

Exemple :

```
$toto = "titi";
${$toto} = 123;
echo $titi;                // la variable $titi vaut 123
```

La portée d'une variable est limitée au bloc dans lequel elle a été crée. Une variable locale à une fonction n'est pas connue dans le reste du programme. Tout comme une variable du programme n'est pas connue dans une fonction. Une variable créée dans un bloc n'est pas connue dans les autres blocs, mêmes supérieurs.

### 3. Constantes

L'utilisateur peut définir des constantes dont la valeur est fixée une fois pour toute. Les constantes ne portent pas le symbole $ (dollars) en début d'identificateur et ne sont pas modifiables.

define(''var'',valeur) : définit la constante var (sans $) de valeur valeur

Exemple 1 :

```
Define("toto","titi");
echo toto;              // affiche 'titi'
```

Exemple 2 :

```
define(année,2003);
 echo année;            // affiche 2003
```

 Contrairement aux variables, les identificateurs de constantes (et aussi ceux de fonction) ne sont **pas sensibles à la casse**.

### 4. Références

On peut à ,la manière des pointeurs en C, faire référence à une variable grâce à l'opérateur & (ET commercial).

Exemple 1 :

```
$toto = 100;      // la variable $toto est initialisée à la valeur 100
$titi = &$toto; // la variable $titi fait référence à $toto
$toto++;                  // on change la valeur de $toto
echo $titi;       // qui est répercutée sur $titi qui vaut alors 101
```

Exemple 2 :

```
function change($var) {
        $var++;           // la fonction incrémente en local l'argument
}
$nbr = 1;         // la variable $nbr est initialisée à 1
change(&$nbr);          // passage de la variable par référence
echo $nbr;      // sa valeur a donc été modifiée
```

### 5. La valeur sans valeur

La valeur spéciale NULL représente l'absence de valeur. Une variable avec la valeur NULL n'a pas de valeur. La constante NULL est insensible à la casse.

```
<?php
  $var = Null;
?>
```

### 6. opérateurs divers

#### *a) Opérateurs d'assignement :*

Pour l'affectation on utilise = (pour la comparaison = =)

, *= /=, +=, -=, %= ; .=

```
$a=3 ;
$x*=$y //  équivalent à $x=$x*$y),
```

#### *b) Opérateurs de comparaison :*

,== (égalité), < (inférieur strict), <= (inférieur large), >, >=, != (différence)

#### *c) Opérateurs 'affectation conditionnelle*

Signalons un opérateur très spécial qui équivaut à une structure conditionnelle complexe if then else à la différence qu'il renvoie un résultat de valeur pouvant ne pas être un booléen : l'opérateur ternaire.

Syntaxe :

(condition)?(expression1):(expression2);

Si la condition est vrai alors évalue et renvoie l'expression1 sinon évalue et renvoie l'expression2.

Exemple :

```
$nbr = ($toto>10)?($toto):($toto%10);
```

Dans cet exemple, la variable $nbr prend $toto pour valeur si $toto est strictement supérieur à 10, sinon vaut le reste de la division entière de $toto par 10.

### 7. Variables prédéfinies

| *Signification* | *Nom (Php4)* | *Ancien nom (Php3)* |
|---|---|---|
| Variables de serveur : | $_SERVER | $HTTP_SERVER_VARS |
| Variables d'environnement | $_ENV | $HTTP_ENV_VARS. |
| HTTP Cookies | $_COOKIE | $HTTP_COOKIE_VARS. |
| HTTP GET variables | $_GET | $HTTP_GET_VARS. |
| HTTP POST variables | $_POST | $HTTP_POST_VARS. |
| Variable de téléchargement de fichier via HTTP | $_FILES | $HTTP_POST_FILES. |
| Variables de requête | $_REQUEST Un tableau associatif constitué du contenu des variables $_GET, $_POST, $_COOKIE, et $_FILES. | pas d'équivalent. |
| Session variables | $_SESSION | $HTTP_SESSION_VARS. |
| Variables globales | $GLOBALS Un tableau associatif contenant les références sur toutes les variables globales actuellement définies dans le contexte d'exécution global du script. Les noms des variables sont les index du tableau. | idem |

## D. Types

Même si les variables n'ont pas à être explicitement déclarées, PHP n'en conserve pas moins la notion de type

### 1. Booléens

C'est le type le plus simple. Un booléen prend les valeurs de TRUE ou FALSE.

#### *a) Opérateurs logiques :*

and, && (et), or, || (ou), xor (ou exclusif), ! (non)

### 2. Les nombres

#### *a) Les entiers:*

Les entiers sont de nombres dont la plage dépend des plate-formes, mais reste équivalente à la portée du type long en C. Sur une plate-forme 32 bits, les valeurs vont de -2 147 483 648 à +2 147 483 647. En cas de dépassement de capacité, PHP convertit automatiquement l'entier incriminé en nombre décimal.

On pourra exprimer un entier en décimal, hexadécimal ou encore en octal.

```
$decimal = 10;
$hexa = 0x0F;
$octal = 020;
```

### b) Les nombres décimaux:

308 à 1.7E+308. On peut exprimer ce type sous forme de nombre normal avec un point décimal, ou en notation scientifique.

```
$normal = 0.017;
$scientifique = 17.0E-3;
```

### c) Les opérateurs

+ (addition), - (soustraction), * (multiplié), / (divisé), % (modulo), ++ (incrément), --(décrément). Ces deux derniers peuvent être pré ou post fixés

```
$i = 5 ;
$i++ ;    // i vaut 6 equivalent à i+=1 ou encore à i=i+1
echo $i++ ; // affiche 6 et ensuite donne à i la valeur 7
echo ++$i ; // donne à i la valeur 8 et ensuite affiche 8
```

### d) Les constantes mathématiques

| Constante | Valeur | Description |
|---|---|---|
| M_PI | 3.14159265358979323846 | Pi |
| M_E | 2.7182818284590452354 | e |
| M_LOG2E | 1.4426950408889634074 | log_2 e |
| M_LOG10E | 0.43429448190325182765 | log_10 e |
| M_LN2 | 0.69314718055994530942 | log_e 2 |
| M_LN10 | 2.30258509299404568402 | log_e 10 |
| M_PI_2 | 1.57079632679489661923 | pi/2 |
| M_PI_4 | 0.78539816339744830962 | pi/4 |
| M_1_PI | 0.31830988618379067154 | 1/pi |
| M_2_PI | 0.63661977236758134308 | 2/pi |
| M_SQRTPI | 1.77245385090551602729 | sqrt(pi) [4.0.2] |
| M_2_SQRTPI | 1.12837916709551257390 | 2/sqrt(pi) |
| M_SQRT2 | 1.41421356237309504880 | sqrt(2) |
| M_SQRT3 | 1.73205080756887729352 | sqrt(3) [4.0.2] |
| M_SQRT1_2 | 0.70710678118654752440 | 1/sqrt(2) |
| M_LNPI | 1.14472988584940017414 | log_e(pi) [4.0.2] |
| M_EULER | 0.57721566490153286061 | Euler constant [4.0.2] |

### e) Les fonctions

| Nom | Signification |
|---|---|
| Abs | Valeur absolue |
| Acos | arc cosinus |
| acosh | Arc cosinus hyperbolique |
| Asin | arc sinus |
| asinh | Arc sinus hyperbolique |
| Atan2 | arc tangent de deux variables |
| Atan | arc tangent |
| atanh | Arc tangeant hyperbolique |
| base_convert | Convertit un nombre entre des bases arbitraires. |
| BinDec | Convertit de binaire en décimal |

| | |
|---|---|
| Ceil | Arrondit au nombre supérieur |
| Cos | cosinus |
| cosh | Cosinus hyperbolic |
| DecBin | Convertit de décimal en binaire |
| DecHex | Convertit de décimal en hexadécimal |
| DecOct | Convertit de décimal en octal |
| deg2rad | Convertit un nombre de degrés en radians |
| Exp | exponentielle |
| expm1 | Returns exp(number) - 1, computed in a way that is accurate even when the value of number is close to zero |
| Floor | Arrondi à l'entier inférieur |
| fmod | Returns the floating point remainder (modulo) of the division of the arguments |
| getrandmax | Plus grande valeur aléatoire possible. |
| hexdec | Convertit de hexadécimal en décimal |
| hypot | Returns sqrt( num1*num1 + num2*num2) |
| is_finite | |
| is_infinite | |
| is_nan | |
| lcg_value | Générateur de congruence combinée linéaire |
| Log10 | logarithme en base 10. |
| log1p | Returns log(1 + number), computed in a way that accurate even when the val ue of number is close to zero |
| Log | Logarithme naturel (népérien) |
| max | La plus grande valeur. |
| min | La plus petite valeur. |
| mt_getrandmax | La plus grand valeur aléatoire possible. |
| mt_rand | Génère une meilleure valeur aléatoire. |
| mt_srand | Initialise une meilleure valeur aléatoire |
| OctDec | Convertit d'octal en décimal. |
| pi | Retourne la valeur de pi |
| pow | Puissance |
| rad2deg | Convertit de radians en degrés |
| rand | Génère une valeur aléatoire. |
| round | Arrondi. |
| Sin | Sinus |
| sinh | Sinyus hyperbolique |

| | |
|---|---|
| Sqrt | Racine carrée. |
| srand | Initialise le générateur de nombres aléatoires |
| Tan | Tangente |
| tanh | Tangente hyperbolique |

### 3. Les chaines:

Une chaine est une suite de caractères. Elle peut être délimitée par des guillements simples ou doubles. Attention, leur utilisation ne produira pas le même effet: les chaines entourées de guillemets doubles sont sujettes aux **substitutions** de variables et au traitement des séquences d'échappement, contrairement aux chaines entourées de guillemets simples.

```
<?
$titi = "toto";
echo "Je m'appelle $titi";
echo 'Je m\'appelle $titi ';
?>
```

Le premier appel à echo va afficher « je m'appelle toto » tandis que le deuxième va afficher « je m'appelle $titi ».

Les séquences d'échappement de PHP:

- \n: Nouvelle ligne
- \t: Tabulation
- \r: Retour chariot
- \\: Anti slash
- \$: Signe dollar
- \' : l'apostrophe
- \" : le guillemet

#### *a) L'opérateur de Concaténation*

le . (point) .

```
echo 'Je m'. "appelle ".$titi";  // va afficher « je m'appelle toto »
a$ = "Bonjour " ;
a$ .= "monde" ; // équivalent à a$ = a$ .  "monde" ; a contient « Bonjour monde »
```

#### *b) Les fonctions*

| ***Nom*** | ***Signification*** |
|---|---|
| addcslashes | Ajoute des slashes dans une chaîne, à la mode du langage C |
| addslashes | Ajoute des anti-slashs dans une chaîne |
| bin2hex | Convertit des données binaires en représentation hexadécimale |
| chop | Alias de rtrim() |
| chr | Retourne un caractère spécifique |
| chunk_split | Scinde une chaîne |
| convert_cyr_string | Convertit une chaîne d'un jeu de caractères cyrillique à l'autre |
| count_chars | Retourne des statistiques sur les caractères utilisés dans une chaîne |
| crc32 | Calcule la somme de contrôle CRC32 |

| | |
|---|---|
| crypt | Chiffrage indéchiffrable (hashing) |
| echo | Affiche une chaîne de caractères |
| explode | Coupe une chaîne en segments |
| fprintf | Ecrit une chaîne formatée dans un flôt |
| get_html_translation_table | Retourne la table de traduction des entités utilisée par htmlspecialchars() et htmlentities() |
| hebrev | Convertit un texte logique hébreux en texte visuel |
| hebrevc | Convertit un texte logique hébreux en texte visuel, avec retours à la ligne |
| html_entity_decode | Convertit toutes les entités HTML en caractère normal |
| htmlentities | Convertit tous les caractères éligibles en entités HTML |
| htmlspecialchars | Convertit les caractères spéciaux en entités HTML |
| implode | Rassemble les éléments d'un tableau en une chaîne |
| join | Rassemble les éléments d'un tableau en une chaîne |
| levenshtein | Calcule la distance Levenshtein entre deux chaînes |
| localeconv | Lit la configuration locale |
| ltrim | Supprime les caractères invisibles de début de chaîne |
| md5_file | Calcule le md5 d'un fichier |
| md5 | Calcule le md5 d'une chaîne |
| metaphone | Calcule la clé métaphone |
| money_format | Met un nombre au format monétaire |
| nl_langinfo | Rassemble des informations sur la langue et la configuration locale. |
| nl2br | Insère un retour à la ligne HTML à chaque nouvelle ligne |
| number_format | Formate un nombre pour l'affichage |
| ord | Retourne le code ASCII d'un caractère |
| parse_str | Analyse une requête HTTP |
| print | Affiche une chaîne de caractères |
| printf | Affiche une chaîne de caractères formattée |
| quoted_printable_decode | Convertit une chaîne quoted-printable en chaîne 8 bits |
| quotemeta | Echappe les méta-caractères |
| rtrim | Supprime les espaces de fin de chaîne |
| setlocale | Modifie les informations de localisation |
| sha1_file | Calcule le sha1 d'un fichier |
| sha1 | Calcule le sha1 d'une chaîne de caractères |
| similar_text | Calcule la similarité de deux chaînes |
| soundex | Calcule la clé soundex |
| sprintf | Retourne une chaîne formattée |

| | |
|---|---|
| sscanf | Analyse une chaîne à l'aide d'un format |
| str_ireplace | Version insensible à la casse de str_replace() |
| str_pad | Complète une chaîne jusqu'à une taille donnée |
| str_repeat | Répéte une chaîne |
| str_replace | Remplace toutes les occurrences dans une chaîne |
| str_rot13 | Effectue une transformation rot13 |
| str_shuffle | Mélange les caractères d'une chaîne de caractères |
| str_split | Convertit une chaîne de caractères en tableau |
| str_word_count | Compte le nombre de mots utilisés dans une chaîne |
| strcasecmp | Comparaison de chaînes binaires |
| strchr | Trouve la première occurence d'un caractère dans une chaîne |
| strcmp | Comparaison binaire de chaînes |
| strcoll | Comparaison de chaînes localisée |
| strcspn | Trouve un segment de chaîne ne contenant pas certains caractères |
| strip_tags | Supprime les balises HTML et PHP d'une chaîne |
| stripcslashes | Supprime les anti-slash d'une chaîne C |
| stripos | Rercherche la position d'une occurence dans une chaîne, sans tenir compte de la casse |
| stripslashes | Supprimer les anti-slash d'une chaîne |
| stristr | Trouve la première occurence dans une chaîne (insensible à la casse) |
| strlen | Calcule la taille d'une chaîne |
| strnatcasecmp | Comparaison de chaînes avec l'algorithme d'"ordre naturel" (insensible à la casse) |
| strnatcmp | Comparaison de chaînes avec l'algorithme d'"ordre naturel" |
| strncasecmp | Compare en binaire des chaînes de caractères |
| strncmp | Comparaison binaire des n premiers caractères |
| strpos | Trouve la position d'un caractère dans une chaîne |
| strrchr | Retourne la fin de la chaîne |
| strrev | Inverse une chaîne |
| strripos | Trouve la position de la dernière occurence d'une chaîne dans une autre de façon insensible à la casse. |
| strrpos | Trouve la position de la dernière occurence d'un caractère dans une chaîne |
| strspn | Trouve le premier segment de chaîne |
| strstr | Trouve la première occurence dans une chaîne |
| strtok | Coupe une chaîne en segments |
| strtolower | Renvoit une chaîne en minuscules |
| strtoupper | Renvoit une chaîne en majuscules |
| strtr | Remplace des chaînes dans une chaîne |

| | |
|---|---|
| substr_count | Compte de le nombre d'occurences de segments dans une chaîne |
| substr_replace | Remplace un segment dans une chaîne |
| substr | Retourne un segment de chaîne |
| trim | Supprime les espaces en début et fin de chaîne |
| ucfirst | Met le premier caractère en majuscule |
| ucwords | Met en majuscule la première lettre de tous les mots |
| vprintf | Affiche une chaîne formatée |
| vsprintf | Retourne une chaîne formatée |
| wordwrap | Effectue la césure d'une chaîne |

## 4. Les tableaux:

### a) *Tableaux simples*

Un tableau est un type de données pouvant contenir plusieurs valeurs, indexées numériquement ou à l'aide de chaines.

```
$tableau[0] = "titi";
$tableau[1] = "toto!";
```

 On remarquera que lorsqu'on veut ajouter un élément au tableau, il suffit de faire une affectation dans spécifier d'indice:

```
$tableau[]="tata";
```

Cette technique est aussi valable pour les tableaux dont les valeurs ne sont pas indexées successivement. (i.e si les postes 1, 10, et 3 d'un tableau sont affectées, PHP affectera dans ce cas le poste d'indice 11)

Précision sur les chaines de caractères: notez que celles-ci peuvent tout à fait être considérées comme des tableaux de caractères. Une affectation de type $MaChaine[3]='a'; est tout à fait possible.

```
$jours = array(1=>"Lundi","Mardi","Mercredi","Jeudi", "Vendredi","Samedi","Dimanche");
```

Les tableaux peuvent également, comme nous l'avons dit, utiliser des chaines pour leur indexation. On parle alors de tableaux associatifs. Il est même possible de mélanger indexation numérique et indexation par chaine dans le même tableau.

### b) *Tableaux associatifs*

La clé (index) peut ne pas être numérique

exemple de tableau associatif

```
$mois["Janvier"] = 31;
$mois["Février"] = 28;
$mois["Mars"] = 31;
```

on peut aussi écrire :

```
$mois= array("Janvier" => 31, "Février" => 28 , "Mars" => 31);
```

### c) *Tableaux multidimentionnels*

Il s'agit de tableaux de tableaux. Exemple :

```
$toto = array(array(1,2,3),array(4,5,6)) ;
print $toto[1][2];    //  affiche 6  ; les indices (clé) commencent à 0 !
$toto = array(1 => array(1 => 1,2,3),array(1 => 4,5,6)) ;
print $toto[2][3];    //  affiche 6  ;  ici les indices (clé) commencent à 1 !
```

### *d) fonctions*

| *Nom* | *Signification* |
|---|---|
| array_change_key_case | Change la casse des clés du tableau |
| array_chunk | Sépare un tableau en tableaux de taille inférieure |
| array_combine | Creates an array by using one array for keys and another for its values |
| array_count_values | Compte le nombre de valeurs dans un tableau |
| array_diff_assoc | Calcule la différence de deux tableaux, en prenant en compte les clés |
| array_diff | Calcule la différence entre deux tableaux |
| array_fill | Remplis un tableau avec une même valeur |
| array_filter | Filtre les éléments d'un tableau |
| array_flip | Remplace les clés par les valeurs, et les valeurs par les clés |
| array_intersect_assoc | Calcule l'intersection de deux tableaux avec des tests sur les index |
| array_intersect | Calcule l'intersection de tableaux |
| array_key_exists | Vérifie si une clé existe dans un tableau |
| array_keys | Retourne toutes les clés d'un tableau |
| array_map | Applique sur fonction sur des tableaux |
| array_merge_recursive | Combine plusieurs tableaux ensembles, récursivement |
| array_merge | Fusionne plusieurs tableaux |
| array_multisort | Tri multi-dimensionnel |
| array_pad | Complète un tableau jusqu'à la longueur spécifiée, avec une valeur. |
| array_pop | Dépile un élément de la fin d'un tableau |
| array_push | Empile un ou plusieurs éléments à la fin d'un tableau |
| array_rand | Prend une ou plusieurs valeurs, au hasard dans un tableau |
| array_reduce | Réduit itérativement un tableau |
| array_reverse | Renverse l'ordre des éléments d'un tableau |
| array_search | Recherche dans un tableau la clé associée à une valeur |
| array_shift | Dépile un élément au début d'un tableau |
| array_slice | Extrait une portion de tableau |
| array_splice | Efface et remplace une portion de tableau |
| array_sum | Calcule la somme des valeurs du tableau |
| array_unique | Dédoublonne un tableau |
| array_unshift | Empile un ou plusieurs éléments au début d'un tableau |
| array_values | Retourne les valeurs d'un tableau |
| array_walk | Exécute une fonction sur chacun des membres d'un tableau. |
| array | Crée un tableau |

| | |
|---|---|
| arsort | Trie un tableau en ordre inverse |
| asort | Trie un tableau en ordre |
| compact | Crée un tableau contenant les variables et leur valeur |
| count | Compte le nombre d'éléments d'un tableau |
| current | Transforme une variable en tableau |
| each | Retourne chaque paire clé/valeur d'un tableau |
| end | Positionne le pointeur de tableau en fin de tableau |
| explode | Coupe une chaîne en segments |
| extract | Importe les variables dans la table des symboles |
| implode | Rassemble les éléments d'un tableau en une chaîne |
| in_array | Indique si une valeur appartient à un tableau |
| is_array | Détermine si une variable est un tableau |
| join | Rassemble les éléments d'un tableau en une chaîne |
| key | Retourne une clé d'un tableau associatif |
| krsort | Trie un tableau en sens inverse et suivant les clés |
| ksort | Trie un tableau suivant les clés |
| list | Transforme une liste de variables en tableau |
| natcasesort | Tri d'un tableau avec l'algorithme à "ordre naturel" insensible à la casse |
| natsort | Tri d'un tableau avec l'algorithme à "ordre naturel" |
| next | Avance le pointeur interne d'un tableau |
| pos | Retourne l'élément courant d'un tableau |
| prev | Recule le pointeur courant de tableau |
| preg_split | Rassemble les éléments d'un tableau en une chaîne |
| range | Crée un tableau contenant un intervalle d'éléments |
| reset | Remet le pointeur interne de tableau au début |
| rsort | Trie en ordre inverse |
| shuffle | Mélange les éléments d'un tableau |
| sizeof | Alias de count() |
| sort | Trie un tableau |
| split – | Scinde une chaîne en un tableau, grâce à une expression régulière. |
| uasort | Trie un tableau en utilisant une fonction de comparaison personnalisée |
| uksort | Trie un tableau par ses clés en utilisant une fonction de comparaison définie par l'utilisateur |
| usort | Trie un tableau en utilisant une fonction de comparaison personnalisée |

### *e) Exemple*

```
$toto = array("abc", "def", "ghij", "klmn", "opqrst", "uvwx");
```

```
$titi = each($toto);
print_r($titi);     //   affiche  Array ( [1] => abc [value] => abc [0] => 0 [key] => 0 )
echo '<br />';
$titi = each($toto);
print_r($titi);     //   affiche  Array ( [1] => def [value] => def [0] => 1 [key] => 1 )
echo '<br />';
$titi = each($toto);
print_r($titi);     //   affiche  Array ( [1] => ghij [value] => ghij [0] => 2 [key] => 2 )
echo '<br />';
$titi = each($toto);
print_r($titi);     //   affiche  Array ( [1] => klmn [value] => klmn [0] => 3 [key] => 3 )
echo '<br />';

reset($toto);    // réinitialisation du pointeur d'éléments du tableau $toto
$titi = each($toto);
print_r($titi);     //   affiche  Array ( [1] => abc [value] => abc [0] => 0 [key] => 0 )
echo '<br />';

print($titi[0].' , '.$titi[1]);     //   affiche  3 , klmn
echo '<br />';
print($titi[key].' , '.$titi[value]);     //   affiche  3 , klmn
echo '<br />';

list($a,$b,,$c)=$toto;

print($b);     //   affiche    def
echo '<br />';

list(,$d)=$titi ;
print($d);     //   affiche    klmn
echo '<br />';
```

## E. Les structures de contrôle

Elles ressemblent fortement à celle de C, Java, Javascript. On se contente donc de donner ici une série d'exemples

 PHP propose 2 façons de rassembler des instructions à l'intérieur d'un bloc, pour les fonctions de contrôle if, while, for, foreach et switch soit :

- En encadrant les instructions par des accolades {}
- En remplaçant l'accolade d'ouverture par deux points (:) et l'accolade de fermeture par, respectivement, endif;, endwhile;, endfor;, ou endswitch;.

### 1. Conditionnelle

#### *a) Le Si*

Syntaxes :

- ***if*** **(**test1**)** instructions1 ***elseif*** **(**test2**)** instructions2 ***else*** instructions3
- ***if*** **(**test1**) :** instructions1 ***elseif*** **(**test2**) :** instructions2 ***else*** instructions3 ***endif***

Elseif et else sont facultatifs les parenthèses autour des tests sont obligatoires. Si les instructions sont multiples, il faut les mettre entre accolades {}.

```
$a = 3;
```

```
$b = 5;
if ($a > $b)
   print "a est plus grand que b";
 elseif ($a == $b)
   print "a est &eacute;gal &agrave; b";
 else
   print "a est plus petit que b";
```

dans l'exemple ci-dessous, 2 instructions sont exécutées si le test est valide, on les met donc entre accolades {} et il n'y a pas de clause elseif ni else.

```
if ($a > $b) {
   print "a est plus grand que b";
   $b = $a;
}
```

Autre syntaxe :

```
if ($a == 5):
   print "a &eacute;gale 5";
   print "...";
elseif ($a == 6):
   print "a &eacute;gale 6";
   print "!!!";
else:
   print "a ne vaut ni 5 ni 6";
endif;
```

#### *b) Le Cas*

```
switch ($i) {
   case 0:
      print "i &eacute;gale 0";
      break;
   case 1:
      print "i &eacute;gale 1";
      break;
   case 2:
      print "i &eacute;gale 2";
      break;
}
```

est équivalent à

```
if ($i == 0) {
   print "i &eacute;gale 0";
}
if ($i == 1) {
   print "i &eacute;gale 1";
}
if ($i == 2) {
   print "i &eacute;gale 2";
}
```


le break est obligatoire,par exemple :

```
$i=1;
switch ($i) {
   case 0:
      print "i &eacute;gale 0";
   case 1:
      print "i &eacute;gale 1";
   case 2:
      print "i &eacute;gale 2";
}
```

affiche

i égale 1i égale 2

### 2. Boucles

#### *a) Boucle pour*

Syntaxe : for (expr1; expr2; expr3) instruction

La première expression (expr1) est évaluée (exécutée), quoi qu'il arrive au début de la boucle.

Au début de chaque itération, l'expression expr2 est évaluée. Si l'évaluation vaut TRUE, la boucle continue et l'instruction est exécutée. Si l'évaluation vaut FALSE, l'exécution de la boucle s'arrête.

A la fin de chaque itération, l'expression expr3 est évaluée (exécutée).

expr1; expr2; expr3 sont facultatifs

```
// exemple 1
for ($i = 1; $i <= 10; $i++) {
    print $i;
}
// exemple 2
for ($i = 1;;$i++) {
    if ($i > 10) {
        break;
    }
    print $i;
}
// exemple 3
$i = 1;
for (;;) {    // boucle infinie
    if ($i > 10) {
        break;    // rupture de boucle
    }
    print $i;
    $i++;
}
/* exemple 4 : le même que le 3 mais avec la
syntaxe alternative (sans {}) et contraction de
2 lignes */
$i = 1;
for (;;) :
    if ($i > 10) {
        break;
    }
    print $i++;
endfor;
// exemple 5
for ($i = 1; $i <= 10; print $i, $i++) ;
// exemple 6
for ($i = 1; $i <= 10; print $i++) ;
// exemple 7
for ($i = 0; $i <= 9; print ++$i) ;
 // exemple 8
for ($i = 1; $i <= 10; print $i, ++$i) ;
```

Les 8 exemples affichent tous 12345678910

### *b) Boucle pour chaque*

Syntaxes :

foreach(array_expression as $value) commandes

foreach(array_expression as $key => $value) commandes

La première forme passe en revue le tableau array_expression. A chaque itération, la valeur de l'élément courant est assignée à $value et le pointeur interne de tableau est avancé d'un élément (ce qui fait qu'à la prochaine itération, on accédera à l'élément suivant).

La deuxième forme fait exactement la même chose, mais c'est la clé de l'élément courant qui est assigné à la variable $key.

Lorsque foreach démarre, le pointeur interne de fichier est automatiquement ramené au premier élément du tableau. Cela signifie que vous n'aurez pas à faire appel à reset() avant foreach.

 Note : De plus, notez que foreach travaille sur une copie du tableau spécifié, et pas sur le tableau lui-même. Par conséquent, le pointeur de tableau n'est pas modifié, comme il le serait avec le fonction each(), et les modifications faites dans le tableau ne seront pas prises en compte dans le tableau original.

```
 $toto = array("abc", "def", "ghij", "klmn", "opqrst", "uvwx");
 foreach ($toto as $valeur) {
    echo "Valeur: $valeur<br>\n";
 }
foreach ($toto as $cle => $valeur) {
   echo "Cl&eacute;: $cle; Valeur: $valeur<br>\n";
 }
```

affiche :

Valeur: abc
Valeur: def
Valeur: ghij
Valeur: klmn
Valeur: opqrst
Valeur: uvwx

Clé: 0; Valeur: abc
Clé: 1; Valeur: def
Clé: 2; Valeur: ghij
Clé: 3; Valeur: klmn
Clé: 4; Valeur: opqrst
Clé: 5; Valeur: uvwx

Tableau multidimmentionnels Exemple:

Avec :

```
$toto = array(1 => array(1 => 1,2,3),array(1 => 4,5,6)) ;
```

```
foreach ($toto as $valeur)
  echo "Valeur: $valeur<br>\n";
```

donne :

Valeur: Array
Valeur: Array

il faut 2 foreach :

```
foreach ($toto as $tablo)
  foreach ($tablo as $valeur)
    echo "Valeur: $valeur ,  \n";
```

Valeur: 1 , Valeur: 2 , Valeur: 3 , Valeur: 4 , Valeur: 5 , Valeur: 6 ,

### *c) Boucle tant que*

Syntaxes :

While (test) instruction ;
While (test)  : instruction1 ; instruction2; ...; instructionP endwhile;

Dans la première version, la boucle n'exécute qu'une seule instruction, pour en exécuter plusieurs, il faut les mettre entre accolades { }

```
$i = 1;
while ($i <= 10)   print $i++;
```

Affiche 12345678910

### *d) Boucle Faire tant que*

Les boucles `do..while` ressemblent beaucoup aux boucles `while`, mais l'expression est testée à la fin de chaque itération plutôt qu'au début. La principale différence par rapport à la boucle `while` est que la première itération de la boucle `do..while` est toujours exécutée (l'expression n'est testée qu'à la fin de l'itération), ce qui n'est pas le cas lorsque vous utilisez une boucle `while` (l'expression est vérifiée au début de chaque itération).

Exemple :

```
$i = 5;
do
  print $i--; // post décrémenté
 while ($i>0);
echo '<br>';
$i = 5;
do {
  print --$i;    // prédécrémenté
} while ($i>0);
```

Affichent :

54321
43210

### 3. Rupture et continuité de boucle

Break permet de quitter immédiatement une boucle :

```
for (;;) :
   if ($i > 10) {
      break;
   }
   print $i++;
endfor;
```

continue permet de passer à l'itération suivante sans exécuter le reste des instructions :

```
$toto=array(1,7,2,6,4,5,8,9);
while (list ($cle, $valeur) = each ($toto)) {
  if (!($valeur % 2)) { // évite les valeurs paires
     continue;
  }
  print("$cle : $valeur , ");
}
```

Affiche : 0 : 1 , 1 : 7 , 5 : 5 , 7 : 9 ,

## F. Les fonctions

Syntaxe :

***function*** nom ***(***$arg_1, $arg_2, ..., $arg_n***) {***
  corps de fonction
  ***return*** $retval**;**
***}***

- On peut déclarer les fonctions n'importe où dans le programme. Cependant une bonne règle consiste à les déclarer avant de les utiliser (Ne pas suivre l'exemple ci-dessous)
- On peut donner une valeur par défaut aux arguments

### 1. Passage des arguments se fait par valeur

La valeur des paramètres n'est pas modifié :

```
$c=2;
print incr($c,3);
 print incr($c);
print $c;

function incr($a,$b=1){
  $a += $b;  // $a = $a + $b
  return $a ;
}
```

Affiche : 532

 Remarque On aurai pu se contenter de mettre function incr($a,$b=1){ return $a+$b ;}

### 2. Passage des arguments se fait par adresse (par variable)

On place un **&** devant le paramètre et sa valeur s'en trouve modifiée

```
function incr2(&$a,$b=1){
  $a += $b;  // $a = $a + $b
  return $a ;
}

 $c=2;
```

```
 print incr2($c,3);
 print incr2($c);
print $c;
```
affiche : 566

## G. Les procédures

C'est la même syntaxe que pour les fonctions on ne met pas de ligne ***return***

## H. Portée des variables

La portée d'une variable dépend du contexte dans lequel la variable est définie. Pour la majorité des variables, la portée concerne la totalité d'un script PHP. Mais, lorsque vous définissez une fonction, la portée d'une variable définie dans cette fonction est locale à la fonction. Par exemple:

```
$a = 1; /* portée globale */
function test() {
   echo " a vaut :";
   echo $a; /* portée locale */
}
```
affiche le message :
a vaut :
La valeur globale (1) de a n'est pas connue localement et donc, il n'y a pas d'affichage pour la valeur de a.

En PHP, une variable globale doit être déclarée à l'intérieur de chaque fonction afin de pouvoir être utilisée dans cette fonction. Par exemple:

```
$a = 1;
$b = 2;
function somme() {
   global $a, $b;
   $b = $a + $b;
}
somme();
echo $b;
```
Le script ci-dessus va afficher la valeur 3. En déclarant globales les variables $a et $b locales de la fonction somme(), toutes les références à ces variables concerneront les variables globales. Il n'y a aucune limite au nombre de variables globales qui peuvent être manipulées par une fonction.

Une deuxième méthode pour accéder aux variables globales est d'utiliser le tableau associatif prédéfini $GLOBALS. Le précédent exemple peut être réécrit de la manière suivante:

```
 $a = 1;
 $b = 2;
 function somme() {
   $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
 }
 somme();
 echo $b;
```

## I. Classes et Objets

Php permet la poo (programmation orientée objet) comme java C++ javasctipt ou Delphi Il est possible de définir des classes et d'instancier les objets correspondants

Cette partie suppose une connaissance des langages orientés objet. On ne donnera que la syntaxe sur l'exemple très classique des carrés et rectangles

- Le constructeur porte le même nom que la classe (comme en java)
- Le mécanisme d'héritage s'obtient par ***extends***

- ***This*** fait référence, dans une classe, à l'objet correspondant à l'instance de cette classe

```
<?php

class rectangle {
 var $larg,$Long;
  function rectangle($Long,$larg){
    $this->Long = $Long;
    $this->larg = $larg;
  }
  function surface (){
    return $this->Long * $this->larg ;
  }
  function double(){
    $this->Long *= 2;
    $this->larg *= 2;
  }
  function perimetre(){
    return 2*($this->Long + $this->larg) ;
  }
}


class carre extends rectangle {
  function carre($cote){
    $this->Long = $cote;
    $this->larg = $cote;
  }
}

$rect = new rectangle(5,12);

print $rect->surface();
echo '<br/>';
$rect->double();
print $rect->surface();
echo '<br/>';
print $rect->perimetre();


echo '<br/>';
$K = new carre(5);

print $K->surface();
echo '<br/>';
$K->double();
print $K->surface();
echo '<br/>';
print $K->perimetre();

?>
```

## J. Entrées/Sorties

### 1. sorties

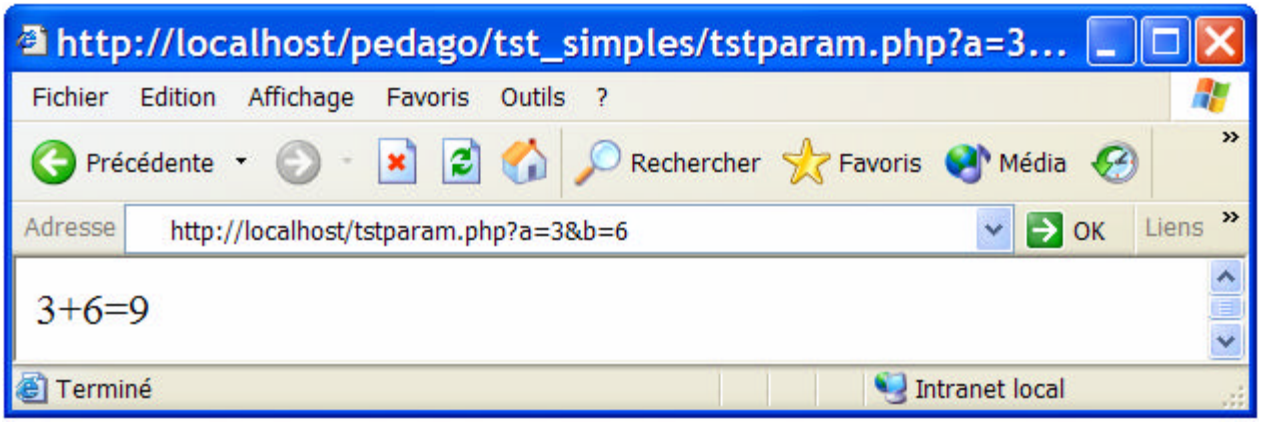
Pour obtenir du serveur l'envoi vers le client (navigateur) de caractères constituant du code html, on utilise indiferemment ***print()*** ou ***echo()*** avec ou sans parenthèses.

### 2. entrées

Pour que le serveur obtienne des informations du client, on peut envoyer par l'***url*** des paramètres avec leur valeur (méthode ***get***) de la manière suivante :

```
<?php
 echo "$a+$b=".($a+$b)
?>
```

Le code précédent, contenu dans le fichier tstparam.php et appelé de cette façon :




Le code ci-dessus (ou similaire) peut être exécuté :

- Soit directement
- Soit avec la balise ***href***
- soit par le biais des formulaires soit avec ***get*** soit avec ***post***
  la méthode ***get*** transmet les informations dans l'URL avec la syntaxe ci-dessus(la taille des info est davantage limitée qu'avec ***post*** )
  la méthode ***post*** les transmet sans qu'elles soient visible dans l'URL On préférera donc en général la méthode ***post*** pour des questions de confidentialité, la méthode ***get*** est bien utile pour vérifier le bon passage des informations durant la phase de développement.

Le passage de paramètres s'effectue de la manière suivante :

- Le **?** entre l'URL et le premier paramètre
- le **&** séparant chaque paramètre
- le **=** entre le nom de variable et sa valeur

## K. Fichiers

La lecture et l'écriture dans un fichier local ou distant se fait de la manière suivante :

Pour l'écriture :

```
$filename = 'ess.txt';
$somecontent = "Ajout de cha&icirc;ne dans le fichier \n";

// Assurons nous que le fichier est accessible en &eacute;criture
if (is_writable($filename)) {

   // Dans notre exemple, nous ouvrons le fichier $filename en mode d'ajout
   // Le pointeur de fichier est plac&eacute; &agrave; la fin du fichier
   // c'est l&agrave; que $somecontent sera plac&eacute;
   if (!$handle = fopen($filename, 'a')) {
        print "Impossible d'ouvrir le fichier ($filename)";
        exit;
   }

   // Write $somecontent to our opened file.
   if (!fwrite($handle, $somecontent)) {
     print "Impossible d'&eacute;crire dans le fichier ($filename)";
     exit;
   }

   print "L'&eacute;criture de ($somecontent) dans le fichier ($filename) a r&eacute;ussi";

   fclose($handle);

} else {
   print "Le fichier $filename n'est pas accessible en &eacute;criture.";
}
```

Et pour la lecture

```
// Lit un fichier, et le place dans une cha&icirc;ne
$filename = "./ess.txt";
$fic = fopen ($filename, "r");
$contenu = fread ($fic, filesize ($filename));
print $contenu;
fclose ($fic);
```

```
?>
```

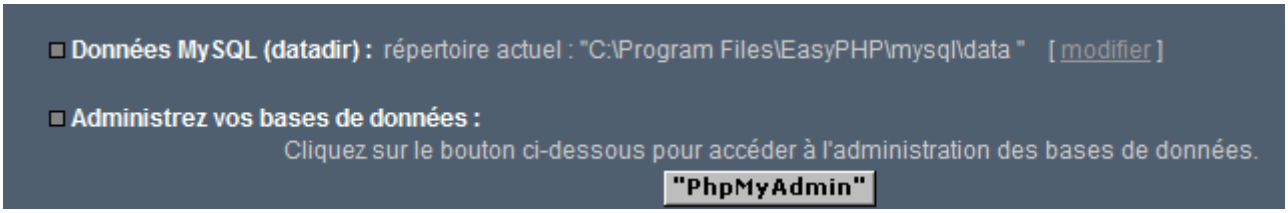
## L. Inclusion de fichiers

Pour inclure un fichier contenant du php dans du code php, on utilise :

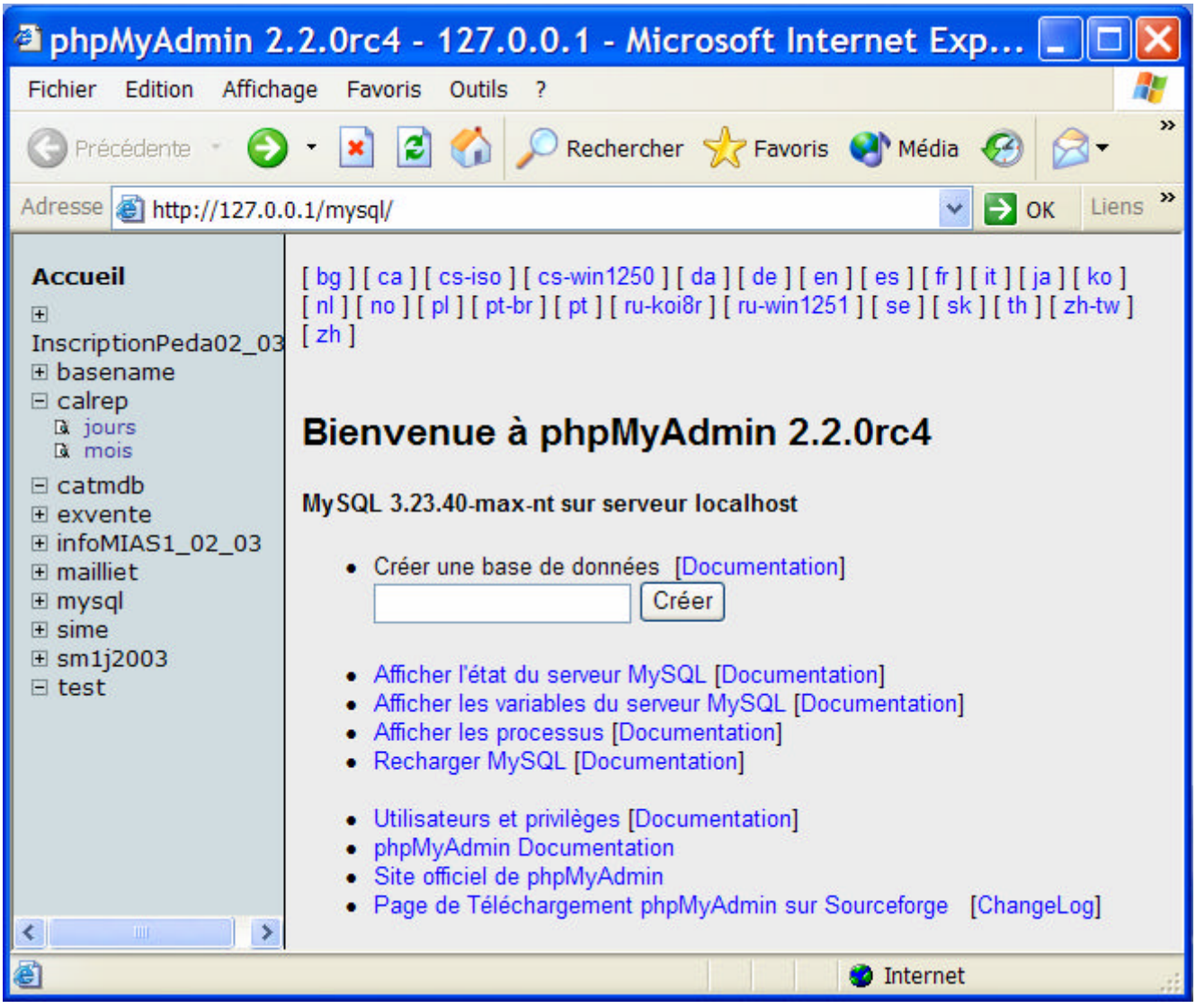
include(), include_once(), require() et require_once().

# VIII. PhpMyAdmin

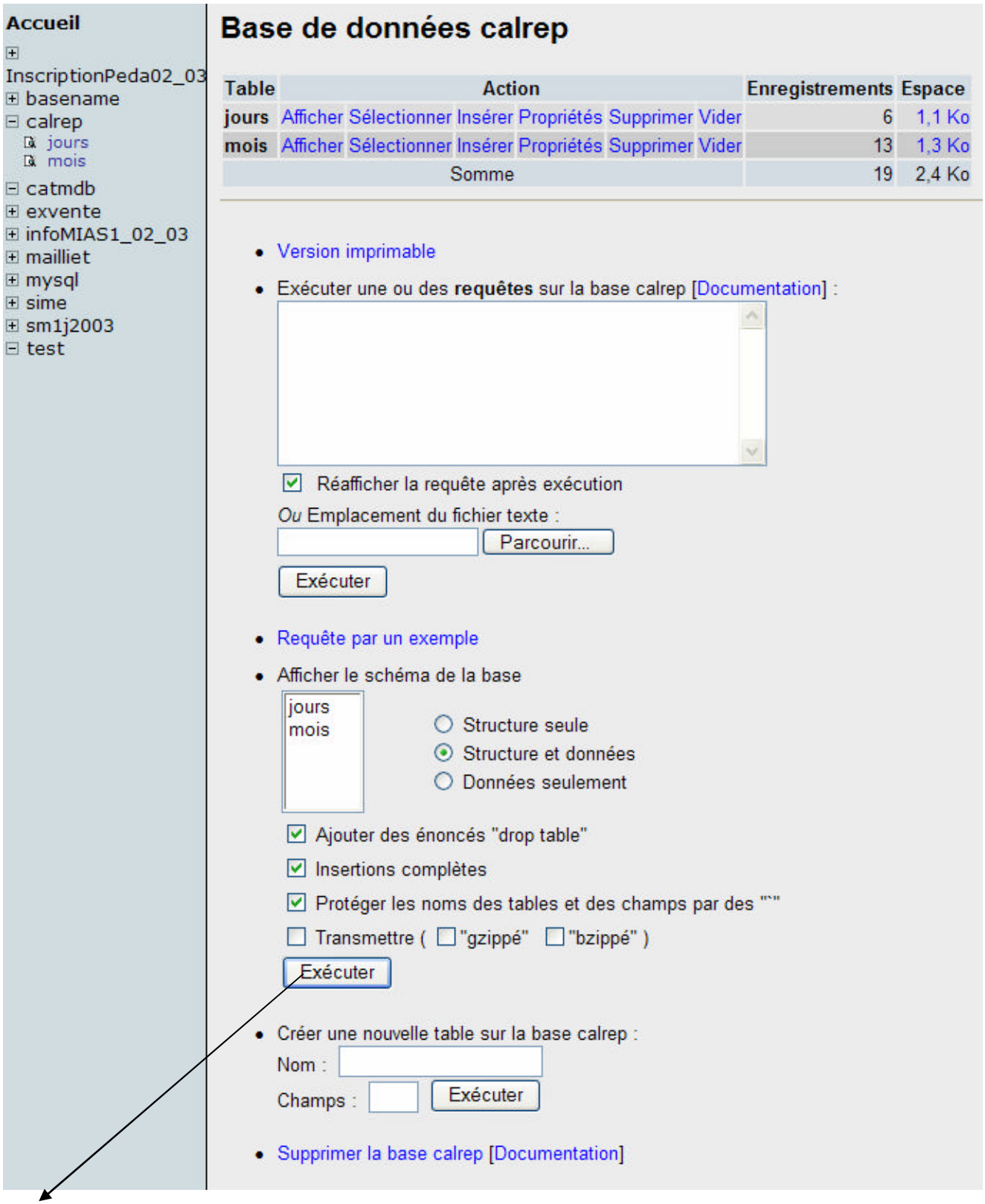
On y accède dans le menu principal de easyphp sous windows (ou après avoir téléchargé et installé PhpMyAdmin sous linux) On voit ci-dessous le répertoire de stockage des bases et des tables.




Le bouton ci-dessus donne accès à :




On voit ci-dessus les différentes bases que l'on peut développer en cliquant sur le + et découvrir les tables qui la compose




donnera :

```
# phpMyAdmin MySQL-Dump
# version 2.2.0rc4
# http://phpwizard.net/phpMyAdmin/
# http://phpmyadmin.sourceforge.net/ (download page)
#
```

```
# Serveur: localhost
# Généré le : August 6, 2003, 6:52 pm
# Version du serveur: 3.23.40
# Version de PHP: 4.0.6
# Base de données: `calrep`
# --------------------------------------------------------

#
# Structure de la table `jours`
#

DROP TABLE IF EXISTS `jours`;
CREATE TABLE `jours` (
  `num_jour` int(11) default NULL,
  `nom_jour` varchar(20) default NULL
) TYPE=MyISAM;

#
# Contenu de la table `jours`
#

INSERT INTO `jours` (`num_jour`, `nom_jour`) VALUES (1,'de la vertue');
INSERT INTO `jours` (`num_jour`, `nom_jour`) VALUES (2,'du genie');
INSERT INTO `jours` (`num_jour`, `nom_jour`) VALUES (3,'du travail');

INSERT INTO `jours` (`num_jour`, `nom_jour`) VALUES (4,'de l&acute;opinion');
INSERT INTO `jours` (`num_jour`, `nom_jour`) VALUES (5,'des recompenses');
INSERT INTO `jours` (`num_jour`, `nom_jour`) VALUES (6,'de la révolution');
# --------------------------------------------------------

#
# Structure de la table `mois`
#

DROP TABLE IF EXISTS `mois`;
CREATE TABLE `mois` (
  `num_mois` int(11) default NULL,
  `nom_mois` varchar(15) default NULL
) TYPE=MyISAM;

#
# Contenu de la table `mois`
#

INSERT INTO `mois` (`num_mois`, `nom_mois`) VALUES (1,'vendemiaire');
INSERT INTO `mois` (`num_mois`, `nom_mois`) VALUES (2,'brumaire');
INSERT INTO `mois` (`num_mois`, `nom_mois`) VALUES (3,'frimaire');
INSERT INTO `mois` (`num_mois`, `nom_mois`) VALUES (4,'nivose');
INSERT INTO `mois` (`num_mois`, `nom_mois`) VALUES (5,'pluviose');
INSERT INTO `mois` (`num_mois`, `nom_mois`) VALUES (6,'ventose');
INSERT INTO `mois` (`num_mois`, `nom_mois`) VALUES (7,'germinal');
INSERT INTO `mois` (`num_mois`, `nom_mois`) VALUES (8,'floreal');
INSERT INTO `mois` (`num_mois`, `nom_mois`) VALUES (9,'prairial');
INSERT INTO `mois` (`num_mois`, `nom_mois`) VALUES (10,'messidor');
INSERT INTO `mois` (`num_mois`, `nom_mois`) VALUES (11,'thermidor');
INSERT INTO `mois` (`num_mois`, `nom_mois`) VALUES (12,'fructidor');
```

```
INSERT INTO `mois` (`num_mois`, `nom_mois`) VALUES (13,'sansculottides');
```

On peut sauvegarder les lignes dans un fichier ***ess.sql*** (par exemple) et ajouter au début du fichier ( par exemple après

```
# --------------------------------------------------------
```

) les lignes :

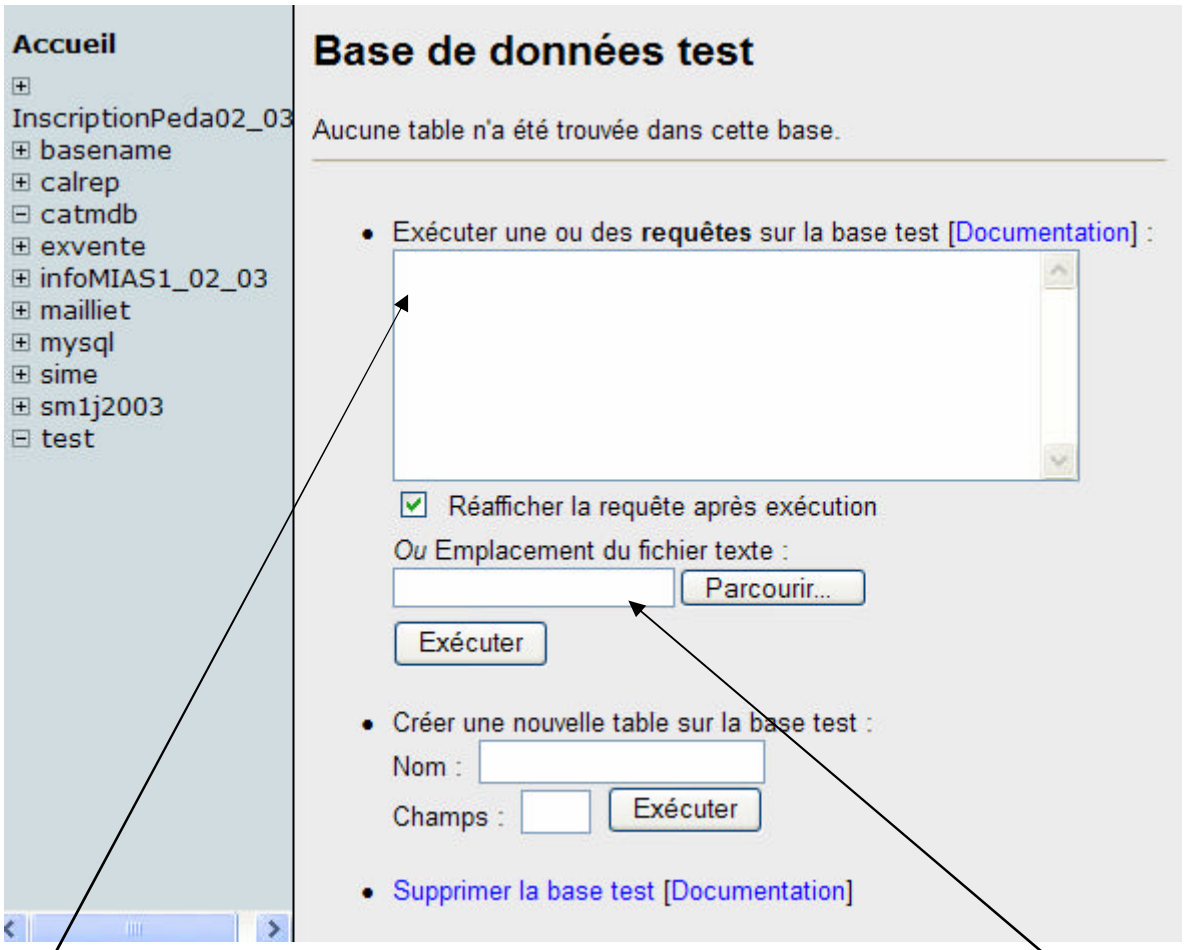
```
# -- création de la base ---

CREATE DATABASE `calrep`;

USE calrep;
```

Remarque : # débute un commentaire

Cette sauvegarde permet par exemple de recréer la base, les tables et les données très facilement :




Soit par un copier/coller des lignes de ***ess.sql***

Soit en indiquant le chemin d'accès à ***ess.sql***

Remarque : pour créer Base/Tables/Données, on peut soit utiliser l'interface proposée par ***phpMyAdmin*** puis sauvegarder (c'est plus prudent) dans un fichier. Soit créer directement le fichier grâce à un éditeur de texte et utiliser le procédure ci-dessus. Les documentations ne manquent pas (voir les copies d'écran ci-dessus) Ne pas hésiter à les consulter

# IX. Interfaçage PhP/MySql

Il est intéressant de créer un fichier ***connect.php*** :

```
<?php
define(NOM,"xxx");
define(PASSE,"xxx");
define(SERVEUR," xxx");
define(BASE,"xxx ");

function Erreur($message){
  print("<DIV style=\"color:red\">");
  print($message);
  print("</DIV>");
  exit;
}
function FermerBase(){
   if (mysql_close()==false)
      Erreur("Fermeture impossible de la base");
}

function ouvrirBase(){
$connexion=mysql_connect(SERVEUR,NOM,PASSE);
if ($connexion ==0)
   Erreur("impossible d'etablir la connexion");
if(mysql_select_db(BASE, $connexion) == 0)
   Erreur("impossible de se connecter a la base "+BASE);
}
?>
```

Remplacer les xxx par les valeurs correspondantes :

- Le nom de l'utilisateur autorisé à accéder à la base
- Son mot de passe
- Adresse IP ou nom du serveur MySql (par exemple : localhost)
- Nom de la base


Il est important de donner à ce fichier l'extension ***.php*** alors qu'une autre extension conviendrait mais n'offrirait pas de sécurité puisqu'un accès directe par l'url permettrait de lire son contenu puisque l'interprète php ne l'interpréterait pas (à cause de l'extension qui ne serait pas reconnu)

Il suffit ensuite de mettre au début du fichier : ***require("connect.php");*** pour l'inclure dans les programme qui doivent effectuer des requêtes mysql

```
<?php
 require("connect.php");
 ouvrirBase();
 $requete="select * from xxx";
 $resultat=mysql_query($requete);
 if($resultat == 0)
    Erreur("impossible d'effectuer la requete <BR>$requete");

 if (mysql_num_rows($resultat)==0)
    Erreur("d&eacute;sol&eacute; nous n'avons aucune information");
 // récupèration du nombre de champs
 $nbChamps = mysql_num_fields($resultat) ;
```

```
 // récupération du nom des champs
 for ($i=0; $i<$nbChamps; $i++) {
   echo (mysql_field_name($resultat,$i));
   echo' , ';
 }
 echo '<br/>';
 echo '<br/>';
 // récupération des enregistrements
 while ($data=mysql_fetch_row($resultat)){
   for ($i=0; $i<$nbChamps; $i++) {
     echo $data[$i] ;
     echo' , ';
   }
   echo '<br/>';
 }
 FermerBase();
?>
```

remplacer les xxx de la requête Sql par le nom de la table

On peut aussi obtenir la liste des bases et des tables d'une base par des requêtes adaptées :

```
 $requete="show tables";
```

et

```
 $requete="show databases";
```

Le programme ci-dessus donne la liste des champs et des enregistrements presque sans aucune mise en forme, **echo' , ';** sépare les éléments de chaque enregistrement et **echo '<br/>';** insère un passage à la ligne entre les différents enregistrements



# Cours Internet



D. Mailliet