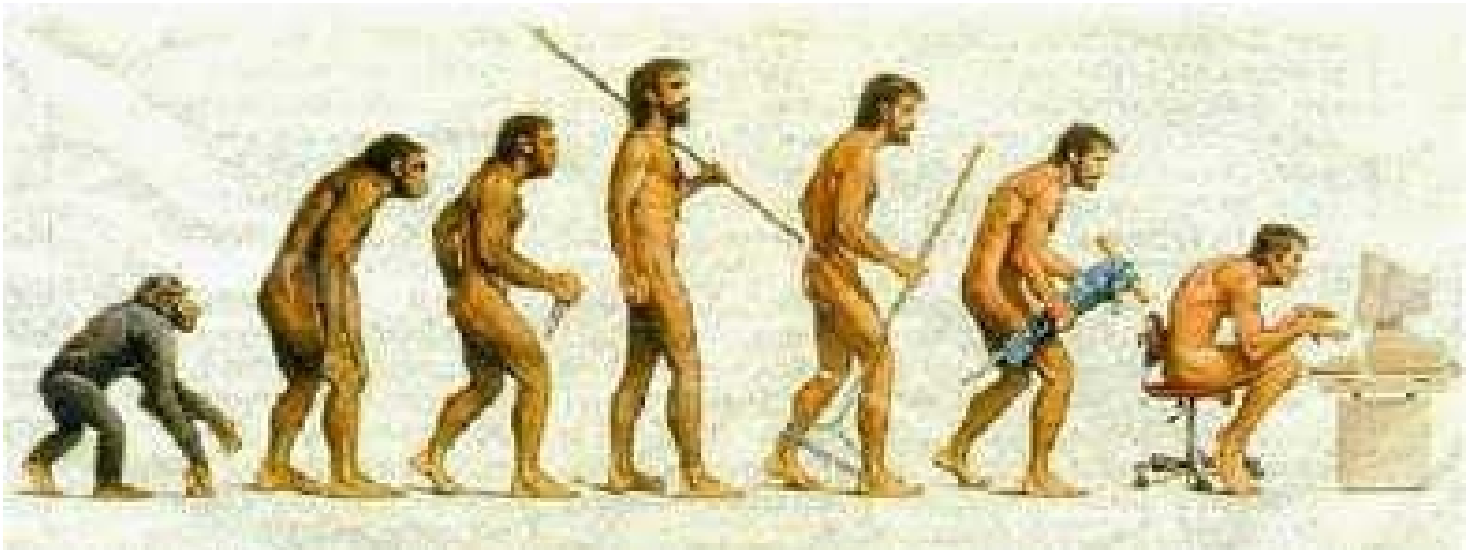
LES TECHNOLOGIES LIEES A L'INTERNET

UN EVOLUTION ?



2006-2007
Antony Chien Chow Chine

SOMMAIRE

~~~~~~~~~~~~~~

~~~~~~~~~~~~~~

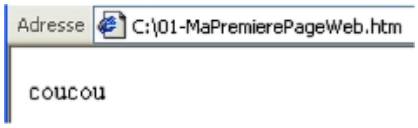
INTERNET

Voici un mot qui sous-entend aujourd'hui (d'un point de vue informatique) de nombreuses technologies. Nous essaierons ici de faire une ouverture sur un maximum d'entres elles. Il vous appartiendra de les développer plus avant.

I/ UNE PAGE WEB

Une page WEB se définie comme étant une page pouvant être affichée dans un navigateur (comme Internet, Netspace, Mozilla, Opera, …)

Le premier langage utilisé pour la conception d'une page Web est le Hyper Text Markup Language, le HTML. Il ne s'agit pas d'un langage de programmation au sens propre, mais d'un simple langage de description.
Le fichier qui contient la description de cette page porte en général l'extension `.htm` (ou `.html`). Il est constitué du texte et des liens aux images à afficher, répartis entre des balises, qui déterminent la façon dont ces éléments seront présentés dans le navigateur.



Ma première page Web

Cette page se définie par le code suivant : `premierepage.htm`

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

I.1- L'attitude

Firefox, Netscape et Microsoft prennent à eux 5 à 90 % du marché des browsers (navigateurs). Le reste, soit 10 à 15 %, est occupé par une trentaine de navigateurs plus ou moins connus.

Et bien que les balises soient plus ou moins compatibles, votre page Web ne sera pas à 100% parfaite sur chacun de ces navigateurs, qui, rappelons-le, est le choix de l'utilisateur et non le votre. Des défauts plus ou moins important de présentation et/ou d'affichage apparaîtront. Alors comment faire ?

I.1.a) Attitude 1: Je veux écrire pour tous les browsers

Autant vous dire que c'est mission impossible! Avec tous les navigateurs sur le marché et parfois des déclinaisons selon les systèmes d'exploitation, avec les différentes options de l'utilisateur et les différentes résolutions d'écran, il n'est pas du domaine du possible de pouvoir tester et adapter le codage des balises à toutes les situations rencontrées sur le Web.

100% - impossible -

« Il faut bien accepter qu'un certain pourcentage de la clientèle ne puisse être prospectée car les coûts et les moyens à mettre en oeuvre sont beaucoup trop importants pour le peu de clients potentiels ».

I.1.b) Attitude 2 : Je teste mon site sous un seul navigateur

On passe d'un extrême à l'autre. Un site testé sous un seul navigateur ne sera parfait que pour (un maximum de) 30% de vos visiteurs. Pour les autres 70%, ce sera "au petit bonheur la chance".

Pour des pages personnelles, vous pouvez peut-être vous en contenter mais si votre site représente une société commerciale, un organisme, une association, les retombées en terme d'image pour ces 70% visiteurs insatisfaits, ne seront guère favorables [euphémisme].

max 30%

Accepteriez-vous que votre clientèle soit à 70% mal ou incomplètement démarchée?

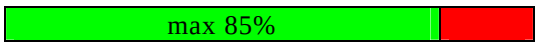
I.1.c) Attitude 3 : Site high-tech, écrit pour les browsers de dernière génération

Ces sites ne s'adressent qu'à un maximum de 55 % des visiteurs du Web. Car tous les internautes « n'évoluent » pas forcement aussi rapidement que les technologies (fautes de technicités, de moyens, …).

« Avez-vous démarché les bons internautes ? »

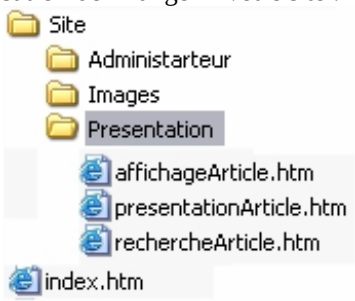
I.1.d) Attitude 4 : Ecrire pour le maximum « raisonnable » d'internautes

Vous l'avez compris, la compatibilité d'un site ne peut être qu'un compromis.



I.2- Organisation d'un site

Les sites sont constitués d'un ensemble de pages reliées entre elles (ce qui permet la navigation). Il vous appartient dans un souci de maintenance et d'organisation de « ranger » votre site :



Conseils :

Compatibilité des navigateurs

- Plus votre Html sera simple, plus votre site sera compatible.

Compatibilité des « operating system »

- Si le langage Html est compatible avec toutes les plates-formes, les noms des fichiers de votre site doivent respecter les conventions des "`operating system`" :
  - éviter les : $ % ' ` @ ^ ! & { } ( ) " ~ \ / : * ? < > | …, ainsi que les redondances dans les noms.
  - **Ecrivez en minuscule :** **`affichagearticle.htm`**
- Placer l'ensemble de vos pages par thèmes dans des répertoires et placer une première page nommée `index.htm` à la racine.

Compatibilité des graphiques

- Préférez toujours le codage hexadécimal des couleurs plutôt que le nom de couleurs.
- Utilisez le plus possible des images de petite résolution (jouer sur la taille, le nombre de couleurs, …). Cela diminue le temps de chargement et le risque de problèmes d'affichage.
- Testez votre site sous diverses résolutions d'écran 640x480, 800x600 et 1024x768. Et trouvez un compromis.

I.3- Méthode de construction

Développer et afficher une page Web n'est pas l'unique « raison d'être » d'un site. Son but est d'être visité et utilisé par un maximum d'internaute.

Ainsi, il est recommandé que le site s'inscrive dans une démarche, dans un projet. Ceci afin de guider la réalisation sans oublier d'étapes, de pouvoir travailler en équipe, de maintenir aisément des pages sans modifier l'intégralité du site, d'utiliser des règles et des conventions qui permettront un développement et un suivi facile.

Exemple de méthode :

- ✓ **Connaître** : Faire une analyse externe et interne du projet.
  - **Analyse externe** : étudier d'autres sites, afin d'avoir une idée de ce que l'on souhaite obtenir, dans son aspect graphique ou dans son fonctionnement (benchmarking).

- **Analyse interne** : dégager l'utilité du site pour l'entreprise, les avantages et les inconvénients. Le développement d'un site peut entraîner la participation d'un certain nombre de personnes, cela demande de l'organisation (volonté et capacité des personnels à utiliser et/ou à aider).

✓ **Cibler** : À qui doit s'adresser le site ? Ceci permettra de préciser le contenu, la charte graphique, le style ou la forme.

✓ **Choisir** : Etude des besoins au niveau des performances (choix logiciels et matériels), des délais (temps de création du site), des coûts (hébergement, promotion, moyen humains et matériels, …), des actions à mener, des attentes sur le contenu, sur les services (forum, téléchargement,…), le design (apparence physique), l'ergonomie (prise en main), la navigation (architecture du site), l'animation.

✓ **Construire** : Nous distinguerons plusieurs types de site :

- **Le site vitrine** : beaucoup de présentation, peu d'informations « trop précises » (prix, noms, …), afin de faciliter la maintenance.
- **Le site catalogue** : site plus descriptif et plus précis, mais seul l'utilisateur obtient des réponses. L'utilisateur ne peut « communiquer » avec le site.
- **Le site marchand** : le plus complet. L'interaction se fait dans les deux sens. Le site fournit des renseignements à l'internaute. Mais l'internaute peut aussi disposer de services lui permettant de poser des questions (forum, news, …), laisser des informations le concernant (fiche client) ou répondre à des questionnaires (statistiques).

✓ **Communiquer** :

- **Se faire connaître** : nom de domaine, publicité externe (journaux, télé,…), feed-back (retour d'informations)
- **Référencement** : choix des mots-clés, popularité des pages (page-rank), optimisation du site (url-rewrinting), …

✓ **Changer** : la maintenance d'un site est obligatoire pour sa survie. Elle doit concerner autant le fond que la forme. Notez que certains hébergeurs testent la fréquence de modifications des pages.

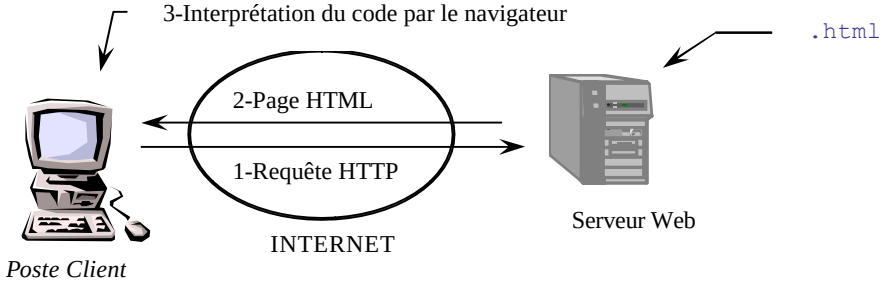
II/ L'ARCHITECTURE CLIENT-SERVEUR

Un site Internet est un ensemble de page, situé sur un serveur (serveur Web) et accessible à tous. Le serveur Web est généralement un fournisseur d'accès ou un hébergeur (Free, Voila, OVH, …).

Le protocole utilisé pour transférer des informations sur Internet s'appelle `http` (acronyme de `Hyper Text Transfert Protocol`). Une requête `http` :

http://	www.MonSite	/MonRepertoire	/index.htm (ou asp / php)
Protocole	Nom du serveur	Nom de l'alias	Page de démarrage

est envoyée vers le serveur afin d'accéder à la page désirée. Cette page est renvoyée et interprétée par le navigateur du poste client :



III/ INTERNET, INTRANET ET EXTRANET

L'Internet représente l'interconnexion de tous les ordinateurs du monde. Ce grand réseau ressemble à une gigantesque toile d'araignée d'où le World Wide Web.

Au fur et à mesure de son évolution, l'Internet ne s'est plus contenté d'offrir de l'information, mais a proposé de plus en plus de services aux internautes (`forum`, `ftp`, `https`, …). Bref, un immense réseau fédérateur, puisque chaque personne, quelque soit son ordinateur, sa plate-forme ou son environnement, peut avoir accès et partager tout type d'information.

Ces technologies ont séduit les entreprises, qui ont souhaité ramener cette technologie à l'échelle de leurs entreprises et avoir ainsi leur propre réseau Internet, où chaque employé peut avoir accès et échanger des données de l'entreprise. On parle d'Intranet.

Enfin, les entreprises ont souhaité élargir encore un peu plus leur intranet, en y intégrant leurs partenaires (clients privilégiés, fournisseurs, sous-traitants, …). Mais l'utilisation de l'Internet ne leur paraissant pas assez sécurisé, les entreprises ont choisi d'intégrer dans leurs réseaux (par des moyens matériels et logiciels) les réseaux de leurs partenaires, formant ainsi un Extranet : réseau privé sortant des limites de l'entreprise.

LE LANGAGE HTML

I/ LE WEB PARLE HTML

HTML est le langage universel utilisé pour communiquer sur le Web.

Pour transiter le plus rapidement possible sur les lignes téléphoniques, on a adopté un format de `texte` très compact mais aussi (par conséquence) peu sophistiqué. C'est le bon vieux format de texte pur et dur, sans fioritures du `Bloc-notes`.

En plus du texte adressé à votre lecteur, il vous faudra inclure des instructions pour le browser de celui-ci. Ces instructions seront différenciées de votre texte par les signes `<` et `>` par exemple `<html>`. Ces "instructions" s'appellent des tags ou des balises.

Quand vous écrirez les balises de votre page HTML, il faudra garder à l'esprit :
- qu'une balise marque une action pour le browser (ce qu'il doit faire...).
- que les attributs précisent les modalités de cette action (comment il doit le faire...).

II/ VOS PREMIERS OUTILS

II.1- Vous avez besoin

- ✓ d'un éditeur de texte comme le `Bloc-notes`, mais de plus sophstiqués apparaissent tel que : `Amaya`, `Bluefish` ou encore `NVU`.
- ✓ d'un browser : Firefox de Mozilla Corporation (http://www.firefox.eu.com/fr et ajoutez des `extensions`), Internet Explorer de Microsoft (http://www.microsoft.com, n'oubliez pas la `mise à jour`) ou tout autre équivalent dans votre système d'exploitation.

II.2- Vous n'avez pas besoin

- ✓ **d'être connecté à Internet** pour écrire, voir et peaufiner vos pages `Html`.
- ✓ d'avoir le dernier éditeur `Html`.

II.3- Conseils

Le langage `Html` étant un ensemble de balises et d'attributs, il nous paraît utile sinon indispensable de les passer en revue car :

- si les éditeurs `Html` vous faciliteront grandement la tâche, ils ne sont pas parfaits. Il faudra (régulièrement) vous plonger dans le code source pour corriger les dysfonctionnements.
- les codes source de vous pages préférées sont disponibles (et sans copyright). Il est alors possible de s'en inspirer pour reprendre le procédé sans avoir à réinventer la roue.
- vous aurez besoin d'une connaissance pointue du `Html` pour inclure les technologies suivantes.

III/ LE DOCUMENT HTML

III.1- Vos premières balises

`<html>`	Début d'un document de type HTML.
`</html>`	Fin d'un document de type HTML.
`<head>`	Ceci est le début de la zone d'en-tête (prologue au document contenant des informations destinées au browser).
`</head>`	Fin de la zone d'en-tête.
`<title>`	Début du titre de la page.
`</title>`	Fin du titre de la page.
`<body>`	Début du document.
`</body>`	Fin du document.

- Vous aurez remarqué qu'à chaque balise de début d'une action, soit `<...>`, correspond une balise de fin d'une action `</...>`.
- Bien que les balises ne soient pas (à ce stade) « case sensitive », n'oubliez pas de tout écrire en minuscule !

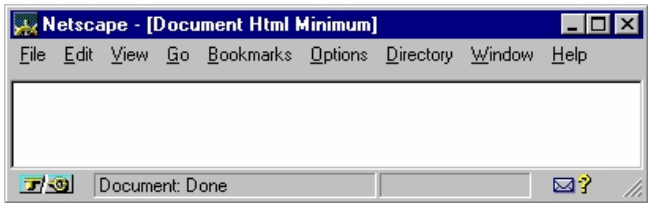
III.2- Notre premier document Html

1- Ouvrir l'éditeur de texte.
2- Ecrire le code Html suivant :

```
<html>
  <head>
    <title>Document Html minimum</title>
  </head>
  <body>
  </body>
</html>
```

3- Enregistrer le document avec l'extension `.htm` (ou `.html`).

4- Ouvrir le navigateur.
5- Afficher le document via le menu `File/Open file...`
6- Admirer votre premier document Html.



Celui-ci est vide (et c'est normal), mais tout à fait opérationnel. Il faudra maintenant lui fournir votre information à l'intérieur des balises `<body></body>`. Remarquez que votre `"title"` est présent dans la fenêtre.

Pour vos éventuelles modifications, il n'est pas nécessaire de rouvrir à chaque fois le navigateur.

- Retourner dans l'éditeur de texte (sans fermer le navigateur).
- Modifier le code Html.
- **Enregistrer le fichier.**
- Utiliser la commande `"Reload"` du browser ou cliquer dans la barre `"Location"` et faire `"Enter"`.

IV/ MISE EN FORME HTML

Nous verrons dans les chapitres suivants que la MISE EN FORME DOIT ETRE GEREE PAR DES FEUILLES DE STYLE.

Les deux points qui suivent sont donnés à titre d'information.

IV.1- Le texte

Quelques balises permettant d'agrémenter du texte :

Gras	[Bold]	`<b>...</b>` `<strong>...</strong>`	Début et fin de zone en gras
Italique	[Italic]	`<i>...</i>` `<em>...</em>`	Début et fin de zone en italique
Taille de caractère	[Font size]	`<font size=?>...</font>`	Début et fin de zone avec cette taille
Couleur de caractère	[Font color]	`<font color="#$$$$$$"></font>`	Début et fin de zone en couleur
A la ligne	[Line break]	` `	Aller à la ligne
Commentaires	[Comments]	`<!-- *** -->`	Ne pas afficher
Centrage	[Center]	`<center></center>`	Centrer

Quelques commentaires s'imposent :

- La taille dans `<font size=?>` peut être indiquée de deux façons :
  1. avec un nombre de 1 à 7. La valeur par défaut étant 3.
  2. de façon relative par rapport à la valeur par défaut (ici 0). Soit -3 -2 -1 0 +1 +2 +3.

- Quelques codes couleurs :

Bleu	#0000FF	Vert	#00FF00
Blanc	#FFFFFF	Violet	#8000FF
Rouge	#FF0000	Jaune	#FFFF00
Gris clair	#C0C0C0	Noir	#000000

Et les noms de code :

aqua	bleu clair	white	blanc
black	noir	yellow	jaune
blue	bleu	red	rouge
fucshia	lilas	lime	vert clair
gray	gris	green	vert
maroon	marron	navy	bleu marine
olive	kaki	silver	argenté
purple	pourpre	teal	cyan foncé

IV.2- Les frames

Les frames permettant une structuration plus fonctionnelle d'un site. Mais ne répondant pas aux normes d'accessibilités, celles-ci sont aujourd'hui abandonnées au profit du positionnement par feuille de style.

Pour diviser l'écran en plusieurs fenêtres, les balises sont peu nombreuses :

Zone avec des fenêtres

```
<framset>
</framset>
```

Début de zone avec des fenêtres
Fin de zone avec des fenêtres

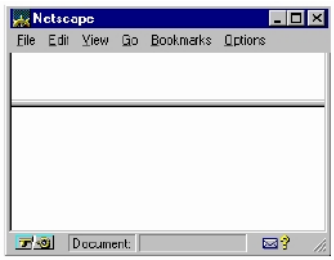
Agencement des fenêtres

```
<framset rows="...">
<framset cols="...">
```

Fenêtres horizontales
Fenêtres verticales

Exemple : Fenêtre avec bandeau

```
<html>
<head></head>
<framset rows="30%,70%">
  <frame>
  <frame>
</framset>
</html>
```



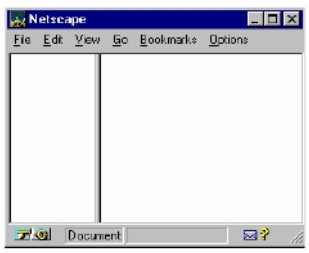
Attention ! `<framset></framset>` remplace `<body></body>`

IV.2.a) L'attribut Rows

L'attribut `rows="hauteur1, hauteur2, ..., hauteurN"` définit la hauteur des différentes fenêtres en cas de division horizontale. La hauteur s'exprime en `pixels` ou en `%`. Dans ce cas, on veillera à ce que le total soit égal à 100%.

Exemple : Fenêtre avec sommaire

```
<framset cols="30%,70%">
  <frame>
  <frame>
</framset>
```

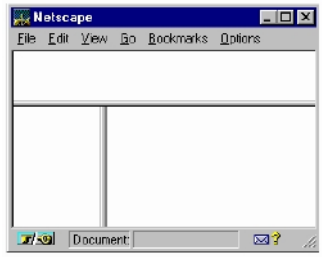


IV.2.b) L'attribut Cols

L'attribut `cols="largeur1, largeur2, ..., largeurN"` définit la largeur des différentes fenêtres en cas de division verticale. La largeur s'exprime en `pixels` ou en `%`. Dans ce cas, on veillera à ce que le total soit égal à 100%.

Exemple : Fenêtre type

```
<framset rowsS="30%,70%">
  <frame>
  <framset cols="30%,70%">
    <frame>
    <frame>
  </framset>
</framset>
```



IV.2.c) Contenu des frames

Chacun des cadres doit faire référence à une page `.htm` :

```
<frame src="URL ou adresse du document">
```

Exemple : 3 fichiers Html élémentaires que l'on place dans le même répertoire que le fichier de frames :

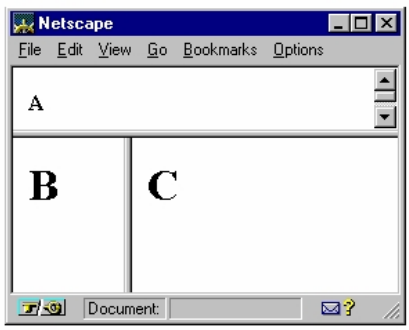
A.htm
```
<html><body>
  <h4>A</h4>
</body></html>
```

B.htm
```
<html><body>
  <h1>B</h1>
</body></html>
```

C.htm
```
<html><body>
  <h1>C</h1>
</body></html>
```

On reprend le fichier de frame précédent que l'on complète :

```
<framset rows="30%,70%">
  <frame src="A.htm">
  <framset cols="30%,70%">
    <frame src="B.htm">
    <frame src="C.htm">
  </framset>
</framset>
```



Les ascenseurs, comme à la fenêtre A, apparaissent automatiquement. Par l'attribut de la balise `<frame scrolling="yes/no/auto">` vous pouvez indiquer si la fenêtre doit ou non posséder une barre de défilement.

IV.2.d) Une frame cible

La balise `<frame name="NOM">` indique le nom de la fenêtre, de telle sorte que cette frame puisse être utilisée comme cible d'un lien hypertexte.

Exemple : je voudrais faire un lien sur `B` pour afficher le contenu de ce lien (prenons le fichier `A.htm`) dans `C`.

```
<framset rows="30%,70%">
  <frame src="A.htm">
  <framset cols="30%,70%">
    <frame src="B.htm">
    <frame src="C.htm" name="fenetreC">
  </framset>
</framset>
```

Et on met un lien vers `A.htm` dans le fichier `B.htm` en désignant comme cible la `frame C.`

```
<html><body>
  <a href="A.htm" target="fenetreC"><h1>B</h1></a>
</body></html>
```

L'attribut `TARGET` peut aussi prendre certaines valeurs prédéfinies :

- `_blank` : indique au browser qu'il doit créer une nouvelle fenêtre, afin d'y afficher le fichier.
- `_self :` indique que le fichier sera chargé dans la même fenêtre que celle dans laquelle se trouve le lien.
- `_top` : implique l'affichage du fichier sur toute la surface de la fenêtre du browser.

Commentaires :

- Il est possible de supprimer les bordures qui séparent les frames. (attribut `"border=0"` sous Netscape et `"frameborder=no"` et `"framespacing=0"` sous IE)
- La balise `<noframes>...</noframes>` est utilisée pour indiquer le texte que doivent afficher les browsers incapables de gérer les frames.

V/ LES TITRES ET LES LISTES

Tout document d'une certaine consistance se doit de présenter, les différents niveaux de son exposé.

En-têtes	[Heading]	`<hn> </hn> avec n=1 à 6`
Liste non ordonnée	[Bullet list]	`<ul> </ul>`
Liste ordonnée	[Numbered list]	`<ol> </ol>`
Elément de liste	[List items]	`<li>`
Paragraphe	[Paragraph]	`<p> </p>`

VI/ LES LIENS

`Html` (`Hyper Text Markup Language`) est un langage hypertexte qui vous permet en cliquant sur un mot, généralement souligné (ou une image) de vous transporter :

- vers un autre endroit du document.
- vers un autre fichier Html situé sur votre ordinateur.
- vers un autre ordinateur situé sur le Web.

Ce sont ces liens qui vous permettent de surfer de page en page. La syntaxe de ces liens est :

```
<a href="URL ou adresse">mot faisant le lien</A>
```

VI.1- Lien externe

Tout ordinateur situé sur le réseau Internet possède une adresse ou une URL (`Universal Ressource Location`), elles sont du type :

```
HTTP://serveur/chemin.../fichier
FTP://serveur/chemin.../fichier
mailto:utilisateur@hôte
```

VI.2- Lien local

Si l'ensemble de vos pages est situé dans un même répertoire, il vous suffit d'indiquer le nom du fichier :

```
fichier.htm
```

VI.3- Lien mixte

Nous entendons par là un lien vers un fichier situé à un autre endroit de votre ordinateur (et donc non situé dans le répertoire de votre site). L'adresse prendra la forme :

`file:///lecteur:/répertoire/fichier.htm` (en adressage absolu).

ou

`../../fichier.htm` (en adressage relatif : `../` permet d'accéder au répertoire parent).

Attention : Une réflexion doit être apportée lors de l'utilisation du type de lien.

VI.4- Les ancres

Des liens peuvent aussi pointer vers un endroit précis du même document ou d'un autre fichier. C'est ce qu'on appelle les ancres, ancrages ou pointeurs [Anchor].

Point d'ancrage	`<a name="AncreLabas"> Là-bas </a>`
Lien vers une ancre dans la même page	`<a href="#AncreLabas"> Cliquer ici pour aller là-bas </a>`
Lien vers une ancre dans une autre page	`<a href="URL#AncreLabas"> Cliquer ici pour aller là-bas </a>`

VII/ LES IMAGES

Le code `Html` pour insérer une image est :

```
<img src="Adresse de l'image"> Afficher l'image
```

La balise image possède de nombreux attributs :

`alt="Mon Image"`	Texte alternatif
`width = x` `height = x`	Hauteur et largeur (en pixels)
`border = X`	Bordure (en pixels)

`align=top`
`align=middle`
`align=botton`
`align=left`
	Alignement

```
align=right
```

Commentaires :

- En Html, l'image ne fait pas partie de votre document. Le browser va la chercher à l'adresse indiquée.
- Une image peut servir de lien : `<a href="fichier.htm"><img src="image.jpg"></a>`

VIII/ LES SEPARATEURS

Les `browsers` intègrent un outil de mise en forme : la ligne horizontale.

```
<hr> Insérer une ligne horizontale
```

Valeurs de ces différents attributs :

`<hr size=?>`	Epaisseur en pixels
`<hr width=?>`	Largeur en pixels
`<hr width="%">`	Largeur en % de la fenêtre
`<hr align=left>`	Alignement à gauche
`<hr align=right>`	Alignement à droite
`<hr align=center>`	Alignement centré

Commentaires :

- On préférera spécifier la largeur en % de la fenêtre plutôt qu'en pixels.

IX/ LES ARRIERE-PLANS

L'arrière-plan [background] d'un document peut être coloré ou composé d'une image :

```
<body bgcolor="#$$$$$$">
```

Des balises sont prévues pour modifier les couleurs utilisées par défaut par le browser pour le texte et les liens :

Couleur de texte	`<body text ="#$$$$$$">`
Couleur de lien	`<body link ="#$$$$$$">`
Lien visité	`<body vlink ="#$$$$$$">`
Lien actif	`<body alink ="#$$$$$$">`

On peut aussi prévoir un fond en image :

```
<body backgroung="Adresse">
```

X/ LES TABLEAUX

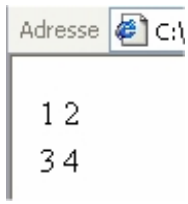
Les tableaux servent à placer des éléments à l'emplacement que vous souhaitez (comme dans les journaux vous « maquettez » vos pages).

Un tableau est composé de lignes et de colonnes qui forment les cellules du tableau :

Définition du tableau	[Table]	`<table> </table>`
Définition d'une ligne	[Table Row]	`<tr> </tr>`
Définition d'une cellule	[Table Data]	`<td> </td>`

Exemple 1 : Un tableau à deux lignes et deux colonnes (quatre cellules)

```
<table>
  <tr><td>1</td><td>2</td></tr>
  <tr><td>3</td><td>4</td></tr>
</table>
```



Exemple 2 : Adjoindre des bordures

Bordure de cadre [Border] `<table border=?></table>`

```
<table border=2>
  <tr><td>1</td><td>2</td></tr>
  <tr><td>3</td><td>4</td></tr>
</tale>
```

Il y a encore trois éléments (définis par défaut mais modifiables) :

L'espace entre les cellules ou l'épaisseur des lignes du quadrillage	`<table cellspacing=?>`
L'enrobage des cellules ou l'espace entre le bord et le contenu	`<table cellpadding=?>`
La largeur de la table	`<table width=?>` `<table width=%>`

X.1- Les cellules des tableaux

Les cellules peuvent contenir tous les éléments `Html` déjà passés en revue soit du `texte`, des `images`, des `liens`, des `arrière-plans` et même des `tableaux`.

Chaque cellule est un petit univers à part qui a ses propres spécifications :

Largeur d'une cellule	`<td width=?>` en pixels `<td width=%>` en pourcentage
Fusion de lignes	`<td rowspan=?>`
Fusion de colonnes	`<td colspan=?>`
Alignement Ligne	`<tr align=left/center/right>` `<tr valign=top/middle/bottom>`
Alignement Cellule	`<td align=left/center/right>` `<td valign=top/middle/bottom>`

Exemple : tableau centré qui occupe 60% de la fenêtre avec sur une ligne, trois colonnes égales.

```
<center><table width=60% border=1>
<tr>
  <td width=33%>cellule1</td>
  <td width=33%>cel. 2</td>
  <td width=34%>3</td>
</tr>
</table></center>
```

cellule 1	cel 2	3

Exemple : Tableau de 2 lignes, avec la première ligne qui prend toute la largeur. La première cellule doit donc déborder sur 3 cellules horizontales.

```
<center><table width=60% border=1>
<tr>
  <td colspan=3>cellule 1</td>
</tr>
<tr>
  <td width=33%>cellule 1</td>
  <td width=33%>cel 2</td>
  <td width=34%>3</td>
</tr>
</table></center>
```

cellule 1		
cellule 1	cel 2	3

Exemple : La première colonne prend toute la hauteur de la colonne. La première cellule doit donc déborder sur 2 cellules verticales.

```
<center><table width=60% border=1>
<tr>
  <td width=33% rowspan=2>cel 1</td>
  <td width=33%>cel 2</td>
  <td width=34%>3</td>
</tr>
<tr>
  <td width=33%>cel 2</td>
  <td width=34%>3</td>
</tr>
</table></center>
```

cellule 1	cel 2	3
	cel 2	3

Commentaires :

- `colspan` et `rowspan` peuvent être combinés.
- La liste des attributs n'est pas exhausitve puisqu'il existe aussi : `cellspacing` et `cellpadding` (espacement entre les cellules), `rules` (quadrillages avec attributs : `none`, `all`, `rows`, `cols`, `groups`)

XI/ LES IMAGES ANIMEES ET MAPPEES

XI.1- Les images animées

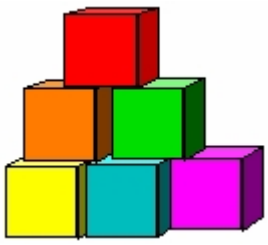
Les images animées qui agrémentent les pages Web sont des images `GIF`, composées un peu comme les dessins animés, par des logiciels conçus à cet effet : `Microsoft Gif Animator`, ou `Animation Shop` de Paint Shop Pro.

XI.2- Les images mappées

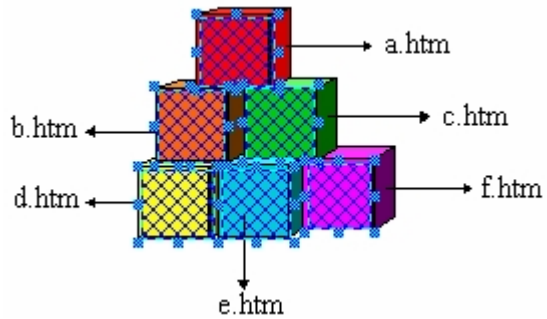
XI.2.a) Le concept

Les images mappées permettent d'effectuer des liens en fonction de zones [`area`] prédéfinies de l'image. Cette particularité peut se révéler fort utile pour définir, par exemple, des outils de navigation dans votre site.

On prend une image :



On va définir des zones dans l'image, un peu comme avec une carte de géographie [map en anglais] et associer à chacune de ces zones, un fichier :



On obtient ainsi une image dite "mappée".

XI.2.b) Les balises

Il existe plusieurs méthodes, la plus couramment utilisée est la méthode : `CSIM [Client Side Image Maps]` qui fait partie du langage `Html` (`Html 3.0`).

La balise de l'image mappée

```
<img src="image.gif" usemap="#nom_du_map"> pour une map dans le même fichier
```

ou

```
<img src="image.gif" usemap="fichier.html#nom_du_map"> pour une map située dans un autre fichier.
```

En fait, on ajoute simplement à la balise classique de l'image, l'attribut `USEMAP` pour avertir le navigateur qu'il doit employer pour elle une map et le nom de la map en question. Remarquez le système de `#`, propre aux ancres.

Les balises de la map

```
<map name="nom_du_map">
<area shape=méthode coords="coordonnées" href="lien" alt="commentaires" target="target_frame">
  ...autres balises AREA ...
</map>
```

L'attribut SHAPE

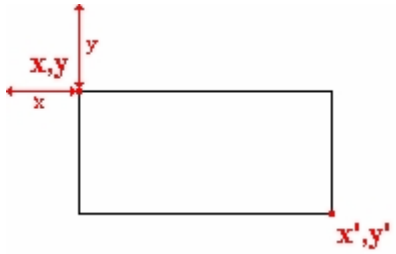
SHAPE = `rect` ou `circle` ou `polygon`

- `rect` pour un rectangle.
- `circle` pour un cercle.
- `polygon` pour un polygone irrégulier.

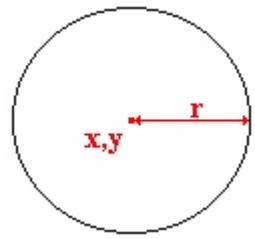
L'attribut COORDS

L'attribut COORDS note les coordonnées qui permettront au navigateur de reconstituer la forme géométrique.

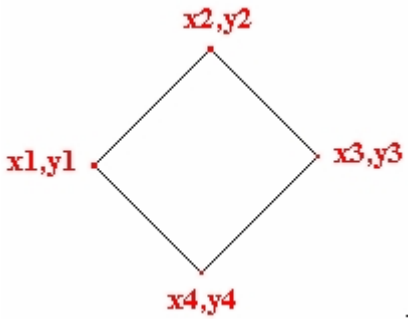
Pour un rectangle : `coords="x,y,x',y'"`



Pour un cercle : `coords="x,y,r"` soit le point central et le rayon



Pour un polygone : `coords="x1,y1,x2,y2,x3,y3,x4,y4"` et ce pour autant de points qu'il y a dans le polygone.



L'attribut LINK : L'attribut `link` spécifie le fichier associé à la zone sélectionnée. L'adressage se fait de la façon tout à fait classique en HTML.

L'attribut ALT (facultatif) : L'attribut `aLT` permet d'ajouter un commentaire. Ce commentaire, tout comme l'attribut `aLT` des images, sera affiché par certaines versions récentes de navigateur.

L'attribut TARGET (facultatif) : L'attribut `target` permet de spécifier la fenêtre de frame dans laquelle doit s'ouvrir le fichier spécifié.

XI.2.c) Les coordonnées

La véritable difficulté est de trouver les coordonnées des points dans l'image. Des logiciels de traitement de l'image (comme Paint Shop Pro) permettent de déterminer les coordonnées, mais bien plus facile des éditeurs (comme Frontpage ou Dreamweaver) permettent de dessiner directement les formes (`shape`) sur l'image et génère automatique le code et donc les coordonnées.

Le fichier créé ressemble à ceci :

```
<html> <body>
<center>
  <img src="cubes.gif" usemap="#cartons" height=121 width=136>
</center>
<map name="cartons">
  <area shape=rect coords="37,9,72,40" href="a.htm">
  <area shape=rect coords="18,46,46,79" href="b.htm">
  <area shape=rect coords="61,43,93,78" href="c.htm">
  <area shape=rect coords="9,84,36,119" href="d.htm">
  <area shape=rect coords="48,85,77,116" href="e.htm">
  <area shape=rect coords="89,81,123,115" href="f.htm">
</map>
</body> </html>
```

XII/ LES FORMULAIRES

Avec les formulaires, Html vous ouvre les portes de l'interactivité et vous permet de recevoir des informations provenant directement de votre lecteur et éventuellement de lui répondre directement.

Pour sortir de la page ou de votre ordinateur, les moyens disponibles sont :

- permettre à certains utilisateurs triés sur le volet, d'écrire sur le serveur et d'en exploiter les ressources. C'est la procédure des CGI, Php et autres langages. (voir plus loin)
- utiliser le courrier électronique. C'est la procédure mailto.
- transférer les informations en interne, à l'intérieur d'une page ou d'un site Web. C'est le Javascript.

XII.1- Définition d'un formulaire

Avant de pouvoir utiliser les différentes sortes de formulaires (ligne de texte, liste déroulante, cases à cocher...), il faut déclarer au browser qu'il devra gérer des formulaires et ce qu'il devra en faire.

```
<form name="f1" method="post" action="URL_Expédition" enctype="text/plain">
  instructions formulaires
</form>
```

Commentaires :

- L'attribut "method" vous offre le choix entre get et post. La différence entre ces deux méthodes repose sur la façon dont les données seront transmises au serveur et exploitées par celui-ci.
- L'attribut "action" spécifie l'adresse d'expédition des données.
  - Via un CGI : `<form method="post" action="http://www.serveur/cgi-bin/ma_cgi.pl">`
  - Via le protocole mailto : `<FORM method="post" action="mailto:antony@ldr.fr">`
- L'attribut "enctype" (optionnel) spécifie l'encodage utilisé pour le contenu du formulaire. Ce paramètre ne peut être utilisé qu'accompagné par la méthode post.

XII.2- Ligne de texte

input type="text" indique un champ de saisie d'une seule ligne.

```
<form>
  <input type="text" name="nom" size="50">
</form>
```

Commentaires :

- L'attribut name="nom_zone" va identifier la chaîne de caractères du champ de saisie.
- L'attribut size (optionnel) définit la longueur du champ de saisie.
- L'attribut maxlength="x" (optionnel) qui limite le nombre réel de caractères que l'on peut introduire dans le champ de saisie.
- La balise <input> n'a pas de balise de fin.

XII.3- Zone de saisie

La balise <textarea></textarea> introduit une zone de texte multilignes et non plus une simple ligne de texte.

```
<form>
  <textarea name="nom" rows=4 cols=40>
                              val par défaut</textarea>
</form>
```

Commentaires :

- L'attribut rows=x détermine le nombre de lignes de la zone de texte.
- L'attribut cols=y détermine le nombre de colonnes.
- L'attribut wrap (optionnel) détermine la façon dont les sauts de ligne seront traités lors d'un changement de ligne.
  - wrap=virtual, les changements de lignes sont effectués automatiquement dans la zone de texte, mais le tout sera transmis en une seule ligne.
  - wrap=physical, les changements de lignes sont effectués automatiquement dans la zone de texte et ceux-ci sont également transmis.
  - wrap=off, il n'y a aucun changement de ligne.

XII.4- Liste déroulante

La balise `<select></select>` indique au browser l'usage d'une liste déroulante. Les éléments de la liste sont introduits par la balise `<option> ... </option>`.

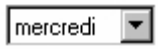
```
<form>
  <select name="nom" size="1">
    <option>lundi
    <option>mardi
    <option>mercredi
    <option>jeudi
    <option>vendredi
  </select>
</form>
```

Commentaire :

- L'attribut `size="x"` détermine le nombre d'éléments de liste affiché dans la boite d'entrée. En fait, `size="1"` est optionnel, car "1" est la valeur par défaut. Le même exemple avec `size="3"` donne :

- On peut présélectionner l'élément affiché dans la boite d'entrée (par défaut, le premier élément de la liste sera retenu). On utilise pour ce faire l'attribut `selected` de la balise `<option>` :

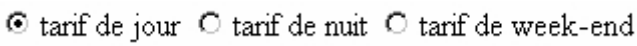
```
<form>
  <select name="nom" size="1">
    <option>lundi
    <option>mardi
    <option selected>mercredi
    <option>jeudi
    <option>vendredi
  </select>
</form>
```



XII.5- Bouton d'option

Il serait plus logique de parler de boutons d'option car ils n'ont de sens que s'ils sont plusieurs. Ainsi on parle d'un groupe de boutons d'options, aussi appelés boutons radio.

```
<form>
  <input type= "radio" name="tarif" value="jour" checked> tarif de jour
  <input type= "radio" name="tarif" value="nuit"> tarif de nuit
  <input type= "radio" name="tarif" value="week6end"> tarif de week-end
</form>
```



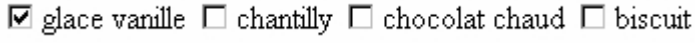
Commentaires :

- L'attribut `name="nom"` **doit avoir le même nom** : il forme le groupe de boutons d'option.
- L'attribut `"checked"` (optionnel) permet de présélectionner un bouton radio.
- Le contenu de l'attribut `"value"` du bouton retenu pourra être réutilisé plus tard.

XII.6- Case à cocher

La philosophie des cases à cocher [`checkbox`] est assez similaire aux boites d'option, cependant, plusieurs choix simultanés peuvent être réalisés.

```
<form>
  <input type="checkbox" name="choix1" value="1" checked> glace vanille
  <input type="checkbox" name="choix2" value="2"> chantilly
  <input type="checkbox" name="choix3" value="3"> chocolat chaud
  <input type="checkbox" name="choix4" value="4"> biscuit
</form>
```



Commentaire :

- L'attribut `name="nom"` doit employer des noms différents pour chaque case à cocher.
- L'attribut `checked` (optionnel) permet de présélectionner une case à cocher.
- Le contenu de l'attribut `"value"` du bouton retenu pourra être réutilisé plus tard.

XII.7- Bouton de commande

Avec le bouton de commande, il n'est pas nécessaire d'avertir le browser qu'on utilisera du `Javascript` par une balise du genre `<script language="javascript">`.

```
<form>
  <input type="button" name="nom" value= "Bouton de test" onclick="alert('Test réussi !')">
</form>
```

XII.8- Submit et Reset

Submit

Le bouton `Submit` a la tâche spécifique de transmettre toutes les informations contenues dans le formulaire à l'URL désignée dans les attributs `action` et `method` du tag `<form>`.

```
<form>
  <input type="submit" name="nom" value="Envoyer">
</form>
```

Reset

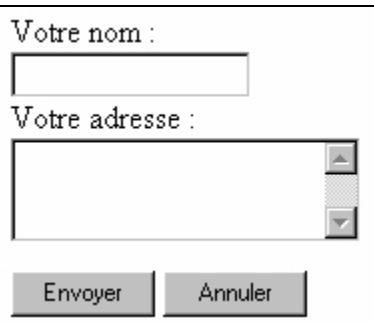
Le bouton `Reset` permet d'annuler les modifications apportées aux contrôles d'un formulaire et de restaurer les valeurs par défaut.

```
<form>
  <input type="reset" name="nom" valeu="Annuler">
</form>
```

XII.9- Exemple : Un livre d'or

Une des premières applications des formulaires est celle d'un livre d'or [`guestbook`]. Voici donc un exemple de livre d'or que vous pouvez adapter :

```
<form method="post" action="mailto:adr_mail">
  Votre nom :<br>
  <input type="text" name="nom"><br>
  Votre adresse :<br>
  <textarea name="adresse" rows="2" cols="35">
  </textarea><br>
  <input type="submit" value="Envoyer">
  <input type="reset" value="Annuler">
</form>
```



LE XHTML

XHTML est l'acronyme de eXtensible HyperText Markup Language, amener à remplacer le HTML. XHTML 1.0 est devenu une recommandation officielle le 26 janvier 2000, c'est du HTML 4.01 + du XML.

I/ LES NOUVEAUTES

Les documents xhtml sont conformes à xml, et donc directement lisibles et editables avec des outils xml. Il fonctionne quelques soient les navigateurs et surtout avec les nouveaux.

Les documents xhtml peuvent être utilisés dans des applications qui reposent sur DOM HTML autant que DOM XML.

I.1- Séparer le fonds de la forme

Le développement de site web était une affaire de professionnels, il est aujourd'hui une affaire de spécialistes. Ainsi, le but est à nouveau de séparer le fonds de la forme.

Le code de la page ne doit concerner que la structure et la mise en forme doit être gérée par des feuilles de style (voir chapitre suivant).

Les cadres, ne respectant pas les critères d'accessibilités, deviennent obsolètes (mais encore utilisés).

I.2- Structure minimale

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3C.org/TR/xhtm/DTD
                                                                    /xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
  <head>
    <title> Ma premiere page XHTML</title>
    <meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
  </head>
  <body>
    <p>Bonjour tout le monde !</p>
  </body>
</html>
```

La déclaration du type de document

Le fichier minimal doit contenir le format de codage des caractères :

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

La norme ISO 8859-1, dont le nom complet est ISO/CEI 8859-1, et qui est souvent appelée Latin-1, forme la première partie de la norme internationale ISO/CEI 8859, qui est une norme de l'Organisation internationale de normalisation pour le codage des caractères en informatique.

Ainsi que la déclartion du type de document :

```
<!DOCTYPE HTML PUBLIC ".//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3C.org/TR
                                                                    /xhtml1-strict.dtd">
```

Le fichier ci-dessus présente du XHTML 1.0.

L'informatique fonctionnant par évolution, le prédécésseur était du HTML4.0, vous pourriez ainsi trouvez de manière similaire :

```
<!DOCTYPE HTML PUBLIC ".//W3C//DTD HTML4.01//EN" "http://www.w3C.org/TR/html4/strict.dtd">
```

La balise HTML

Cette dernière doit être agrémentée de l'attribut xmlns. La définition de cet espace de nommage permet au navigateur de s'informer sur la provenance et le langage des éléments.

```
<html xmlns="http://www.w3C.org/1999/xhtm">
  Contenu de la page
</html>
```

L'attribut lang= est remplacé par xml:lang=. Il permet de définir (optionellement) la langue et le sens de lecture.

```
<html lang="fr" dir="ltr">
```

- la langue : fr
- le sens de lecture : ltr = left to right (et l'inverse : rtl = right to left)

I.3- Le respect du squelette

La rigueur apportée par le xml est l'une des bases du XHTML, ainsi :

- Toute balise ouverte doit être fermée.
- Tout doit être écrit en minuscule.
- Il faut éviter les passages à la ligne inutile.

Les éléments vides

Certaines balises ne possèdent pas de balise de fin, ainsi la balise <img> ne nécessitait pas de balise </img>. Il est maintenant nécessaire de les fermer par **/>.**

```
<img src="photo.jpg" alt="Ma photo" />
```

Attention à l'espace obligatoire avant le />.

Les ancres

Certaines balises manquaient de précisions, ainsi une ancre se définissait ainsi :

```
<a href="#mot_clé"> Texte </a>
```
qui pointe vers
```
<a name="mot_clé"> Texte 2 </a>
```

En xhtml, il faut maintenant utiliser l'attribut universel id= pour identifier l'ancre :

```
<a href="#mot_clé"> Texte </a>
```
qui pointe vers
```
<a id="mot_clé" name="mot_clé"> Texte 2 </a>
```

Les éléments <script> et <style>

Le contenu des éléments <script> et <style> doit se trouver à l'intérieur d'une section marquée CDATA, ceci pour éviter la transformation des signes <, > et &.

```
<script>
  < ![CDATA [contenu du script] ]>
</script>
```

En outre, elles doivent contenir l'attribut type : <style type="text/css"> … </style>.

Le type Mine

Le type mine (multipurpose Internet Mail Extensions) peut se touver dans l'entete de la page. Il indique le type des données contenues dans la page. Il permet une vérification du type de fichier plus sûre que par la méthode de l'extension.

```
type="test/css"
type="application/pdf"
type="application/xhtml+xml"
```

Les règles d'imbrication

- L'élément a ne doit contenir aucun autre élément a.
- L'élément label ne doit contenir aucun autre élément label.
- L'élément form ne doit contenir aucun autre élément form.
- L'élément pre ne doit contenir aucun élément img, object, big, small, sub ou sup.
- L'élément button ne doit contenir aucun élément input, select, textarea, label, button, form, field-set, iframe ou isindex.

I.4- En résumé …

- Il est nécessaire d'écrire en minuscule.
- Supprimez toutes les balises de mise en forme et utilisez une feuille de style.
- Les éléments xhtml doivent être correctement imbriqués, pas d'entrelacement.
- Toutes les balises doivent être fermées. Les balises vides se ferment ainsi :
.
- Toutes les valeurs d'attributs doivent être mises entre guillemets, y compris les valeurs numériques.

II/ LA VALIDATION DU DOCUMENT

Le World Wide Web Consortium, abrégé W3C, est un consortium fondé en octobre 1994 pour promouvoir la compatibilité des technologies du World Wide Web telles que HTML, XHTML, XML, CSS, PNG, SVG et SOAP. Le W3C n'émet pas des normes, mais des recommandations.

Ainsi toutes les règles précédentes ne sont en faites que des recommandations et non une norme (mais cela deviendra peut une norme de fait).

« Chacun pouvant apporter sa pierre à l'édifice », ces recommandations évoluent constatment, il est donc important de se tenir au courant : http://www.w3.org/.

II.1- Xhtml : Valide ou non valide ?

Le W3C a mis en place un « validateur » permettant de confirmer (ou non) la conformité de votre page aux recommandations.

Ainsi après avoir écrit une page, pensez à la tester : http://validator.W3.org/ :

Notez qu'une page valide permet d'obtenir le message suivant :

Attention, c'est « presque gagné » si vous obtenez :

Et c'est « raté » si vous obtenez :

II.2- Un gage de qualité

Une fois votre document valide, vous aurez la possibilité d'ajouter le code suivant :

```
<p>
  <a href="http://validator.w3.org/check?uri=referer">
    <img src="http://www.w3.org/Icons/valid-xhtml10"
    alt="Valid XHTML 1.0 Strict" height="31" width="88" />
  </a>
</p>
```

Il vous permettra de faire apparaître l'icône ci-dessous sur votre site, gage de qualité de votre site.

II.3- Css : Valide ou non valide ?

Vous pouvez faire de même avec les feuilles de styles (voir chapitre suivant) :
http://jigsaw.w3.org/css-validator/

III/ XHTML 2.0

XHTML 2.0 est en cours de développement. Il ne sera pas compatible en ascendance et en descendance, notamment à cause de l'introduction de XForms et de XFrames.

Dès lors nombreux sont les spécialistes qui pensent qu'en raison du manque de compatibilité avec les anciens navigateurs, le XHTML 2.0 ne se répandra que très lentement.

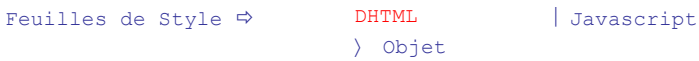
L'INTERACTIVITE

I/ DHTML OU LE HTML DYNAMIQUE

Le DHTML n'est pas un langage de balises, de scripts ou de programmation. Le DHTML n'est pas une quelconque spécification.

Les technologies que le DHTML met en oeuvre sont :

- **Le HTML**, nécessaire pour présenter le document ;
- **Les feuilles de style** (CSS), permettant de définir un style pour plusieurs objets et le positionnement de ceux-ci ;
- **Le modèle objet de document** (DOM), proposant une hiérarchie d'objets, afin de faciliter leur manipulation ;
- **Le Javascript**, un petit langage de script essentiel pour définir des événements utilisateur (ou VBScript).



II/ LES FEUILLES DE STYLE

II.1- Concept

Le concept de feuilles de style est apparu en 1996 avec la publication par le W3C d'une nouvelle recommandation intitulée « Cascading StyleSheets » (feuilles de style en cascade), notée CSS.

Le principe des feuilles de style consiste à regrouper dans un même document des caractéristiques de mise en forme associées à des groupes d'éléments. Il suffit de définir par un nom un ensemble de définitions et de caractéristiques de mise en forme, et de l'appeler pour l'appliquer à un texte.

Ainsi il est ainsi possible de créer un groupe de titres en police Arial, de couleur verte et en italique. Et dès lors, lorsque la charte graphique d'un site composé de plusieurs centaines de pages web doit être changée, il suffit de modifier la définition des feuilles de style en un seul endroit pour changer l'apparence du site tout entier !

Elles sont appellées « feuilles de style en cascade » (en anglais « Cascading Style Sheets »), car il est possible d'en définir plusieurs et que les styles peuvent être hérités en cascade. La priotité devient dès lors : **Style en ligne > Style du document > Style importé > Style externe.**

Depuis le 12 mai 1998, la norme CSS 2.0 est le standard en vigueur.

II.1.a) Avantages

- ***Séparation du contenu et de la mise en forme***.
- ***Cohésion de la présentation*** tout au long du site avec les feuilles de style externes.
- Modifier l'aspect d'un page ou d'un site sans en modifier le contenu et cela en quelques lignes plutôt que de devoir changer un grand nombre de balises.
- Une façon d'écriture concise et nette par rapport au Html, qui devient vite fouillis.
- Réduire le temps de chargement des pages.

II.1.b) Définition d'un style

La définition de base d'un style est simple :

```
balise { propriété de style : valeur; propriété de style : valeur }
```

Exemple :

```
H3 { font-family : Arial; font-style : italic }
```

La balise H3 sera en Arial et en italique. Et dans le document, toutes les balises <H3> seront au style Arial et italique.

II.2- Implantation des styles

Il faut maintenant incorporer les styles dans le document Html. Cela peut se faire par « styles internes », « styles externes » ou « styles importés ».

II.2.a) Styles internes

A l'intérieur des balises <HEAD></HEAD>

Soit :

```
<HTML>
  <HEAD>
    <STYLE type="text/css">
      <!-- La ou les feuille(s) de style -->
    </STYLE>
  </HEAD>
<BODY>
…
```

- La balise `<STYLE>` avertit le navigateur que l'on va utiliser des feuilles de style.
- L'attribut `type="text/css"` informe que ce qui suit est du texte et qu'il s'agit de `cascading style sheets (css)`.
- La balise Html de commentaires `<!-- ... -->` empêche que les browsers qui ne connaissent pas les feuilles de style, tentent d'interpréter ces instructions.
- Pour vos propres commentaires à propos des feuilles de style, on utilisera la convention désormais classique de `/* commentaires */`.

Exemple :

```
<HTML>
  <HEAD>
    <STYLE type="text/css">
      <!-- H3 { font-family : Times New Roman; font-style : italic } --!>
    </STYLE>
  </HEAD>
<BODY>
<H1> coucou </H1>
<P>
<H3> coucou </H3>
</BODY> </HTML>
```

A l'intérieur des balises <BODY></BODY>

On peut aussi utiliser, au coup par coup, les feuilles de style dans le corps (`BODY`) du document. Cette façon de faire est peu conforme à l'esprit des feuilles de style, qui est de définir un style déterminé valable pour la globalité du document et séparé du code. Mais elle existe pour quelques utilisations spécifiques...

Exemple :

```
<HTML>
  <BODY>
    /* Le style ne porte que sur cette balise */
    <H1 style="font-family: Arial; font-style: italic"> coucou </H1>
  </BODY>
</HTML>
```

II.2.b) Styles externes

`CSS` nous propose de définir une présentation de style valable pour plusieurs pages et même pour toutes les pages d'un site. Ceci est possible, en créant une page externe, qui regroupera toutes les feuilles de style, et en reliant chaque page à cette page de style.

1- Création du fichier avec l'extension `.css (ex: styles.css),` qui contiendra toutes les feuilles de style :

```
/* File name   : CascadeStyleSheet                  // Commentaires
*/
H1 {                                                // Définition style H1
  font-family: verdana,arial,helvetica,serif;
  font-size: 30pt;
  font-weight: bold;
```

```
  color: #003366;
  text-align: right}
H2 {                                                    // Définition style H2
  font-family: verdana,arial,helvetica,serif;
  font-size: 10pt;
  color: #CC88CC;
  text-align: center}
```

2- Intégration et utilisation de la feuille de style dans une page Web (`page1.htm`) :

```
<HTML>
  <HEAD>
    <LINK rel = stylesheet type = "text/css" href = "styles.css">
  </HEAD>
<BODY>
<H1> coucou </H1>
<P>
</BODY> </HTML>
```

Commentaires :

- La balise `<LINK>` avertit le browser qu'il faudra réaliser un lien vers un fichier.
- L'attribut `rel=stylesheet` précise qu'il y trouvera une feuille de style externe.
- L'attribut `type="text/css"` précise que l'information est du texte et du genre `cascading style sheets`.
- L'attribut classique de lien `href=" ... "` donne le chemin d'accès et le nom du fichier à lier.

II.2.c) Style importé

Les recommandations du `W3C` offrent une dernière façon d'inclure des feuilles de style dans un document : en important des feuilles de style. Il est en effet possible d'importer des feuilles de style externes au niveau de la déclaration du style de document, en insérant la commande `@IMPORT` immédiatement après la balise style.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
    <STYLE type="text/css">
    <!--
      @IMPORT URL(url de la feuille à importer);
      // Définition des styles du document;
    -->
    </STYLE>
  </HEAD>
<BODY></BODY>
</HTML>
```

II.3- Les classes et les ID

II.3.a) Notion de classes

On peut souhaiter affecter des styles différents à une même balise, ceci sera possible avec des classes `[class]`.

La définition d'un style était :

```
balise { propriété de style: valeur }
```

Elle devient :

```
balise.nom_de_classe { propriété de style: valeur }
```

ou, comme la mention de la balise est facultative :

```
.nom_de_classe { propriété de style: valeur }
```

L'emploi du point (.) devant le nom de classe est indispensable.

Pour appeler l'effet de style dans le document, on ajoute le nom de la classe à la balise :

```
<balise class="nom_de-classe"> .... </balise>
```

Exemple :

1- Mettre ce qui est important dans le texte en gras et en bleu ⇨ création de la classe `.essentiel`

```
.essentiel { font-weight: bold; font-color: #000080 }
```

2- Dans le document Html, il suffit d'appeler la classe `essentiel` quand cela s'avère nécessaire :

Utilisation sur "texte normal"

```
<P class = "essentiel"> blabla important </P>
```

Utilisation sur balise H1

```
<H1 class = "essentiel">Titre 1</H1>
```

Utilisation dans cellule de tableau

```
<TABLE><TR><TD class = "essentiel">cellule</TD></TD>
```

II.3.b) Notion des ID

Comme la convention `nom.nom` est utilisée aussi en `Javascript`, il a fallu trouver une autre convention d'écriture lorsqu'on désire utiliser les feuilles de style avec du Javascript.

Ce sont les `ID`, aussi appelés les identifiants. Les `ID` fonctionnent exactement comme les classes.

La syntaxe est :

```
#nom_de_ID { propriété de style: valeur }
```

Et pour l'appeler :

```
<balise id="nom_de_ID"> .... </balise>
```

II.3.c) Les pseudo-classes

Les `pseudo-classes` permettent d'affiner le style appliqué à un certain nombre de balises en définissant une réaction à un événement ou bien à la position relative de la balise au sein des autres balises.

Contrairement aux classes, le nom des pseudo-classes est prédéfini, il n'est donc pas possible de créer ses propres pseudo-classes. Autres inconvénients : elles ne sont pas toutes reconnues par l'ensemble des navigateurs.

Les pseudo-classes dynamiques

Les `pseudo-classes dynamiques` permettent de modifier le style d'une balise en fonction d'un événement (mouvement de la souris, clic, ou bien appui sur une touche du clavier).

- ✓ La `pseudo-classe :hover` permet d'affecter un style à la balise sélectionnée lors d'un survol par le curseur de la souris : `A:hover {font-decoration: underline;}`
- ✓ La `pseudo-classe :focus` permet de définir un style à la balise sélectionnée lorsque le focus lui est donné (par exemple lors d'un clic dans un élément de formulaire) : `TEXTAREA:focus {color: #FF0000;}`
- ✓ La `pseudo-classe :active` permet de définir un style à la balise sélectionnée lorsque l'utilisateur clique sur l'élément (entre le moment où l'utilisateur clique sur le bouton de la souris et celui où il le relâche) : `A:active {color: #FF0000;}`

Les pseudo-classes de lien

Les `pseudos-classes de lien` sont des pseudo-classes spécifiques à la balise `<A>` :

- ✓ La `pseudo-classe :link` permet de définir le style des liens hypertextes n'ayant pas encore été consultés par le client.
- ✓ La `pseudo-classe :visited` permet de définir le style des liens hypertextes que le client a déjà visités.

La pseudo-classe descendante

Une `pseudo-classe descendante` permet d'appliquer un style à la première balise au sein d'un élément. La syntaxe de cette pseudo-classe est la suivante : `Element_Parent > Balise:first-child {style;}`

Ainsi la déclaration suivante s'applique à la première balise `<A>` situé dans un jeu de balises `<P> </P>` :

```
P > A:first-child {color: #00FF00;}
```

Les pseudo-classes de texte

Les `pseudo-classes de texte` permettent d'appliquer un style à une partie du texte délimité par les balises auxquelles ils s'appliquent. Ainsi les pseudo-classes de texte s'utilisent généralement conjointement avec la balise de paragraphe (`<P>`).

- ✓ `:first-line` : permet d'appliquer un style à la première ligne d'un paragraphe.

```
P:first-line { text-transform: uppercase }
```

- ✓ `:first-letter` : permet d'appliquer un style à la première lettre d'un paragraphe afin de produire un effet typographique. L'exemple suivant par exemple double la taille et met en gras la première lettre d'un paragraphe.

```
P:first-letter { font-size: 200%; font-weight: bold; }
```

Les pseudo-classes de langue

Il est possible de définir un style en fonction de la langue du document (spécifiée dans les en-têtes HTTP ou dans les balises méta) ou de la langue d'un élément `HTML` ou `XML` (spécifié grâce à l'attribut optionnel `LANG`) si celle-ci est précisée.

- ✓ `:lang(Langue)`. La pseudo classe de langue suivante permet de définir les guillemets selon la notation française `HTML`.

```
lang(fr) { quotes: '« ' ' »' }
```

Les pseudo-classes de page

Le sélecteur `@page` permet de modifier les définitions de mise en page d'une page `HTML` (taille, marge, etc.) à l'impression, telles que les marges (`margin-left, margin-top, margin-right, margin-bottom`), la taille (`size`). Il faut alors imaginer la page web comme un ensemble de pages constituant un ouvrage papier.

- ✓ `@page:left` : permet de définir les propriétés des pages de gauche (inversement : `@page:right`).

```
@page:left { size: landscape; margin-left: 2cm; }
```

- ✓ `@page:first` : permet de définir les propriétés de la première page d'un document.

II.3.d) Les sélecteurs de balise

On appelle « `sélecteur de balise` » (ou « `sélecteur d'éléments` ») le ou les mots clés précédant l'accolade et servant à indiquer le ou les balises du document auxquelles le style entre accolades s'applique.

Commentaires :

- Il n'y a pas de limite pour le nombre de couples "propriétés de style/valeur".
- Vous pouvez écrire vos styles sur plusieurs lignes :

```
H3 {font-family: Arial;
    font-style: italic;
    font-color: green}
```

- On peut attribuer plusieurs valeurs à une même propriété (séparées par des virgules) :

```
H3 {font-family: Arial, Helvetica, sans-serif}
```

Pour définir le style d'une balise `HTML` spécifique, il suffit de mettre le nom de la balise (sans les caractères `<` et `>`) :

```
<HTML><HEAD>
  <STYLE type="text/css">
    <!--
      balise {propriétés}
    -->
  </STYLE>
</HEAD>
<BODY>
  <balise> ... </balise>
</BODY></HTML>
```

Sélecteur multiple

Il est également possible d'appliquer le style à plusieurs balises en séparant le nom de ces balises par une virgule (`,`).

```
selecteur-de-balise1, selecteur-de-balise2  { /* style */  }
```

Sélecteur universel

Grâce au sélecteur universel (« `*` ») il est possible de définir un style s'appliquant à tous les éléments HTML :

```
* { /* style */  }
```

Sélecteur d'éléments imbriqués

Il permet de créer une règle ne s'appliquant que lorsque un élément Y est imbriqué dans un élément X.

```
selecteur_X selecteur_Y { /* style; */ }
```

Exemple :

```
<p> <i> Attention </i>, ceci est un <b> avertissement </b> </p>
<b> Prière d'en tenir compte </b>
```

La règle `P B { font-weight: bold; color: red; }` ne sélectionne que le mot « avertissement » (le seul entouré de balises `<B>`, elles-mêmes imbriquées dans une balise `<P>`).

Le sélecteur d'éléments consécutif

Il permet de créer une règle ne s'appliquant que lorsque un élément Y suit immédiatement un élément X.

```
selecteur_X + selecteur_Y { /* style; */ }
```

Exemple :

```
<p> <i> Attention </i>, ceci est un <b> avertissement </b> </p>
<b> Prière d'en tenir compte </b>
```

La règle `I + B { font-weight: bold; color: red; }` ne sélectionne que le mot « avertissement » (le seul entouré de balises `<B>`, elles-mêmes suivant une balise `<I>`).

Le sélecteur d'éléments père-fils

Il permet de créer une règle ne s'appliquant que lorsque un élément Y est fils direct d'un élément X.

```
selecteur_X > selecteur_Y { /* style; */ }
```

Exemple :

```
<p> <b><i> Attention </i></b>, ceci est un <i><b> avertissement </b></i> </p>
<b> Prière d'en tenir compte </b>
```

La règle `P > B { font-weight: bold; color: red; }` sélectionne les éléments « Attention » et « avertissement » (ils sont entourés de balises `<B>`, elles-mêmes directement imbriquées dans une balise `<P>`).

II.4- <SPAN> et <DIV>

II.4.a) Utilité

Il fallait penser à un système pour "déconnecter" certains morceaux de paragraphe ou plusieurs paragraphes de cette logique d'écriture avec des feuilles de style. Ce sont respectivement les balises `SPAN` et `DIV` qui créaient ainsi des petits blocs particuliers dans le document sans devoir repasser par les éléments structurels du `Html` classique.

Notons que `SPAN` et `DIV` s'utilisent toujours avec les classes et les ID.

II.4.b) SPAN

La balise `<SPAN> ... </SPAN>` permet d'appliquer des styles à des éléments de texte d'un paragraphe ou si vous préférez à un morceau de paragraphe. Ainsi je voudrais écrire :

```
Un monde de géants!!!
```

Soit :

```
<HTML>
  <HEAD>
    <STYLE type="text/css">
      .element{font-size: x-large; color: navy}
    </STYLE>
  </HEAD>
  <BODY>
    <P>Un monde de <SPAN class=element>géants</SPAN> !!!</P>
  </BODY> </HTML>
```

II.4.c) DIV

La balise `<DIV> ... </DIV>` permet de regrouper plusieurs paragraphes ou si vous préférez, de délimiter une zone comportant plusieurs paragraphes. Ainsi :

```
La balise DIV
Commentaire :
N'oubliez pas l'attribut class!
```

Soit :

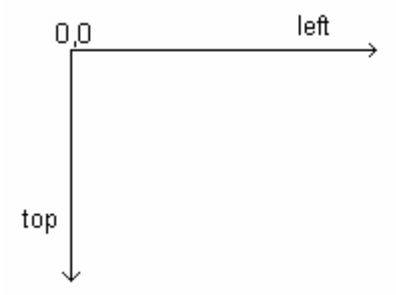
```
<HTML>
  <HEAD>
    <STYLE type="text/css">
```

```
      .zone{font-size: x-small}
    </STYLE>
  </HEAD>
  <BODY>
    La balise DIV
    <DIV class=zone>
      <P>Commentaire :</P>
      <P>N'oubliez pas l'attribut class!</P>
    </DIV>
  </BODY> </HTML>
```

II.5- Positionner avec CSS

Outre la balise `<LAYER>` (Netscape uniquement), il est désormais possible de positionner, au pixel près, du texte ou une image avec les feuilles de style.

La position absolue `{position: absolute}` se détermine par rapport au coin supérieur gauche de la fenêtre du browser. Les coordonnées de ce point sont `top = 0` et `left = 0`. Les coordonnées d'un point s'expriment en pixels, de haut en bas pour top et de gauche à droite pour left.



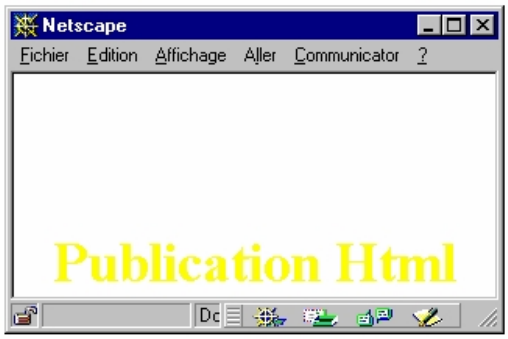
La position relative `{position: relative}` se détermine par rapport à d'autres éléments de la page, par exemple un élément du code Html.

II.5.a) Exemple : Positionner du texte

Plaçons en position absolue le texte "Publication Html" à 50 pixels du haut de la fenêtre (top) et à 150 pixels à gauche (left) :

```
<HTML>
  <HEAD>
    <STYLE type="text/css">
      .pub{position: absolute; top: 100px; left: 25px;
       color: yellow; font-size: x-large; font-weight: bold;}
     </STYLE>
  </HEAD>
  <BODY>
    <DIV class=pub> Publication Html </DIV>
  </BODY> </HTML>
```

ce qui donne :

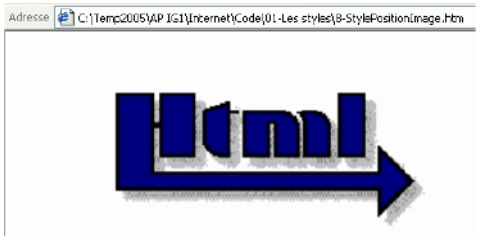


II.5.b) Exemple : Positionner une image

Plaçons l'image `htmlplus.gif` en position absolue à `50 pixels` de haut de la fenêtre (top) et à `100 pixels` à gauche (left). Les dimensions de l'image sont `width=242` et `height=84` :

```
<HTML>
  <BODY>
    <span style="position: absolute; top: 50px; left: 100px; width: 242px; heigth: 84px;">
      <IMG src="htmlplus.gif">
```

```
    </span>
  </BODY>
</HTML>
```



II.5.c) Exemple : Superposer du texte sur une image

Reprenons l'image `htmlplus.gif` et on y superposera le mot `PLUS`, au pixel près, dans la barre qui souligne le terme Html :

```
<HTML>
  <BODY>
    <span style="position: absolute; top: 50px; left: 100px; width: 242px; heigth: 84px;">
      <IMG src="htmlplus.gif">
    </span>
    <span style="position: absolute; top: 135px; left: 200px;
    color: yellow; font-size: x-small; font-weight: bold;">
      PLUS
    </span>
  </BODY>
</HTML>
```

II.5.d) Les unités CSS

Avec les balises DIV et SPAN, le positionnement peut se faire de deux façons :

- ✓ **Le positionnement absolu** se détermine par rapport au coin supérieur gauche de la fenêtre du navigateur. Les coordonnées d'un point s'expriment alors de haut en bas (`top`) et de gauche à droite (`left`).

`cm`	Le centimètre
`in`	Le pouce (en anglais « inch ») correspondant à 2,54 cm
`mm`	Le millimètre
`pt`	Le point
`pc`	Le pica (correspondant à 12 pt)

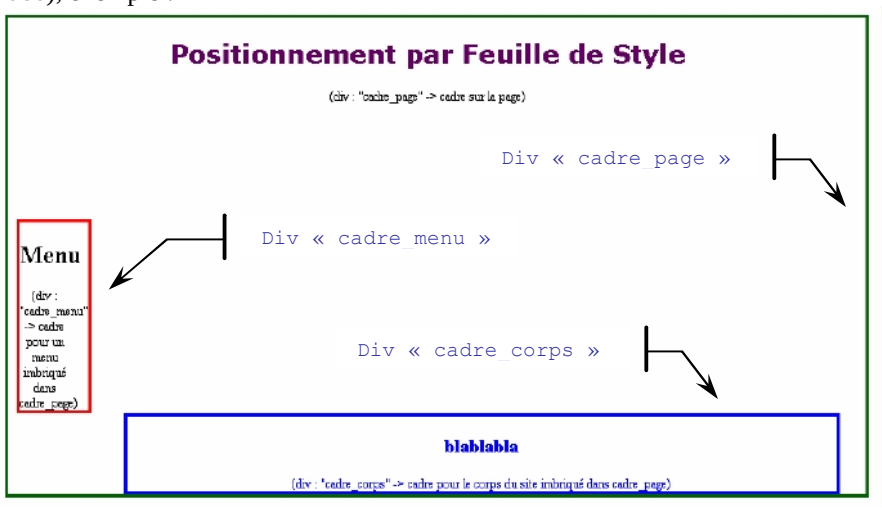
- ✓ **Le positionnement relatif** se fait par rapport à d'autres éléments, comme une image, c'est-à-dire que les éléments contenus dans la balises `DIV` ou `SPAN` seront positionnés à la suite des éléments HTML après lesquels ils se trouvent.

`em`	Unité relative à la taille de police de l'élément sélectionné. Seule exception à cette règle : lorsque la propriété font-size est définie, elle se rapporte à la taille de la police de l'élément parent.
`ex`	Unité relative à la hauteur de la minuscule de l'élément sélectionné. Seule exception à cette règle : lorsque la propriété font-size est définie, elle se rapporte à la hauteur de la minuscule de l'élément parent.
`px`	Le pixel. Il s'agit d'une unité dont le rendu dépend de la résolution du périphérique d'affichage.
`%`	Le pourcentage est une unité relative à la taille de l'élément ou de son parent.

II.6- Les balises DIV en tant que conteneurs

La balise `div` est une balise qui représente la notion de regroupement dans une « `boite` » ou un « `bloc` ». A contrario, la balise `span` définit le contenu comme étant en ligne.

Ainsi, la présentation d'une page web peut être obtenue et/ou améliorée par une combinaison de balises de « styles » (span, div, id et class), exemple :



Soit la page xhtml ci-dessus : 2-Div pour Css.htm

```
<?xml … >
<html … >
  <head>
    <link rel="stylesheet" type="text/css" href="2-Styles.css" />
  </head>
  <body>
  <div id="cadre_page">
    <h1> Positionnement par Feuille de Style </h1>
    (div : "cadre_page" -> cadre sur la page)

    <div id="cadre_menu">
      <h2> Menu </h2>
      (div : "cadre_menu" -> cadre pour un menu imbriqué dans cadre_page)
    </div>

    <div class="cadre_corps">
      <h3> blablabla </h3>
      (div : "cadre_corps" -> cadre pour le corps du site imbriqué dans cadre_page)
    </div>
  </div>
  </body>
</html>
```

Après avoir placé l'entête d'une page xhtml, la page découpe sa structure en 3 blocs, chacun de ses blocs trouve sa définition dans la feuille de style ci-dessous : 2-Styles.css

```
H1 {
  font-family: verdana,arial,helvetica,serif;
  font-size: 25px;
  color: purple;
}
```

Positionnement

```
#cadre_page {
  text-align: center;
  color: black;
  font-size: 12px;
  border: solid green 3 px;
  margin-right: 20px;
  margin-left: 20px;
}
```

Le texte est centré sur la page et de couleur noire. Le cadre se positionne sur l'ensemble de la page en laissant de chaque coté une marge de 20px.

```
#cadre_menu {
  text-align: center;
  color: black;
  font-size: 12px;
  border: solid red 3 px;
  margin-top: 3cm;
  margin-left: 0.2cm;
  margin-right: 20cm;
}
```

Le texte est centré dans le cadre et de couleur noire. Le cadre est à 3cm du haut et 0,2 cm du bord gauche. Le positionnement à 20 cm du bord droit **permet de définir sa taille**.

```
.cadre_corps {
  color: blue;
  font-size: 12px;
  border: solid blue 3px;
  margin-left: 3cm;
```

Le texte est de couleur bleu. Le cadre est à 3cm du bord gauche et sa définition (dans la page xhtml) en dernière position le positionne derrière le cadre_menu.

```
    margin-right: 0.2cm;
}
```

Astuce
Noter que chaque style de cadre possède une définition de sa bordure, telle que `border: solid blue 3px;`. Ceci permet de mieux visualiser les cadres, à la construction de la page. Vous pourrez ensuite mettre `les bordures à 0px`, ou mettre ces lignes en commentaires pour cacher les cadres. (Firefo propose un utilitaire pour cette fonctionnalité en plugs-in)

II.7- FAQ sur les styles

1- Les feuilles de style sont-elles "case sensitive" ?

Non, les feuilles de style ne sont pas sensibles à la case [`case insensitive`].

2- Qui l'emportent : les attributs Html ou les propriétés CSS ?

Les propriétés des feuilles de style l'emportent sur les attributs Html. Si les deux sont spécifiés, les attributs Html seront affichés avec les browsers qui ne supportent pas les CSS et n'auront aucun effet avec les browsers qui supportent les feuilles de style (Ce qui est très bien pour la compatibilité mais qui ne simplifie pas la clarté de l'écriture).

3- Et s'il y a concurrence entre plusieurs éléments de style ?

En cas de concurrence entre plusieurs éléments de style, intervient alors la notion de "cascade" ou d'ordre de priorité.

La concurrence entre plusieurs éléments de style peut provenir des différentes possibilités de localisation de feuilles de style :

- dans un fichier externe avec l'extension `.css.`
- dans la balise `HEAD` du document.
- dans le `BODY` du document.

La règle de priorité appliquée par le browser sera d'appliquer pour la présentation du document la feuille de style la plus proche de l'élément. Ainsi, un style spécifié dans le `BODY` sera retenu par rapport à un style déclaré dans un fichier externe.

Il y a cependant moyen de contourner ces règles de priorité par la déclaration `! important` :

```
BODY {background: white ! important; color: black}
```

III/ LE "DOCUMENT OBJET MODEL" : DOM

III.1- Le DOM

Le `Modèle Objet de Document (DOM)` est une interface de programmation d'applications (API) pour documents `HTML` et `XML`. Il définit la structure logique des documents et la manière dont un document est accédé et manipulé.

Dans `DOM`, les documents ont une structure logique comparable à un arbre. Une propriété importante des modèles de structure `DOM` est l'isomorphisme structural : si deux implémentations distinctes du Modèle Objet de Documents sont utilisées pour créer une représentation du même document, elles créeront le même modèle structurel, avec très précisément les mêmes objets et les mêmes relations entre ces objets.

Le nom "`Modèle Objet de Documents`" a été choisi parce qu'il s'agit d'un "`modèle objet`" au sens traditionnel de la conception orientée objet : les documents sont modélisés en utilisant des objets, et le modèle ne contient pas uniquement la structure du document mais également son comportement et celui des objets dont il est composé. En tant que modèle objet, `DOM` identifie :

- les interfaces et les objets utilisés pour représenter et manipuler un document ;
- la sémantique de ces interfaces et objets - incluant le comportement et les attributs ;
- les relations et collaborations entre ces interfaces et ces objets.

III.2- La structure du DOM

`DOM` présente les documents sous la forme d'une hiérarchie d'objets `Node (noeuds)` qui sont des objets implémentant eux-mêmes d'autres interfaces plus spécialisées. Certains types de noeuds peuvent avoir des noeuds enfants de types variés, et d'autres sont des feuilles (noeuds terminaux) pouvant ne rien avoir en dessous d'eux dans la structure du document.

```
Document          -- Element (un au plus), ProcessingInstruction, Comment, DocumentType
DocumentFragment  -- Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
EntityReference   -- Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
Element           -- Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference
Entity            -- Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
Attr              -- Text, EntityReference
DocumentType      -- pas d'enfant
Comment           -- pas d'enfant
Text              -- pas d'enfant
CDATASection      -- pas d'enfant
Notation          -- pas d'enfant
ProcessingInstruction   -- pas d'enfant
```

De plus, DOM spécifie l'interface NodeList (liste de noeuds) pour gérer des listes ordonnées de Nodes, comme par exemple les listes constituées des enfants d'un Node ou les éléments retournés par la méthode Element.getElementsByTagName, ainsi que l'interface NamedNodeMap (tableaux de noeuds nommés) qui permet de gérer des ensembles non ordonnés de noeuds référencés par leurs noms d'attributs tels que les attributs d'un Element. NodeList et NamedNodeMap sont dynamiques dans DOM, c'est à dire que les changements dans la structure du document sous-jacent sont réfléchis dans toutes les NodeList et les NamedNodeMap s'y rapportant. Par exemple, si un utilisateur de DOM obtient une NodeList contenant les enfants d'un Element et qu'ensuite il ajoute de nouveaux enfants à cet élément (ou en retranche ou en modifie), ces changements sont automatiquement répercutés dans la NodeList sans que l'utilisateur n'ait aucune action particulière à faire. De la même manière, les changements faits sur un Node de l'arbre sont répercutés dans toutes les références faites à ce Node dans les NodeLists et les NamedNodeMaps.

III.3- Manipulations

Le DOM se divise en trois ensembles d'objets : le Core DOM est une interface bas-niveau, compacte mais minimale, qui, si elle est capable de représenter tout document HTML ou XML, n'a pas la facilité d'utilisation de l'HTML DOM ou de l'XML DOM, les deux autres ensembles d'objets permettant des accès plus directs à des types de document plus spécifiques.

Les objets du DOM sont les nœuds d'une arborescence. Parmi les différents nœuds figurent (du plus haut au plus bas niveau) : le document, les éléments (balises), les attributs (des balises), les commentaires, ou encore les contenus textuels.

Créer un nouveau nœud appartenant, par exemple, à cette dernière catégorie, pourra être réalisé par la déclaration Javascript suivante :

```
var txt = document.createTextNode("Un nouveau contenu textuel…");
```

Un élément (ici, une balise <A>) et son attribut (ici HREF) seront, quant à eux, déclarés comme suit :

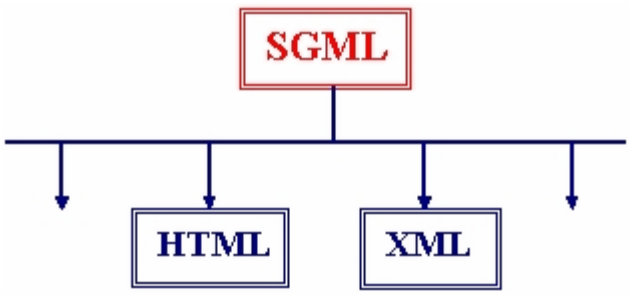
```
var lien = document.createElement('a');
lien.setAttribute = ('href', "www.google.fr");
```

IV/ LE XML

D'abord le SGML (pour Standard Generalized Markup Language), est une norme internationale (ISO 8879), qui régit la structure et le contenu des différents types de documents électroniques. Le SGML est en quelque sorte un "language-mère" utilisé pour décrire les milliers de documents électroniques que l'on retrouve dans les multiples domaines de l'activité humaine.

Le HTML (HyperText Markup Language) n'est qu'un de ces types de documents avec ses balises imposées et qui est utilisé pour les documents électroniques qui circulent sur le Web.

Avec le XML, on retourne aux sources du SGML en permettant au "designer" de définir son propre type de document et, côté spectaculaire de la chose, de créer ses propres balises (voir le X de eXtensible).



(Ce langage sera étudié ultérieurement)

V/ LE JAVASCRIPT

V.1- Introduction

Javascript est un langage de scripts qui incorporé aux balises Html, permet d'améliorer la présentation et l'interactivité des pages Web.

Ces scripts vont être gérés et exécutés par le browser lui-même sans devoir faire appel aux ressources du serveur. Ces instructions seront donc traitées en direct et surtout sans retard par le navigateur.

Javascript a été initialement développé par Netscape et s'appelait alors LiveScript. Adopté à la fin de l'année 1995, par la firme Sun (qui a aussi développé Java), il prit alors son nom de Javascript.

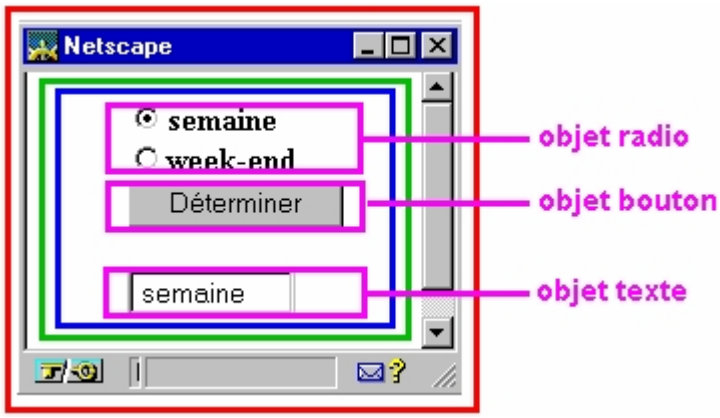
V.1.a) Javascript n'est pas Java

Il importe de savoir que Javascript est totalement différent de Java.

Javascript	Java
Code intégré dans la page Html	Module (applet) distinct de la page Html
Code interprété par le browser au moment de l'exécution	Code source compilé avant son exécution
Codes de programmation simples, mais pour des applications limitées	Langage de programmation beaucoup plus Complexe, mais plus performant
Permet d'accéder aux objets du navigateur	N'accède pas aux objets du navigateur
Confidentialité des codes nulle (code source visible)	Sécurité (code source compilé)

V.1.b) Un peu de théorie objet

Prenons un écran de page Web :



Vous trouvez :

- en rouge (1er cadre) : une page qui s'affiche dans une fenêtre. C'est l'objet fenêtre.
- en vert (2eme cadre) : dans cette fenêtre, il y a un document Html. C'est l'objet document.
- en bleu (3eme cadre) : dans ce document, on trouve un formulaire. C'est l'objet formulaire.
- en rose : Dans ce document, on trouve trois objets. Ceux sont respectivement l'objet radio, l'objet bouton et l'objet texte.

La hiérarchie des objets est donc :

fenêtre document formulaire radio bouton texte

Pour accéder à un objet, il faudra donner le chemin complet de l'objet en allant du contenant le plus extérieur à l'objet à l'objet référencé.

Exemple pour le bouton radio "semaine" :

```
(window).document.form.radio[0].
```

Nous avons mis l'objet window entre parenthèses, car comme il occupe la première place dans la hiérarchie, il est repris par défaut par Javascript et devient donc facultatif.

Et enfin pour les puristes, Javascript n'est pas à proprement parler un langage orienté objet tel que C++ ou Java. On dira plutôt que Javascript est un langage basé sur les objets.

V.1.c) Les propriétés des objets

Une propriété est une caractéristique, une description de l'objet. Par exemple, l'objet livre a comme propriétés son auteur, sa maison d'édition, son titre, son numéro ISBN, etc…

Pour accéder aux propriétés, on utilise la syntaxe :
```
nom_de_l'objet.nom_de_la_propriété
```

Dans le cas du bouton radio "semaine", pour tester la propriété de sélection, on écrira :
```
document.form.radio[0].checked
```

V.2- Le Javascript minimum

V.2.a) La balise <SCRIPT>

De ce qui précède, vous savez déjà que votre script vient s'ajouter à votre page Web. Le langage `Html` utilise des `tags` ou `balises` pour "dire" au browser d'afficher une portion de texte en gras, en italique, etc.

Dans la logique du langage Html, il faut donc signaler au browser par une balise, que ce qui suit est un script et que c'est du Javascript (et non du VBScript). C'est la balise `<SCRIPT LANGUAGE="Javascript">`. De même, il faudra informer le browser de la fin du script : `</SCRIPT>`.

V.2.b) Les commentaires

Tout ce qui est écrit entre le // et la fin de la ligne sera ignoré :
```
// commentaire
```

Il sera aussi possible d'inclure des commentaires sur plusieurs lignes avec le code :
```
/* commentaire sur
plusieurs lignes */
```

V.2.c) Masquer le script pour les anciens browsers

Les browsers qui ne comprennent pas le Javascript ignorent la balise `<script>` et vont essayer d'afficher le code du script sans pouvoir l'exécuter. Pour éviter l'affichage peu esthétique de ses inscriptions cabalistiques, on utilisera les balises de commentaire du langage Html `<!-- ... -->`.

Votre premier Javascript ressemblera à ceci :
```
<SCRIPT LANGUAGE="javascript">
<!-- Masquer le script pour les anciens browsers
...
programme Javascript
...
// Cesser de masquer le script -->
</SCRIPT>
```

Attention, ôter les balises de commentaires si le script ne fonctionne pas !

V.2.d) Où inclure le code en Javascript ?

Le principe est simple. Il suffit de respecter les deux principes suivants :

- n'importe où.
- mais là où il le faut.

Le browser traite votre page `Html` de haut en bas (y compris vos ajouts en `Javascript`). Par conséquent, toute instruction ne pourra être exécutée que si le browser possède à ce moment précis tous les éléments nécessaires à son exécution. Ceux-ci doivent donc être déclarés avant ou au plus tard lors de l'instruction.

Pour s'assurer que le programme script est chargé dans la page et prêt à fonctionner à toute intervention de votre visiteur, on prendra l'habitude de déclarer un maximum d'éléments dans les balises d'en-tête : `<HEAD>` et `</HEAD>` (exemple pour les fonctions).

V.2.e) Une première instruction Javascript

Javascript est "bon enfant", car il n'est pas toujours trop strict sur la syntaxe, mais attention le xhtml l'est !

```
          Html normal   <HTML> <HEAD> <TITLE>Mon premier Javascript</TITLE> </HEAD>
                        <BODY>
                        Bla-bla en Html
      Début du script   <SCRIPT LANGUAGE="Javascript">
    Masquer le script   <!--
               Script   alert("votre texte");
     Fin de masquer     //-->
      Fin du script     </SCRIPT>
        Html normal     Suite bla-bla en Html
                        </BODY> </HTML>
```

`Alert()` est une méthode de l'objet `Window` et pour se conformer à la notation `nom_objet.nom_propriété`, on aurait pu noter `window.alert()`.

Commentaires :

- Javascript est case sensitive.
- Les guillemets " et l'apostrophe ' font partie intégrante du langage Javascript. On peut utiliser l'une ou l'autre forme à condition de ne pas les mélanger. Si vous souhaiter utiliser des guillemets dans vos chaînes de caractères, tapez `\"` ou `\'` pour les différencier vis à vis du compilateur.
- Si vous souhaitez que votre texte de la fenêtre `alert()` s'inscrive sur plusieurs lignes, il faudra utiliser le caractère spécial `/n` pour créer une nouvelle ligne.

V.2.f) Extension .js pour scripts externes

Il est possible d'utiliser des fichiers externes pour les programmes Javascript. On peut ainsi stocker les scripts dans des fichiers distincts (avec l'extension `.js, ex : monScript.js`) et les appeler à partir d'un fichier Html. Le concepteur peut de cette manière se constituer une bibliothèque de script. La balise devient :

maPage.htm

```
<HTML>
<HEAD>
<TITLE>Fonction Javascript</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE='javascript' SRC='monScript.js'></SCRIPT>
</BODY> </HTML>
```

monScript.js

```
alert("votre texte");
```

V.3- Afficher du texte

V.3.a) Exemple de méthode de l'objet document : Write

La page Html qui s'affiche dans la fenêtre du browser est un objet de type document.

A chaque objet Javascript, le concepteur du langage a prévu un ensemble de méthodes (ou fonctions dédiées à cet objet) qui lui sont propres. A l'objet `document`, Javascript a dédié la méthode "écrire dans le document" : c'est la méthode `write()`. Ainsi pour appeler la méthode `write()` du `document`, on notera :

```
document.write();
```

La méthode write()

La syntaxe classique est :	`write("votre texte");`
On peut aussi écrire une variable (resultat) :	`write(resultat);`
L'opérateur de concaténation est le + :	`write("Le résultat est " + resultat);`
On peut utiliser les balises Html :	`write("<B>Le résultat est</B>" + resultat);`

Mais aussi …

Soit :		`str="Something";`
`big();`	correspond aux balises Html : `<BIG> </BIG>`	`document.write("<BIG>"+str+"</BIG>");` ou `document.write(str.big());` ou `document.write("Something".big());`
`small();`	correspond aux balises Html :	`document.write(str.small());`

	`<SMALL> </SMALL>`	
`blink();`	correspond aux balises Html : `<BLINK> </BLINK>.`	`document.write(str.blink());`
`bold();`	correspond aux balises Html : `<B> </B>`	`document.write(str.bold());`
`fixed();`	correspond aux balises Html : `<TT> </TT>`	`document.write(str.fixed());`
`italics();`	correspond aux balises Html : `<I></I>`	`document.write(str.italics());`
`fontcolor(color );`	correspond aux balises Html : `<FONT COLOR="color"> </FONT>`	`str2="red";` `document.write(str.fontcolor(str2));`
`fontsize(x);`	correspond aux balises Html : `<FONT SIZE="x"></FONT>`	`x=3;` `document.write(str.fontsize(3));`
`strike();`	correspond aux balises Html : `<STRIKE></STRIKE>`	`document.write(str.strike());`
`sub();`	correspond aux balises Html : `<SUB></SUB>`	`document.write(str.sub());`
`sup()`	correspond aux balises Html : `<SUP></SUP>`	`document.write(str.sup());`

V.3.b) Les instructions de formatage de document

Ce qui suit est optionnel puisque vous pouvez tout faire à l'aide de l'instruction `document.write()`:

```
document.write("<BODY BGCOLOR="#FFFFFF");
```

`BgColor`	spécifier la couleur d'arrière-plan	`document.bgColor="white";` `document.bgColor="#FFFFFF";`
`fgColor`	spécifier la couleur d'avant-plan (texte)	`document.fgColor="black";` `document.fgColor="#000000";`
`alinkColor`	spécifier la couleur d'un lien actif	`document.alinkColor="white";` `document.alinkColor="#FFFFFF";`
`linkColor`	spécifier la couleur d'un hyperlien	`document.linkColor="white";` `document.linkColor="#FFFFFF";`
`vlinkColor`	spécifier la couleur d'un hyperlien déjà visité	`document.linkColor="white";` `document.linkColor="#FFFFFF";`

V.4- Utiliser des variables

V.4.a) La déclaration de variable

Les variables peuvent se déclarer de deux façons :

- soit de façon explicite.

```
var Numero = 1
var Prenom = "Luc"
```

- soit de façon implicite (à éviter).

```
Numero = 1
Prenom = "Luc"
```

V.4.b) Les données sous Javascript

Type	Description
Des nombres	Tout nombre entier ou avec virgule tel que 22 ou 3.1416
Des chaînes de caractères	Toute suite de caractères comprise entre guillemets telle que "suite de caractères"
Des booléens	Les mots `true` pour vrai et `fase` pour faux
Le mot `null`	Mot spécial qui représente pas de valeur

Commentaire :
Les variables n'ont pas de type statique, c'est-à-dire qu'elles prennent dynamiquement le type des valeurs qu'elles contiennent. Elles peuvent même changer de type en cours de script, même si cette pratique est déconseillée.

Les noms réservés

Les mots de la liste ci-après ne peuvent être utilisés pour des noms de fonctions et de variables.

A	`abstract`
B	`boolean break byte`
C	`case catch char class const continue`
D	`default do double`
E	`else extends`

F	`false final finally float for function`
G	`goto`
I	`if implements import in instanceof int interface`
L	`long`
N	`native new null`
P	`package private protected public`
R	`return`
S	`short static super switch synchronized`
T	`this throw throws transient true try`
V	`var void`
W	`while with`

V.4.c) Variables globales et variables locales

Les variables déclarées tout au début du script, en dehors et avant toutes fonctions, seront toujours globales.

Dans une fonction :

- une variable déclarée par le mot clé var aura une portée locale (limitée à cette seule fonction).
- une variable déclarée contextuellement (sans utiliser le mot var) aura une portée globale.

V.5- Les opérateurs

Les variables, c'est bien mais encore faut-il pouvoir les manipuler ou les évaluer.

Les opérateurs de calcul

Soit :

```
x = 11
```

Signe	*Nom*	*Signification*	*Exemple*	*Résultat*
+	plus	addition	`x + 3`	`14`
-	moins	soustraction	`x - 3`	`8`
*	multiplié par	multiplication	`x * 2`	`22`
/	divisé	par division	`x / 2`	`5.5`
%	modulo	reste de la division par	`x % 5`	`1`
=	a la valeur	affectation	`x = 5`	`5`

Les opérateurs de comparaison

Signe	**Nom**	**Exemple**	**Résultat**
==	égal	`x == 11`	`true`
<	inférieur	`x < 11`	`false`
<=	inférieur ou égal	`x <= 11`	`true`
>	supérieur	`x > 11`	`false`
=<	supérieur ou égal	`x >= 11`	`true`
!=	différent	`x != 11`	`false`

Les opérateurs associatifs

Soit :

```
x = 11 et y = 5.
```

Signe	*Description*	*Exemple*	*Signification*	*Résultat*
+=	plus égal	`x += y`	`x = x + y`	`16`
-=	moins égal	`x -= y`	`x = x - y`	`6`
*=	multiplié égal	`x *= y`	`x = x * y`	`55`
/=	divisé égal	`x /= y`	`x = x / y`	`2.2`

Les opérateurs logiques

Signe	**Nom**	**Exemple**	**Signification**
&&	et	`(condition1) && (condition2)`	`condition1 et condition2`
||	ou	`(condition1) || (condition2)`	`condition1 ou condition2`

Les opérateurs d'incrémentation

Signe	Description	Exemple	Signification	Résultat
x++	incrémentation (x++ est le même que x=x+1)	`y = x++`	`3 puis plus 1`	`4`
x--	décrémentation (x-- est le même que x=x-1)	`y= x--`	`3 puis moins 1`	`2`

La priorité des opérateurs Javascript

Opération	Opérateur
,	virgule ou séparateur de liste
= += -= *= /= %=	affectation
? :	opérateur conditionnel
||	ou logique
&&	et logique
== !=	égalité
< <= >= >	relationnel
+ -	addition soustraction
* /	multiplier diviser
! - ++ --	unaire

V.6- Les fonctions

V.6.a) Déclaration des fonctions

On utilise le mot (réservé) `function` :

```
function nom_fonction(arguments) {
  instructions ...
}
```

Il est conseillé de placer la déclaration de vos fonctions entre les balises `<HEAD> </HEAD>`.

V.6.b) L'appel d'une fonction

L'appel se fait par le nom de la fonction :

```
nom_fonction();
```

Exemple :
Soit la fonction `message()`, qui affiche le texte "Bienvenue à ma page". Cette fonction sera appelée au chargement de la page voir `onLoad=` dans le tag `<BODY>`.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="Javascript">
  <--
  function message() {
    document.write("Bienvenue à ma page"); }
  //-->
</SCRIPT>
</HEAD>
<BODY onLoad="message()">
</BODY>
</HTML>
```

V.6.c) Passer une ou plusieurs valeurs à une fonction

On peut passer des valeurs aux fonctions Javascript via les paramètres.

Exemple :

```
<SCRIPT LANGUAGE="Javascript">
  function msg (arg1, arg2) {
    alert (arg1 + " " + arg2);
  }
  </SCRIPT>
```

Appel de la fonction :

```
<BODY onload="msg ('coucou','antony')">
```

V.6.d) Retourner une valeur

Pour renvoyer un résultat, il suffit d'écrire le mot clé `return` suivi de l'expression à renvoyer. Notez qu'il ne faut pas entourer l'expression de parenthèses :

```
function cube(nombre) {
  var cube = nombre * nombre * nombre ;
  return cube;
}
```

V.7- Les événements

Les événements Javascript, associés aux fonctions, aux méthodes et aux formulaires, ouvrent grand la porte pour une réelle interactivité de vos pages.

V.7.a) Les événements

Description	Evénement
Lorsque l'utilisateur clique sur un bouton, un lien ou tout autre élément.	`Clik`
Lorsque la page est chargée par le browser ou le navigateur.	`Load`
Lorsque l'utilisateur quitte la page.	`Unload`
Lorsque l'utilisateur place le pointeur de la souris sur un lien ou tout autre élément.	`MouseOver`
Lorsque le pointeur de la souris quitte un lien ou tout autre élément.	`MouseOut`
Lorsque un élément de formulaire a le focus c-à-d devient la zone d'entrée active.	`Focus`
Lorsque un élément de formulaire perd le focus c-à-d que l'utilisateur clique hors du champs et que la zone d'entrée n'est plus active.	`Blur`
Lorsque la valeur d'un champ de formulaire est modifiée.	`Change`
Lorsque l'utilisateur sélectionne un champ dans un élément de formulaire.	`Select`
Lorsque l'utilisateur clique sur le bouton Submit pour envoyer un formulaire.	`Submit`

V.7.b) Les gestionnaires d'événements

Il faut qu'à ces événements soient associées des actions prévues par vous. C'est le rôle des gestionnaires d'événements. La syntaxe est :

```
OnEvénement = "fonction()"
```

Exemple :

```
onClick = "alert('Vous avez cliqué sur cet élément')"
```

// sur clic de l'utilisateur, ouvrir une boite d'alerte avec le message indiqué.

onClick

Le clic de la souris :

```
<FORM>
<INPUT TYPE="button" VALUE="Cliquez ici" onClick="alert('Vous avez bien cliqué ici')">
</FORM>
```

Nous reviendrons en détail sur les formulaires dans le chapitre suivant.

onLoad et onUnload

L'événement `Load` survient lorsque la page a fini de se charger. A l'inverse, `Unload` survient lorsque l'utilisateur quitte la page.

Les événements `onLoad` et `onUnload` sont utilisés sous forme d'attributs de la balise `<BODY>` ou `<framset>`. On peut ainsi écrire un script pour souhaiter la bienvenue à l'ouverture d'une page et un petit mot d'au revoir au moment de quitter celle-ci :

```
<HTML>
<HEAD>
  <SCRIPT LANGUAGE='Javascript'>
    function bienvenue() {
      alert("Bienvenue à cette page");
    }
    function au_revoir() {
      alert("Au revoir");
    }
  </SCRIPT>
</HEAD>
<BODY onLoad='bienvenue()' onUnload='au_revoir()'>
  Html normal
</BODY>
</HTML>
```

onMouseOver

L'événement `onMouseOver` se produit lorsque le pointeur de la souris passe au dessus (sans cliquer) d'un lien ou d'une image. Cet événement est fort pratique pour, par exemple, afficher des explications soit dans la barre de statut soit avec une petite fenêtre genre infobulle.

```
<A HREF="" onmouseOver="action()">lien</A>
```

Exemple : Par le survol du lien "message important", une fenêtre d'alerte s'ouvre.

```
<BODY>
  <A HREF="" onmouseOver="alert('Coucou')">message important</A>
<BODY>
```

ou si vous préférez utiliser les balises `<HEAD>`

```
<HTML>
<HEAD>
  <SCRIPT language="Javascript">
    function message(){ alert("Coucou") }
  </SCRIPT>
</HEAD>
<BODY>
  <A HREF="" onmouseOver="message()">message important</A>
</BODY>
</HTML>
```

onMouseOut

L'événement `onMouseOut`, généralement associé à un `onMouseOver`, se produit lorsque le pointeur quitte la zone sensible (lien ou image).

```
<A HREF="" onmouseOver="alert('Coucou')" onmouseOut="alert('Au revoir')"> msg important</A>
```

Commentaire :
Afin de ne pas avoir d'erreur sur un click hasardeux de l'utilisateur sur la syntaxe :

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

prenez l'habitude de mettre l'adresse de la page encours ou plus simplement le signe # :

```
<A HREF="#" onmouseOver="action()"> lien </A>.
```

onFocus

L'événement `onFocus` survient lorsqu'un champ de saisie a le focus c.-à-d. quand son emplacement est prêt à recevoir ce que l'utilisateur à l'intention de taper au clavier. C'est souvent la conséquence d'un clic de souris ou de l'usage de la touche "Tab".

onBlur

L'événement `onBlur` a lieu lorsqu'un champ de formulaire perd le focus. Cela se produit quand l'utilisateur ayant terminé la saisie qu'il effectuait dans une case, clique en dehors du champ ou utilise la touche "Tab" pour passer à un champ. Cet événement sera souvent utilisé pour vérifier la saisie d'un formulaire.

```
<FORM>
<INPUT TYPE=text onBlur="alert('Ceci est un Blur')">
</FORM>
```

onchange

Cet événement s'apparente à l'événement `onBlur,` mais avec une petite différence. Non seulement la case du formulaire doit avoir perdu le focus, mais aussi son contenu doit avoir été modifié par l'utilisateur.

onselect

Cet événement se produit lorsque l'utilisateur a sélectionné (mis en surbrillance ou en vidéo inverse) tout ou partie d'une zone de texte dans une zone de type `text` ou `textarea`.

V.7.c) Gestionnaires d'événement disponibles

Objets	Gestionnaires d'événement disponibles
Fenêtre	`onLoad, onUnload`
Lien hypertexte	`onClick, onMouseOver, onMouseOut`
Elément de texte	`onBlur, onChange, onFocus, onSelect`
Elément de zone de texte	`onBlur, onChange, onFocus, onSelect`

Elément bouton	onClick
Case à cocher	onClick
Bouton Radio	onClick
Liste de sélectionon	Blur, onChange, onFocus
Bouton Submit	onClick
Bouton Reset	onClick

Changement d'images

Avec le gestionnaire d'événement onMouseOver, on peut prévoir qu'après le survol d'un image par l'utilisateur, une autre image apparaisse (pour autant qu'elle soit de la même taille).
Exemple :

```
<HTML> <HEAD>
  <SCRIPT LANGUAGE="Javascript1.1">
    function lightUp() { document.images["monImage"].src="ok_on.gif" }
    function dimDown() { document.images["monImage"].src="ok.gif" }
</SCRIPT>
</HEAD>
<BODY>
  <A HREF="#" onmouseOver="lightUp();" onmouseOut="dimDown();">
  <IMG SRC="ok.gif" name="monImage" width=100 height=50 border=0> </A>
</BODY> </HTML>
```

Commentaire : L'image invisible.
On peut prévoir une image invisible (de la même couleur que l'arrière plan ou même transparente). On la place sur le chemin de la souris de l'utilisateur et son survol peut ,à l'insu de l'utilisateur, déclencher l'action de votre choix.

V.8- Les conditions

Si… alors… Sinon…	`if (condition vraie) { instructions1; } else { instructions2; }`
Si… alors… Sinon…	`(expression) ? inst1 : inst2`
Pour i de <val init> à <cond> pas <prog> Faire …	`for (val init ; cond ; prog) { instructions; }`
Tant Que <condition> Faire …	`while (condition vraie){ instructions }`
Interruption volontaire de boucle	`Break ;`
Saut d'instructions	`Continue`

V.9- Les formulaires

En Javascript, on peut accéder à chaque élément d'un formulaire pour, par exemple, y aller lire ou écrire une valeur, noter un choix auquel on pourra associer un gestionnaire d'événement, ...

Un formulaire est l'élément Html déclaré par les balises `<FORM></FORM>`. Un formulaire contient un ou plusieurs éléments que nous appellerons des contrôles (`widgets`). Ces contrôles sont notés par exemple par la balise `<INPUT TYPE= ...>`.

V.9.a) Déclaration d'un formulaire

La déclaration d'un formulaire se fait par les balises `<FORM> et </FORM>`.

L'attribut `NAME="nom_du_formulaire"` a toute son importance pour désigner le chemin des éléments.

Les attributs `ACTION` et `METHOD` sont facultatifs (pour autant que vous ne faites pas appel au serveur).

V.9.b) Le contrôle ligne de texte

La syntaxe Html est :

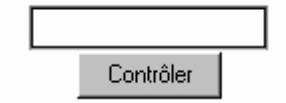
```
<INPUT TYPE="text" NAME="nom" SIZE=x MAXLENGTH=y>
```

Propriété	*Description*
`name`	indique le nom du contrôle par lequel on pourra accéder.
`Defaultvalue`	indique la valeur par défaut qui sera affichée dans la zone de texte.

`Value`	indique la valeur en cours de la zone de texte.

Lire une valeur dans une zone de texte

Exemple : Entrez une valeur et appuyer sur le bouton pour contrôler celle-ci.



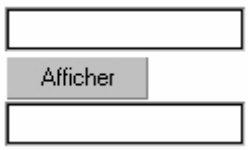
Soit :

```
<HTML> <HEAD>
  <SCRIPT LANGUAGE="javascript">
    function controle(form1) {
      var test = document.form1.input.value;
      alert("Vous avez tapé : " + test);
    }
</SCRIPT>
</HEAD>
<BODY>
  <FORM NAME="form1">
    <INPUT TYPE="text" NAME="input" VALUE=""><BR>
    <INPUT TYPE="button" NAME="bouton" VALUE="Contrôler" onClick="controle(form1)">
  </FORM>
</BODY> </HTML>
```

Commentaire :
Lorsqu'on clique le bouton "`contrôler`", Javascript appelle la fonction `controle()` à laquelle on passe le formulaire dont le nom est `form1` comme argument.
Cette fonction `controle()` définie dans les balises `<HEAD>` prend sous la variable test, la valeur de la zone de texte. Pour accéder à cette valeur, on note le chemin complet de celle-ci. : `document.form1.input.value.`

Ecrire une valeur dans une zone de texte

Exemple : Entrez une valeur quelconque dans la zone de texte d'entrée. Appuyer sur le bouton pour afficher cette valeur dans la zone de texte de sortie.



Zone de texte d'entrée

Zone de texte de sortie

Soit :

```
<HTML> <HEAD>
  <SCRIPT LANGUAGE="javascript">
    function afficher(form2) {
      var testin = document. form2.input.value;
      document.form2.output.value=testin;
    }
</SCRIPT>
</HEAD>
<BODY>
  <FORM NAME="form2">
    <INPUT TYPE="text" NAME="input" VALUE=""> Zone de texte d'entrée <BR>
    <INPUT TYPE="button" NAME="bouton" VALUE="Afficher" onClick="afficher(form2)"><BR>
    <INPUT TYPE="text" NAME="output" VALUE=""> Zone de texte de sortie
</FORM> </BODY> </HTML>
```

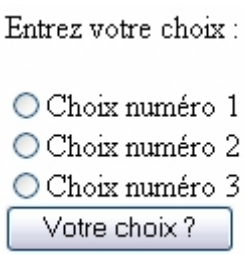
Commentaire:
Lorsqu'on clique le bouton "`Afficher`", Javascript appelle la fonction `afficher()` à laquelle on passe le formulaire dont le nom est cette fois `form2` comme argument.
Cette fonction `afficher()` définie dans les balises `<HEAD>` prend sous la variable `testin`, la valeur de la zone de texte d'entrée. A l'instruction suivante, on dit à Javascript que la valeur de la zone de texte `output` comprise dans le formulaire nommé `form2` est celle de la variable `testin`.

V.9.c) Les boutons radio

Les boutons radio sont utilisés pour noter un choix, et seulement un seul, parmi un ensemble de propositions.

Propriété	Description
`name`	indique le nom du contrôle. Tous les boutons portent le même nom.
`index`	l'index ou le rang du bouton radio en commençant par 0.
`checked`	indique l'état en cours de l'élément radio
`defaultchecked`	indique l'état du bouton sélectionné par défaut.
`value`	indique la valeur de l'élément radio.

Prenons un exemple :



Soit :

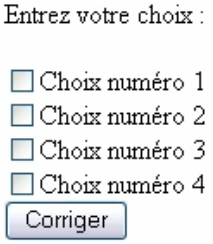
```
<HTML> <HEAD>
  <SCRIPT language="javascript">
    function choixprop(form3) {
      if (form3.choix[0].checked) { alert("Proposition " + form3.choix[0].value) };
      if (form3.choix[1].checked) { alert("Proposition " + form3.choix[1].value) };
      if (form3.choix[2].checked) { alert("Proposition " + form3.choix[2].value) };
    }
  </SCRIPT>
</HEAD>
<BODY>
  Entrez votre choix :
  <FORM NAME="form3">
    <INPUT TYPE="radio" NAME="choix" VALUE="1">Choix numéro 1<BR>
    <INPUT TYPE="radio" NAME="choix" VALUE="2">Choix numéro 2<BR>
    <INPUT TYPE="radio" NAME="choix" VALUE="3">Choix numéro 3<BR>
    <INPUT TYPE="button"NAME="but" VALUE="Votre choix ?" onClick="choixprop(form3)">
  </FORM>
</BODY> </HTML>
```

Commentaire :
Dans le formulaire nommé `form3`, on déclare trois boutons radio. Vient ensuite un bouton qui déclenche la fonction `choixprop()`.
Cette fonction teste quel bouton radio est coché. On accède aux boutons sous forme d'indice par rapport au nom des boutons radio soit `choix[0], choix[1], choix[2]`. On teste la propriété `checked` du bouton par `if(form3.choix[x].checked)`. Dans l'affirmative, une boite d'alerte s'affiche. Ce message reprend la valeur attachée à chaque bouton par le `chemin form3.choix[x].value`.

V.9.d) Les boutons case à cocher (checkbox)

Les boutons case à cocher sont utilisés pour noter un ou plusieurs choix parmi un ensemble de propositions.

Propriété	Description
`name`	indique le nom du contrôle. Toutes les cases à cocher portent un nom différent.
`checked`	indique l'état en cours de l'élément case à cocher.
`defaultchecked`	indique l'état du bouton sélectionné par défaut.
`value`	indique la valeur de l'élément case à cocher.



Soit :

```
<HTML> <HEAD>
  <script language="javascript">
```

```
    function reponse(form4) {
      if ( (form4.check1.checked) == true && (form4.check2.checked) == true &&
                              (form4.check3.checked) == false && (form4.check4.checked) == true) {
        alert("C'est la bonne réponse! ")
      } else {
        alert("Désolé, continuez à chercher.")
      }
    }
  </SCRIPT>
</HEAD>
<BODY>
  Entrez votre choix :
  <FORM NAME="form4">
    <INPUT TYPE="CHECKBOX" NAME="check1" VALUE="1">Choix numéro 1<BR>
    <INPUT TYPE="CHECKBOX" NAME="check2" VALUE="2">Choix numéro 2<BR>
    <INPUT TYPE="CHECKBOX" NAME="check3" VALUE="3">Choix numéro 3<BR>
    <INPUT TYPE="CHECKBOX" NAME="check4" VALUE="4">Choix numéro 4<BR>
    <INPUT TYPE="button"NAME="but" VALUE="Corriger" onClick="reponse(form4)">
  </FORM>
</BODY> </HTML>
```

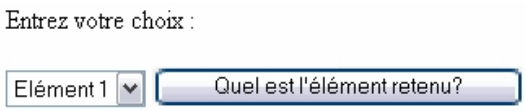
V.9.e) Liste de sélection

Le contrôle liste de sélection vous permet de proposer diverses options sous la forme d'une liste déroulante dans laquelle l'utilisateur peut cliquer pour faire son choix. Ce choix reste alors affiché.

La boite de la liste est crée par la balise `<SELECT>` et les éléments de la liste par un ou plusieurs tags `<OPTION>`.

Propriété	Description
`name`	indique le nom de la liste déroulante.
`length`	indique le nombre d'éléments de la liste. S'il est indiqué dans le tag `<SELECT>`, tous les éléments de la liste seront affichés. Si vous ne l'indiquez pas un seul apparaîtra dans la boite de la liste déroulante.
`selectedIndex`	indique le rang à partir de 0 de l'élément de la liste qui a été sélectionné par l'utilisateur.
`defaultselected`	indique l'élément de la liste sélectionné par défaut. C'est lui qui apparaît alors dans la petite boite.

Exemple :



Soit:

```
<HTML> <HEAD>
  <script language="javascript">
    function liste(form5) { alert("L\'élément " + (form5.list.selectedIndex + 1)); }
  </SCRIPT>
</HEAD>
<BODY>
  Entrez votre choix : <FORM NAME="form5">
  <SELECT NAME="list">
    <OPTION VALUE="1">Elément 1
    <OPTION VALUE="2">Elément 2
    <OPTION VALUE="3">Elément 3
  </SELECT>
  <INPUT TYPE="button"NAME="b" VALUE="Quel est l'élément retenu?" onClick="liste(form5)">
</FORM> </BODY> </HTML>
```

Commentaire :
Le chemin complet de l'élément sélectionné est `form5.nomdelaliste.selectedIndex`. Comme l'index commence à 0, il ne faut pas oublier d'ajouter 1 pour retrouver le juste rang de l'élément.

V.9.f) Le contrôle textarea

L'objet `textarea` est une zone de texte de plusieurs lignes.

```
<FORM>
  <TEXTAREA NAME="nom" ROWS=x COLS=y>
    texte par défaut
  </TEXTAREA>
</FORM>
```

Propriété	*Description*
`name`	indique le nom du contrôle par lequel on pourra accéder.

defaultvalue	indique la valeur par défaut qui sera affichée dans la zone de texte.
value	indique la valeur en cours de la zone de texte. Soit celle tapée par l'utilisateur ou si celui-ci n'a rien tapé, la valeur par défaut.
Rows	Nombre de lignes.
Cols	Nombre de colonnes.

Méthodes associées : `onFocus(), onBlur(), onSelect() et onChange()`.

V.9.g) Le contrôle Reset

Le contrôle `Reset` permet d'annuler les modifications apportées aux contrôles d'un formulaire et de restaurer les valeurs par défaut.

```
<INPUT TYPE="reset" NAME="nom" VALUE "texte">
```

Méthode associée : `onClick()`.

V.9.h) Le contrôle Submit

Le contrôle a la tâche spécifique de transmettre toutes les informations contenues dans le formulaire à l'URL désignée dans l'attribut `ACTION` du tag `<FORM>`.

```
<INPUT TYPE="submit" NAME="nom" VALUE "texte">
```

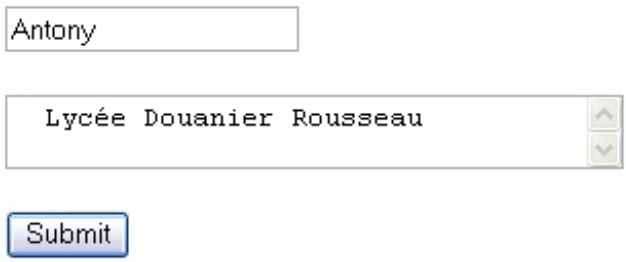
Méthode associée : `onClick()`.

V.9.i) Le contrôle Hidden (caché)

Le contrôle `Hidden` permet d'entrer dans le script des éléments qui n'apparaîtront pas à l'écran. Ces éléments sont donc cachés.

```
<INPUT TYPE="hidden" NAME="nom" VALUE "les données cachées">
```

V.9.j) L'envoi d'un formulaire par Email. (Netscape uniquement ?)



Soit :

```
<FORM METHOD="post" ACTION="mailto:votre_adresse_Email">
  <INPUT TYPE=text NAME="nom">
  <TEXTAREA NAME="adresse" ROWS=2 COLS=35>
  </TEXTAREA>
  <INPUT TYPE=submit VALUE="Submit">
</FORM>
```

Vous recevrez dans notre boite de réception, un truc bizarre du genre :

```
nom=Antony&adresse=Lycée+Douanier+Rousseau17OD%OA
```

où on retrouve les champs `nom=` et `adresse=`, où les champs sont séparés par le signe `&`, où les espaces sont remplacés par le signe `+` et `17%OD%OA` correspond à un passage à la ligne.

V.9.k) Compléments

Les boites de dialogue ou de message

Javascript met à votre disposition 3 boites de message :

- alert()
  - `alert(variable);`
  - `alert("chaîne de caractères");`
  - `alert(variable + "chaîne de caractères");`
- prompt()
  - `prompt("texte de la boite d'invite","valeur par défaut");`
- confirm()
  - `confirm("Voulez-vous continuer ?")`

Une minuterie

Javascript met à votre disposition une minuterie (ou plus précisément un compteur à rebours) qui permettra de déclencher une fonction après un laps de temps déterminé.

```
nom_du_compteur = setTimeout("fonction_appelée()", temps en milliseconde)
```

Pour arrêter le temporisateur avant l'expiration du délai fixé, il y a :

```
clearTimeout(nom_du_compteur)
```

L'emploi de this

Pour désigner l'objet en cours, Javascript met à votre disposition le mot-clé `this`.
Cette écriture raccourcie est souvent utilisée en remplacement du chemin complet de l'objet dans un formulaire.

Exemple :

```
<FORM NAME="form3">
<INPUT TYPE="radio" NAME="choix" VALUE="1">Choix numéro 1<BR>
<INPUT TYPE="radio" NAME="choix" VALUE="2">Choix numéro 2<BR>
<INPUT TYPE="radio" NAME="choix" VALUE="3">Choix numéro 3<BR>
<INPUT TYPE="button"NAME="but" VALUE="Quel et votre choix ?" onClick="choixprop(form3)">
</FORM>
```

Au lieu d'employer `choixprop(form3)`, on aurait pu utiliser `choixprop(this.form).`

Les messages d'erreur

Il y a 3 grandes catégories d'erreurs dans l'utilisation d'un programme Javascript :

- les erreurs au chargement
  Au chargement du script par le browser, Javascript passe en revue les différentes erreurs qui peuvent empêcher le bon déroulement de celui-ci.
  Les erreurs au chargement, nombreuses lors de l'apprentissage de Javascript, sont souvent dues à des fautes de frappe et/ou des erreurs de syntaxe.

- les erreurs à l'exécution
  Ici votre script se charge sans problème, mais une boite de message d'erreurs apparaît lorsque l'exécution du script est demandée. Alors que les erreurs au chargement étaient surtout dues au mauvais usage de la syntaxe, les erreurs à l'exécution proviennent d'un mauvais usage des commandes ou des objets Javascript.

- les erreurs de logique
  Ce sont les plus vicieuses car le "débuggeur" de Javascript ne signale bien entendu aucune erreur et votre script se déroule correctement. Hélas, à l'arrivée, le résultat ne correspond pas à celui espéré.

Transmettre des variables d'une page à l'autre

Vos variables sont définies dans le script de l'entité que constitue votre page Web. Vous pouvez souhaiter continuer à utiliser ces variables dans une autre page ou tout au long de votre site.
La solution est d'utiliser des frames. Javascript permet de passer des variables d'un frame vers des objets appartenant à un autre frame. Et comme tous les browsers Javascript admettent des frames, pourquoi s'en priver (voir un des chapitres suivant).

V.10- L'objet window

V.10.a) Utilisation de la barre d'état

Avec Javascript, la barre d'état (petite bande située au bas de la fenêtre du browser et qui vous informe sur l'état des transferts et des connections) peut être utilisée pour afficher des messages.

Propriété	Description
`status`	valeur du texte affiché dans la barre d'état de la fenêtre.
`DefaultStatus`	valeur par défaut qui s'affiche dans la barre d'état.

Exemple :

```
<HTML> <BODY>
  <A HREF="#" onmouseover="self.status='Votre texte'; return true;"> A voir ici </A>
```

```
</BODY> </HTML>
```

Commentaires :

- `self` : renvoie à la fenêtre en cours.
- Il est indispensable d'ajouter `return true`;

V.10.b) Ouverture et fermeture de fenêtres (théorie)

Méthodes	Description
`open()`	ouvre une nouvelle fenêtre.
`close()`	ferme la fenêtre en cours.

La syntaxe est :

```
[window.]open("URL","nom_de_la_fenêtre","caractéristiques_de_la fenêtre")
```

Commentaires :

- URL : URL de la page que l'on désire afficher dans la nouvelle fenêtre. (Sinon " ").
- caractérstiques_de_la _fenêtre : liste de caractéristiques, séparées par des virgules et sans espaces ni passage à la ligne :

Caractéristique	Description
`toolbar=yes ou no`	Affichage de la barre d'outils
`location=yes ou non`	Affichage de champ d'adresse (ou de localisation)
`directories=yes ou no`	Affichage des boutons d'accès rapide
`status=yes ou no`	Affichage de la barre d'état
`menubar=yes ou no`	Affichage de la barre de menus
`scrollbars=yes ou no`	Affichage des barres de défilement.
`resizable=yes ou no`	Dimensions de la fenêtre modifiables
`width=x en pixels`	Largeur de la fenêtre en pixels
`height=y en pixels`	Hauteur de la fenêtre en pixels

- On peut aussi utiliser 1 ou 0 au lieu de yes ou no.
- Cette nouvelle fenêtre va s'afficher un peu n'importe où sur votre écran.
- Attention : l'ouverture de plusieurs dizaines de fenêtres, ralentit le système et peut finir par le planter. Veillez donc à toujours faire fermer ces nouvelles fenêtres.

Exemple : Ouverture par un bouton (avec code dans le onClick)

`Page1.htm` : page de départ :

```
<FORM>
<INPUT TYPE ="button" value="Ouvrir une nouvelle fenêtre" onClick="open('test.htm', 'new',
'width=300,height=150,toolbar=no,location=no,directories=no,menubar=no,scrollbars=no)">
                                  // sans espaces ni passage à la ligne
</FORM>
```

`test.htm` : petite fenêtre qui affiche « Cecei est un test », avec un bouton dans la page :

```
<HTML> <BODY>
  <H1>Ceci est un test<H1>
  <FORM>
    <INPUT TYPE="button" value= " Continuer " onClick="self.close()">
  </FORM>
</BODY> </HTML>
```

où self.close() fermera la fenêtre courante, c.-à-d. la nouvelle fenêtre.

Exemple : Ouverture par un bouton (avec appel d'une fonction)

`Page1.htm` : page de départ :

```
<SCRIPT LANGUAGE="javascript">
<!--
  function new_window() {
    xyz="open('test.htm', 'new', 'width=300,height=150,toolbar=no,location=no,
                                  directories=no,status=no,menubar=no,scrollbars=no,resizable=no')">
  } // -->
</SCRIPT>
<FORM>
```

```
  <INPUT TYPE ="button" value="Ouvrir une nouvelle fenêtre" onClick="new_window()">
</FORM>
```

Exemple : Fermeture automatique après x secondes

Sans intervention de l'utilisateur, la nouvelle fenêtre se ferme de façon automatique après 4 secondes. En cliquant sur le bouton, l'utilisateur interrompt prématurément le compteur et ferme la fenêtre. Avec ce système, on est certain que le nouvelle fenêtre sera fermée.

`Page1.htm` : page de départ :

```
<FORM>
<INPUT TYPE ="button" value="Ouvrir une nouvelle fenêtre" onClick="open('testc.htm', 'new',
'width=300,height=150,toolbar=no,location=no,directories=no,menubar=no,scrollbars=no')">
</FORM>
```

```
testc.htm
<HTML>
<BODY onLoad='compt=setTimeout("self.close();",4000)'>
  <H1>Ceci est un test</H1>
  <FORM>
    <INPUT TYPE="button" value=" Continuer " onClick="clearTimeout(compt);self.close();">
  </FORM>
</BODY> </HTML>
```

Exemple : Ouverture en cliquant sur un lien ou une image

On ajoute simplement le "`onClick=open...`" à la balise `<A>` du lien ou de l'image.

`Page1.htm` : page de départ :

```
<A HREF="#" onClick="open('testc.htm', '', 'width=300,height=150')"> lien de test </A>
```

Exemple : Ecrire dans la nouvelle fenêtre

On passe par l'ouverture d'un nouvelle fenêtre par l'appel d'une fonction.

`Page1.htm` : page de départ :

```
<SCRIPT LANGUAGE="Javascript">
<!--
  function opnw(){
    msg=window.open("","","width=300,height=50");
    msg.document.write('<HTML> <BODY>' + '<CENTER><H1>Ceci est un test<H1></CENTER>' +
                                                                          '</BODY></HTML>');
  } // -->
</SCRIPT>
```

V.11- L'objet String

Revenons à l'objet `String` pour nous intéresser à la manipulation des caractères.

Instruction	Description
`length`	C'est un entier qui indique la longueur de la chaîne de caractères.
`charAt(x)`	Méthode qui permet d'accéder à un caractère isolé d'une chaîne.
`indexOf(x)`	Méthode qui renvoie la position d'une chaîne partielle à partir d'une position déterminée. (en commençant au début de la chaîne principale soit en position 0).
`LastIndexOf(x)`	Méthode qui renvoie la position d'une chaîne partielle à partir d'une position déterminée. (en commençant à la fin soit en position length moins 1).
`substring(x,y)`	Méthode qui renvoie un string partiel situé entre l position x et la position y-1.
`toLowerCase()`	Transforme toutes les lettres en minuscules.
`toUpperCase()`	Transforme toutes les lettres en Majuscules.

Propriété length

La propriété length retourne un entier qui indique le nombre d'éléments dans une chaîne de caractères. Si la chaîne est vide (" "), le nombre est zéro.

```
x=variable.length;
```

La propriété length sert aussi pour connaître la longueur ou le nombre d'éléments :

- de formulaires. Combien a-t-il de formulaires différents ?
- de boutons radio. Combien a-t-il de boutons radio dans un groupe ?
- de cases à cocher. Combien a-t-il de cases à cocher dans un groupe ?

- d'options. Combien a-t-il d'options dans un Select ?
- de frames. Combien a-t-il de frames "enfants" ?
- d'ancres, de liens, etc.

La méthode CharAt()

Il faut d'abord bien noter que les caractères sont comptés de gauche à droite et que la position du premier caractère est 0. La position du dernier caractère est donc la longueur (length) de la chaîne de caractère moins 1.

```
chaîne :        Javascript (longueur = 10)
                ||||||||||
position :      0123456789 (longueur - 1)
```

La syntaxe de `charAt()` est :

```
chaîne_réponse = chaîme_départ.charAt(x);
```

La méthode indexOf()

Cette méthode renvoie la position, soit x, d'un string partiel (lettre unique, groupe de lettres ou mot) dans une chaîne de caractères en commençant à la position indiquée par y (-1 sinon). Cela vous permet, par exemple, de voir si une lettre, un groupe de lettres ou un mot existe dans une phrase.

Exemple :

```
variable="antony&ldr.fr"
var="@"
x=variable.indexOf(var);
x vaut -1
```

La méthode lastIndexOf()

Méthode fort semblable à `indexOf(),` sauf que la recherche va cette fois de droite à gauche (en commençant donc par la fin).

Notons que même lorsqu'on commence à lire de la fin de la chaîne, la position retournée est comptée depuis le début de la chaîne avec le comptage commençant à zéro.

17.6 La méthode substring()

La méthode `substring()` est du même genre que `indexOf(), lastIndexOf()` et `charAt()`. Elle permet de d'extraire une partie de chaîne d'une longue chaîne de caractères.

Exemple :

```
Javascript
||||||||||
0123456789

str="Javascript";
str1=str.substring(0,4);
str2="Javascript".substring(0,4);
```

Les résultats sont :

```
str1="Java"; soit les positions 0,1,2 et 3.
str2="Java"; soit les positions 0,1,2 et 3.
```

La méthode toLowerCase()

Cette méthode permet de passer une chaîne de caractère en minuscules.

Exemple :

```
str="JavaScript";
str1=str.toLowerCase();
```

Le résultat sera :

```
str1="javascript";
```

La méthode toUpperCase()

Cette méthode permet de passer une chaîne de caractère en majuscules.

Exemple :
```
str="JavaScript";
str2=str.toUpperCase();
```

Le résultat sera :
```
str2="JAVASCRIPT";
```

Commentaire :
Ces deux méthodes sont très utiles si on considère que Javascript est case sensitive.

V.13- L'objet Date

Cette méthode renvoie toutes les informations "date et heure" de l'ordinateur de l'utilisateur.
```
variable=new Date();
```

Soit :
```
"Fri Dec 17 09:23:30 1998"
```

La date et l'heure dans Javascript commence au 1e janvier 1970. La méthode new date () sans arguments renvoie la date et l'heure courante. Pour introduire une date et une heure déterminée, cela se fera sous la forme suivante :
```
variable=new Date("Jan 1, 2000 00:00:00");
```

Méthodes de l'objet `Date` :

`getYear()`	Retourne les deux derniers chiffres de l'année dans variable_date.	`variable_date=new Date();` `an="20"+variable_date.getYear();`
`getMonth()`	Retourne le mois dans variable_date sous forme d'un entier compris entre 0 et 11.	`variable_date=new Date();` `mois=variable_date.getMonth();`
`getDate();`	Retourne le jour du mois dans variable_date sous forme d'un entier compris entre 1 et 31.	`variable_date=new Date();` `jourm=variable_date.getDate();`
`getDay();`	Retourne le jour de la semaine dans variable_date sous forme d'un entier compris entre 0 et 6.	`variable_date=new Date();` `jours=variable_date.getDay();`
`getHours();`	Retourne l'heure dans variable_date sous forme d'un entier compris entre 0 et 23.	`variable_date=new Date();` `hrs=variable_date.getHours();`
`getMinutes();`	Retourne les minutes dans variable_date sous forme d'un entier compris entre 0 et 59.	`variable_date=new Date();` `min=variable_date.getMinutes();`
`getSeconds();`	Retourne les secondes dans variable_date sous forme d'un entier compris entre 0 et 59.	`variable_date=new Date();` `sec=variable_date.getSeconds();`

Exemple : Script qui donne l'heure.
```
function getDt(){
  dt = new Date();
  cal = "" + dt.getDate() + "/"+ (dt.getMonth() + 1) + "/19" + dt1.getYear();
  hrs = dt.getHours();
  min = dt.getMinutes();
  sec = dt.getSeconds();
  tm = " " + ((hrs<10)?"0":"") + hrs + ":";
  tm += ((min<10)?"0":"") + min + ":";
  tm += ((sec<10)?"0":"") + sec + " ";
  document.write(cal+tm);
}
```

Si vous souhaitez que votre affichage de l'heure change toutes les secondes, ajouter :
```
setTimeOut("getDt()",1000);
```

Et aussi :
```
variable.getTime();          variable.getTimezoneOffset();      variable.setYear(x);
variable.setMonth(x);        variable.setDate(x);               variable.setHours(x);
variable.setMinutes(x);      variable.setSeconds(x);            variable.setTime(x);
variable.toGMTString();      variable.toLocaleString()
```

V.15- L'objet Array

L'objet `Array` (ou tableaux) est une liste d'éléments indexés dans lesquels on pourra ranger `(écrire)` des données ou aller reprendre ces données `(lire)`.

V.15.a) Tableau à une dimension

Déclaration

```
nom_du_tableau = new Array (x);
```
// où x est le nombre d'élément du tableau.

Initialisation et Accès

```
nom_du_tableau[i] = "elément";
```
// où i est un nombre compris entre 0 et x-1.

Exemple : un carnet d'adresse avec 3 personnes.

```
carnet = new Array(3);
carnet[0]="Garnier";
carnet[1]="Roblot";
carnet[2]="Igounet";
```

Commentaires :

- Il est conseillé de charger les tableaux avec des boucles
- Il n'est pas nécessaire de déclarer le nombre d'élément du tableau. Javascript prendra comme nombre d'éléments, le nombre i le plus élevé lors de "l'initialisation" (en fait i + 1).

Propriétés et méthodes

Elément	Description
`length`	Retourne le nombre d'éléments du tableau.
`join()`	Regroupe tous les éléments du tableau dans une seule chaîne. Les différents éléments sont séparés par une caractère séparateur spécifié en argument. Par défaut, ce séparateur est une virgule.
`reverse()`	Inverse l'ordre des éléments (ne les trie pas).
`sort()`	Retourne les éléments par ordre alphabétique (à condition qu'ils soient de même nature)

Dans le cadre de notre exemple :

```
document.write(carnet.reverse().join("-"));
```

donne : `Igounet-Roblot-Garnier`

V.15.b) Tableau à deux dimensions

On peut créer des tableaux à deux dimensions (et plus encore). Il suffit de déclarer un tableau dans un tableau :

On déclare d'abord un tableau à 1 dimension :

```
nom_du_tableau = new Array (x);
```

Ensuite, on déclare chaque élément du tableau comme un tableau à 1 dimension :

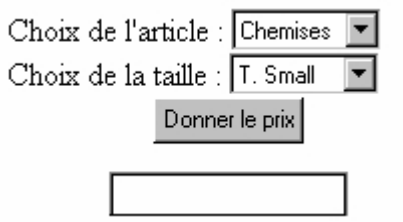
```
nom_du_tableau[i] = new Array(y);
```

Exemple : Pour un tableau de 3 sur 3 :

Tarif	*T. Small*	*T. Médium*	*T. Large*
Chemises	1200	1250	1300
Polos	800	850	900
T-shirts	500	520	540

```
nom = new Array(3);
nom[0] = new Array(3);
nom[1] = new Array(3);
nom[2] = new Array(3);
nom[0][0]="1200"; nom[0][1]="1250"; nom[0][2]="1300";
nom[1][0]="800"; nom[1][1]="850"; nom[1][2]="900";
nom[2][0]="500"; nom[2][1]="520"; nom[2][2]="540";
```

Pour exploiter ces données, voici une illustration de ce qui est possible :



Le formulaire s'écrit :

```
<FORM name="form" >
  <SELECT NAME="liste">
    <OPTION>Chemises
    <OPTION>Polos
    <OPTION>T-shirts
  </SELECT>
  <SELECT NAME="taille">
    <OPTION>T. Small
    <OPTION>T. Médium
    <OPTION>T. Large
  </SELECT>
  <INPUT TYPE="button" VALUE="Donner le prix" onClick="affi(this.form)">
  <INPUT TYPE="TEXT" NAME="txt">
</FORM>

function affi() {
  i = document.form.liste.selectedIndex;
  j = document.form.taille.selectedIndex
  document.form.txt.value=nom[i][j];
}
```

V.17- Les objets (notions de base)

V.17.a) Définition

Définir ces propres types structurés peut se faire à l'aide d'objets :

```
var nom_objet = new Object;
```

Exemple : un produit est composé d'une référence et d'une désignation.

```
<html><body>
<script>
  var produit = new Object;
  produit.ref = "p01";
  produit.des = "Chemise";
  alert(produit.ref + ":" + produit.des);
</script>
</body></html>
```

Exemple : un tableau de produit (un peu laborieux).

```
<html> <body>
<script>
  var i;
  var tabProduit = new Array(3);
  tabProduit[0] = new Object;
  tabProduit[0].ref = "p01";
  tabProduit[0].des = "Chemise";
  tabProduit[1] = new Object;
  tabProduit[1].ref = "p02";
  tabProduit[1].des = "Pantalon";
  tabProduit[2] = new Object;
  tabProduit[2].ref = "p03";
  tabProduit[2].des = "Veste";
  for (i=0;i<3;i++)
    alert(tabProduit[i].ref + ":" + tabProduit[i].des);
</script>
</body> </html>
```

Exemple : un tableau de produit initialisé par une fonction.

```
<html> <head>
<script>
  function produit(reference,designation) {
    this.ref=reference;
    this.des=designation;
  }
</script>
</head>
<body>
  <script>
    var pdt1 = new produit('p01','Chemise');
    var pdt2 = new produit('p02','Pantalon');
    alert(pdt1.ref + ":" + pdt1.des);
    alert(pdt2.ref + ":" + pdt2.des);
    </script>
</form> </body> </html>
```

Commentaires:

- L'affectation est le passage en paramètre d'objet se fait toujours par référence (en lecture / écriture : on peut donc les modifier).
- La fonction `produit` s'appelle un constructeur.
- Le mot-clé "`this`" désigne l'objet en cours.

V.17.b) Rapport entre les objets et les tableaux

Tableau associatif

Il est possible d'indicer un tableau à l'aide d'une chaîne de caractères au lieu d'un entier.

```
<html> <body>
<script>
  var tabPdt=new Array();
  tabPdt["p01"]="Chemise";
  tabPdt["p02"]="Pantalon";
  alert(tabPdt["p01"]);
  alert(tabPdt["p02"]);
</script>
</form> </body> </html>
```

Un tel tableau s'appelle un tableau associatif, car il permet d'associer un ensemble de valeurs (les désignations) à un ensemble de clés (les références). C'est en fait un objet qui possède les propriétés correspondant aux valeurs des clés. On aurait donc pu écrire :

```
alert(tabPdt.p01);
alert(tabPdt.p02);
```

Un objet vu comme un tableau

A l'inverse, un objet peut être utilisé avec la syntaxe réservée aux tableaux associatifs :

```
<html> <head>
<script> function produit(reference,designation) { … } </script>
</head>
<body>
  <script>
    var pdt1 = new produit('p01','Chemise');
    alert(pdt1["ref"]);
    alert(pdt1["des"]);
</script>
</form> </body> </html>
```

Le nom des propriétés est utilisé comme clé.

V.18- Techniques courantes

V.18.a) Contrôle de saisie : Test de numéricité

La fonction `isNaN(val)` retourne `true` si val n'est pas un nombre (`Not A Number`). Mais attention : si val est "`undefined`" (champ non renseigné) la fonction retourne `false`...

Deux autres fonctions permettent de tester la numéricité d'une donnée saisie (les champs de saisie sont toujours de type "text") :

- `parseInt(chaîne val)` retourne un entier correspondant à val ou la valeur `NaN`.
- `parseFloat(chaîne val)` retourne un réel correspondant à val ou la valeur `NaN`.

Exemple :

```
if isNaN(parseInt(val)) alert("erreur !");
```

Attention :

`parseInt("52.6")` retourne 52

`parseInt("48 heures plus tard")` retourne 48.

Pour tester correctement la numéricité d'une entrée saisie :

```
if (parseInt(val) == val)
  alert("La saisie est correcte, il s'agit bien d'un entier");
else
  alert("La saisie est incorrecte, il ne s'agit pas d'un entier");
```

V.18.b) Contrôle de saisie : Validation du formulaire

Les contrôles de saisie servent à valider le contenu d'un formulaire avant de le soumettre au serveur (il est inutile de gaspiller de la bande passante et le temps du serveur en envoyant des données manifestement erronées). Plusieurs posiblités existent, voici les principales :

Contrôler le contenu de chaque champ immédiatement après sa saisie grâce à l'événement **"onChange"**

Exemple de contrôle "à la volée" de deux champs de formulaire :

```
<html> <head>
<script>
  function verif(champ, type) {
    var controle;
    if (type == "entier")
      contrôle = parseInt(champ.value);
    else
      controle = parseFloat(champ.value);
    if (controle != champ.value) {
      alert("Saisie incorrecte !");
     champ.focus();
   }
  }
</script>
<body>
  <form name="saisie">
    Entrez une valeur entière
    <input type="text" size="10" name="unEntier" onChange="verif(this,'entier');"><p>
    Entrez une valeur réelle
    <input type="text" size="10" name="unReel" onChange="verif(this,'reel');"><p>
    <input type="submit" value="envoyer">
    <input type="reset" value="annuler">
  </form>
</body> </html>
```

Commentaires :

- Le premier paramètre de la fonction "`verif`" permet de l'utiliser pour plusieurs champs. Ce type de fonction pourrait trouver sa place dans un fichier de scripts (`.js`).
- L'inconvénient de cette première méthode est que l'utilisateur peut ne saisir aucune valeur. La seconde méthode permet de résoudre ce problème.

Contrôler l'ensemble des champs du formulaire lors de son envoi à l'aide de l'événement **"onSubmit"**

Exemple : Le contrôle est fait en une seule fois.

```
<html><head>
<script>
  function verifSaisie(monForm) {
    var valeur, controle, ok;
    ok=true;
    valeur = monForm.unEntier.value;
    controle = parseInt(valeur);
    if (controle != valeur) {
      alert("Saisie incorrecte !");
      monForm.unEntier.focus();
      ok=false;
    } else {
      valeur = monForm.unReel.value;
      controle=parseFloat(valeur);
      if (controle != valeur) {
        alert("Saisie incorrecte !");
        monForm.unReel.focus();
        ok=false;
      }
    }
    return ok;
  }
</script>
<body>
  <form name="saisie" onSubmit="return verifSaisie(this);">
    Entrez une valeur entière
    <input type="text" size="10" name="unEntier"><p>
    Entrez une valeur réelle
    <input type="text" size="10" name="unReel"><p>
    <input type="submit" value="envoyer">
    <input type="reset" value="annuler">
  </form>
</body> </html>
```

Commentaires :

- Le paramètre de la fonction permet de simplifier l'écriture lors de l'accès aux champs du formulaire. Il faudrait sinon écrire "`valeur = document.forms.saisie.unEntier.value`".
- Attention à l'événement `onSubmit` : on doit écrire "`return maFonction(...);`" et maFonction doit retourner un booléen.

V.20- Les Applets

La vocation d'une `applet` est d'être téléchargée sur une machine donnée à partir d'une machine distante qui en fournit le code.

Un fichier HTML qui lance une applet contiendra une commande particulière fournissant au minimum les informations suivantes :

Nom de paramètre	Infos devant l'accompagner
`CODE=`	Nom du fichier contenant l'applet. Ex : CODE=PremApp.class
`WIDTH=`	Largeur initiale de la fenêtre consacrée à l'applet. Ex : WIDTH=350
`HEIGHT=`	Hauteur initiale de la fenêtre consacrée à l'applet. Ex : HEIGHT=350

Exemple :

```
<HTML> <HEAD> </HEAD>
<BODY> <P>
  <applet code=PremApp.class width=320 height=200> </applet>
</BODY> </HTML>
```

Exemple de l'applet `Watermessage` :

1. Récupérer le fichier `watermessager.class` et le placer à côté de votre fichier `test.htm`

2. Créer votre fichier : `test.htm`

```
<HTML>
<HEAD>
   <TITLE> Applet : WaterMessage</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<CENTER><B>Les messages peuvent sortir de l'eau...</B></CENTER>

<CENTER>
<APPLET code="watermessager.class" align="baseline" width="500" height="150">
  <PARAM name="demicron" value="www.demicron.se">
  <PARAM name="reg" value="A00046">
  <PARAM name="foreground" value="255">
  <PARAM name="background" value="8080">
  <PARAM name="textdelay" value="500">
  <PARAM name="delay" value="40">
  <PARAM name="font" value="Signature">
  <PARAM name="fontsize" value="50">
  <PARAM name="amplitude" value="80">
  <PARAM name="item0" value="WaterMessager">
  <PARAM name="item1" value="Une applet">
  <PARAM name="item2" value="qui sort">
  <PARAM name="item3" value="de l'eau">
</APPLET>
</CENTER>
</BODY>
</HTML>
```

3. Ouvrer le fichier : `test.htm`

LES OUTILS

Ces premières notions nous permettent de construire facilement des sites statiques (site vitrine), mais moins facilement des sites catalogues ou des sites marchands digne de ce nom.

Nous allons donc introduire de nouveaux outils qui vont nous permettre de passer de site statique au site dynamique. De plus, nous nous servirons d'un éditeur Html, afin de se décharger d'un certain nombre de tâches fastidieuses comme les caractères spéciaux, les codes de couleur, l'encodage des balises, etc…

Attention, les éditeurs Html sont généralement WYSIWYG (What You See Is What You Get)et le codage des balises Html n'apparaît plus. Gardez cependant à l'esprit que cette prévisualisation n'est pas (et ne sera jamais) fidèle à 100% par rapport à ce qui sera affiché par le browser et qu'elle ne vous empêche nullement de consulter ou modifier le code source

I/ PRINCIPE DE FONCTIONNEMENT DES SITES WEB

I.1- Site vitrine avec script côté Client.

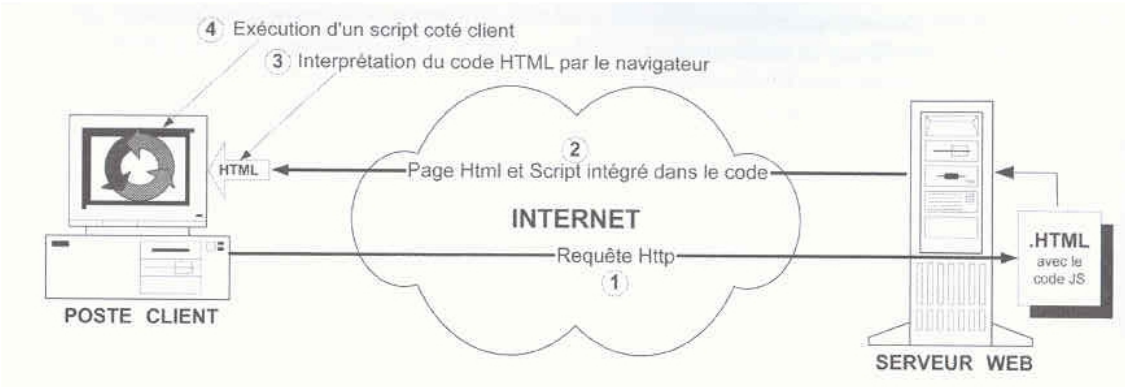


Fig : Utilisation d'un script côté Client.

Avantages :

- Facilité de mise en œuvre.
- Interactivité rapide (utilisation des ressrouces de la machine du client).

Inconvénients :

- Dépendance vis-à-vis du navigateur client (possibilité de non compréhension du code javascript ou des balises HTML)
- Maintenance des pages HTML longues et fastidieuses.

I.2- Site dynamique avec script côté Serveur

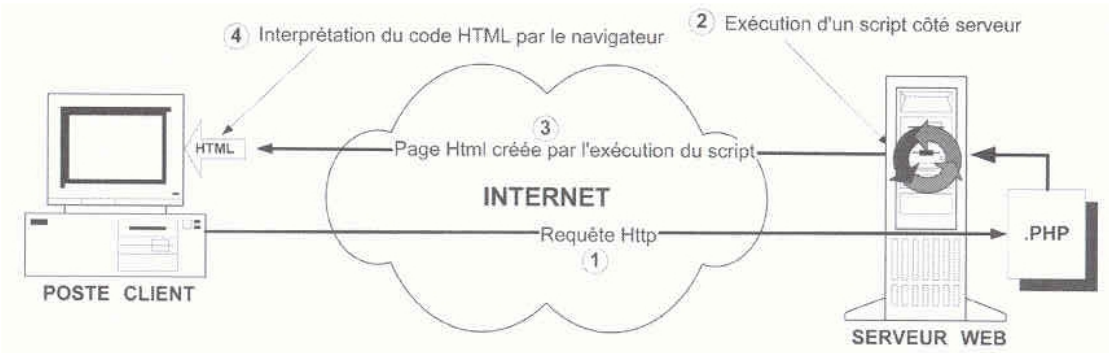


Fig : Utilisation d'un script côté Serveur

Avantages :

- Il n'y a pas de dépendance vis-à-vis du navigateur client.
- Le code source est préservé.

- La maintenance est plus facile.

Inconvénients :
- L'interactivité est plus lente.
- Le dévelopement est plus technique (nécessite des compétences).

I.3- Site dynamique avec base de données

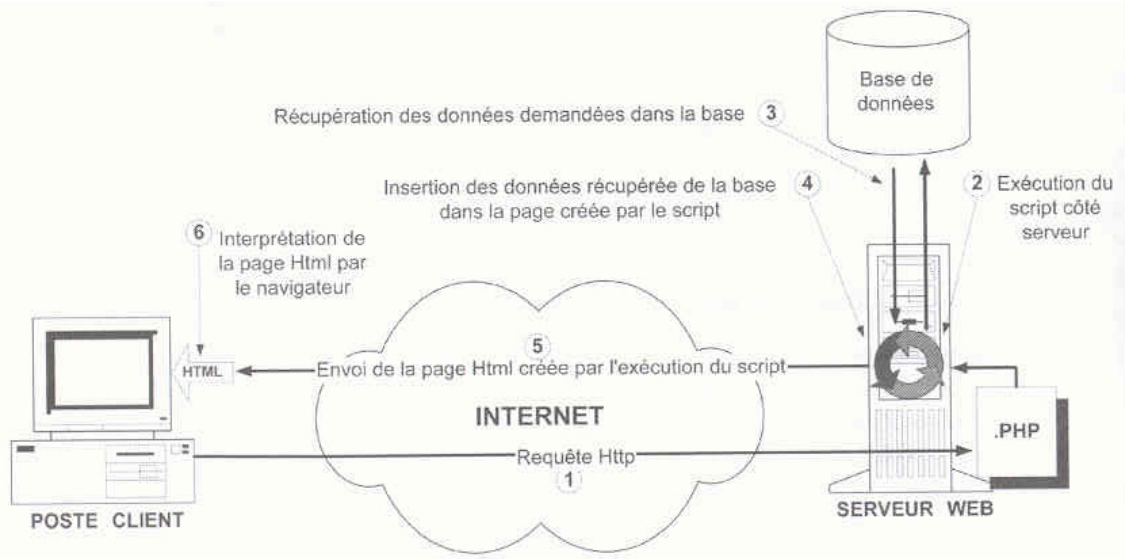


Fig : Utilisation d'une Base de données.

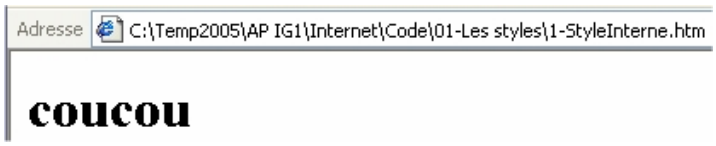
Architecture à trois niveaux (Client – Serveur – BDD) :
Le poste client envoie une requête `HTTP` sur un fichier dynamique (ici `PHP`). Le serveur Web localise le fichier et l'éxécute, ajoutant si nécessaire des informations issues de la base de données. La page ainsi créée est envoyée au client, où elle est interprétée comme une simple page `HTML` par le navigateur.

Avantages :
- Il n'y a pas de dépendance vis-à-vis du navigateur client.
- Le code source est préservé.
- La maintenance et l'évolution sont plus faciles.
- De nombreuses possibilités :
  - Mises à jour automatisées,
  - Maintenance aisée et assistée,
  - Sites multilangues,
  - Recherche multicritères,
  - Programmes divers.

Inconvénients :
- Architecture la plus lente.
- Architecture la plus difficile à mettre en œuvre (necessite des compétences)

I.4- Site Interprété – Site Compilé

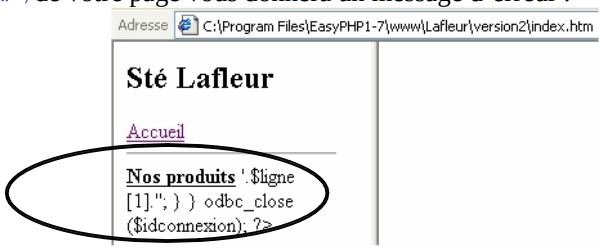
HTML est un langage de balise interprété par le navigateur. A ce titre, une page html peut donc être ouverte par un navigateur et fournir un résultat :



Il en va de même pour le Javascript qui est interprété et exécuté par le navigateur. Le chemin `C:\…` dans la barre d'adresse indique que nous avons ouvert directement le fichier.

Les langages Php, Asp, … ne permettent pas l'ouverture en directe de la page, car ce code doit être compilé, et éxécuté pour obtenir un résultat. C'est (comme l'explique les paragraphes ci-dessus) le résultat qui est retourné au navigateur du client. Ce résultat étant du HTML ou du Javascript, il peut donc être interprété par le Client.

Ainsi, l'appel direct (C:\ …) de votre page vous donnera un message d'erreur :



alors que l'appel de cette même page via le serveur Web (http://…), donnera :

Pour rappel, une requête http se décompose ainsi :

http://	www.MonSite	/MonRepertoire	/index.htm (ou asp / php)
Protocole	Nom du serveur	Nom de l'alias	Page de démarrage

Commentaires :

- Le protocole http:// reste le même puisqu'il s'agit d'appel de page Web.
- Nom du serveur est : localhost (ou 127.0.0.1 : adresse IP du serveur sur la machine locale) puisque c'est votre machine locale qui possède son propre serveur Web.
- Nom de l'alias est un alias pointant sur l'emplacement physique de votre site. C'est le serveur Web qui vous permet de définir des alias.

Remarque scolaire : Accès aux sites des voisins.
Chaque ordinateur possédant un serveur web, il devient hébergeur comme sur le net. La classe est donc un mini-web, puisque possédant 19 hébergeurs. Vous pouvez ainsi accéder aux différents sites de vos camarades en tapant l'addresse de ce dernier.

II/ INFRASTRUCURE

II.1- Infrastructure classique

La construction du site se fait sur le réseau local. Le site, une fois terminé, est publié (via FTP) sur un seveur Web de production (Serveur dédié de l'entreprise ou hébergeurs), où il pourra être consulté.

Le développement des pages du site se fait sur une machine différente de celle où se situe la base de données et les autres pages (html, php, …). Ce qui permet :

- de mieux simuler l'architecture 3 tiers.
- un travail plus rapide (plus de ressources machines).
- l'utilisation des parties du site non modifiées ou la base de données par d'autres utilisateurs.

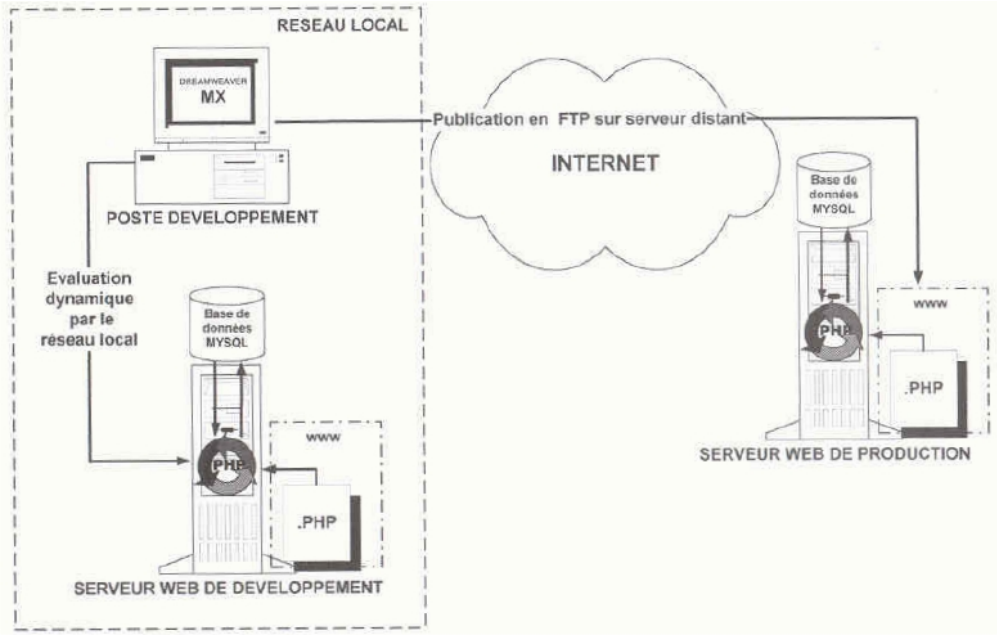
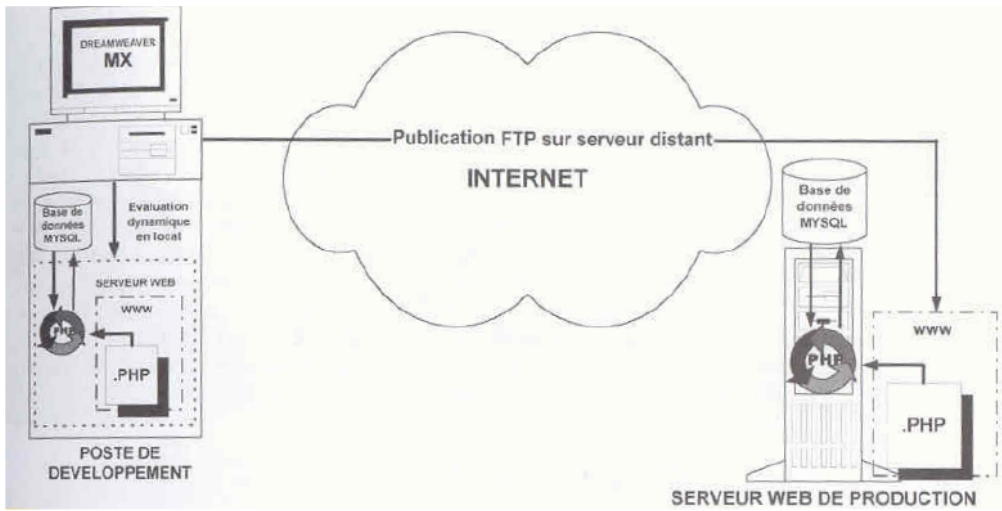


Fig : Infrastrucure serveur utilisant un serveur du réseau local pour les évaluations dynamiques.

II.2- Infrastructure scolaire

L'ensemble de la production du site se fait sur le poste de travail (ou poste de développement). Le site, une fois terminé, est publié `(via FTP)` sur un seveur Web de production (Serveur dédié de l'entreprise ou hébergeurs), où il pourra être consulté.



III/ LES CHOIX LOGICIELS

III.1- Les plates formes et le serveur Web

III.1.a) Microsoft

Plate forme	Microsoft Windows 9x, XP, 200X, etc …	
Serveur Web	IIS (Internet Information Service)	www.microsoft.fr

III.1.b) Monde Libre

Plate forme	Unix, Linux (avec Red Hat, Mandrake, Debian, …)
Serveur Web	Apache

Il existe des utilitaires fonctionnant sous Microsoft et permettant de simuler ces deux outils : EasyPhp (www.easyphp.org) et Wamp (www.wampserver.com).

III.2- Les bases de données

Avec les middlewares (médiateurs), l'ensemble des bases de données est accessible par chacun des langages d'Internet. Mais il existe bien évidemment des affinités entre langages et bases de données.

SGBD

Access	www.microsoft.com
SQL Server	www.microsoft.com
MySQL (via phpMyAdmin)	www.mysql.com
PostgreSQL	www.postgresql.org
hyperSonicSQL	www.hsqldb.org

III.3- Les langages

ASP

Développé par Microsoft, cette technologie s'appuie sur l'utilisation du langage VBScript (version allégée de Visual Basic) et de composants ActiveX (composant objets) pour les traitements applicatifs complexes.

La portabilité de l'ASP est plus restreinte, puisque l'architecture optimale est : plate-forme Microsoft (Windows), Serveur Microsoft (IIS), langage Microsoft (ASP), bases de données Microsoft (Access ou SqlServer).

ASP.net

Semblable au langage ASP. C'est une nouvelle technologie propriétaire développée par Microsoft, qui doit fonctionner sur un serveur .Net Web Server.

La diversité des types de client gérés par cette plate-forme et les possibilités (bibliothèques) « affichées » sont des atouts qui joueront sur sa pérénité.

JSP (Java Server Pages) : www.sun.com

Les JSP sont la réponse de SUN à Microsoft (ou plutôt l'inverse !) Cette technologie s'appuie sur l'utilisation du langage Java et des composants JavaBeans (composants objets).

Le gros avantage est la portabilité, mais vous devrez mettre à jour le serveur Web, afin de disposer des extensions capable de traiter les JSP. Parmi les serveurs http gratuits qui prennent en charge les JSP, citons JSWDK (proposé par SUN) ou Tomcat (proposé par la fondation Apache).

PHP : www.php.net

Langage de script côté serveur, simple, performant et qui s'interface avec de nombreuses bases de données (MySql, Informix, dBase, Oracle, Sybase, PostgreSql, HyperSonic, …).

CFML

ColdFusion Markup Language est une technologie (langage et logiciel du même nom) propriétaire de Allaire Corporation, intégrée dans la gamme de produis Macromédia.

III.4- Les éditeurs

Editeurs

WebExpert	www.visic.com
Front Page	www.microsoft.com
Netscape Editor	home.netscape.com
Claris Home	www.filemaker.com

Nous utiliserons :

Dreamweaver MX	www.macromedia.com
Notepad ++	notepad-plus.sourceforge.net/fr/site.htm

III.5- Plus

III.5.a) Domaine et hébergeurs

Domaine

Dépôts de noms de domaine : www.gandi.net
- .com
- .org
- .net

Dépôts de noms de domaine : www.2fr.fr
- .fr

Hébergeurs
PHP

www.nexenservices.com
www.free.fr
www.multimania.com
www.ovh.com
www.amen.fr

Hébergeurs
ASP

Hébergeurs
ASP.Net

III.5.b) Les autres sites utiles

Site

Dreamweaver MX et PHP	www.phpmx.com
ASP, PHP, …	www.asp-php.net
Cours PHP complet	www.manuelphp.com/cours/
CommentCamarche	www.commentcamarche.net
W3C	www.w3.org
Php	www.php.net

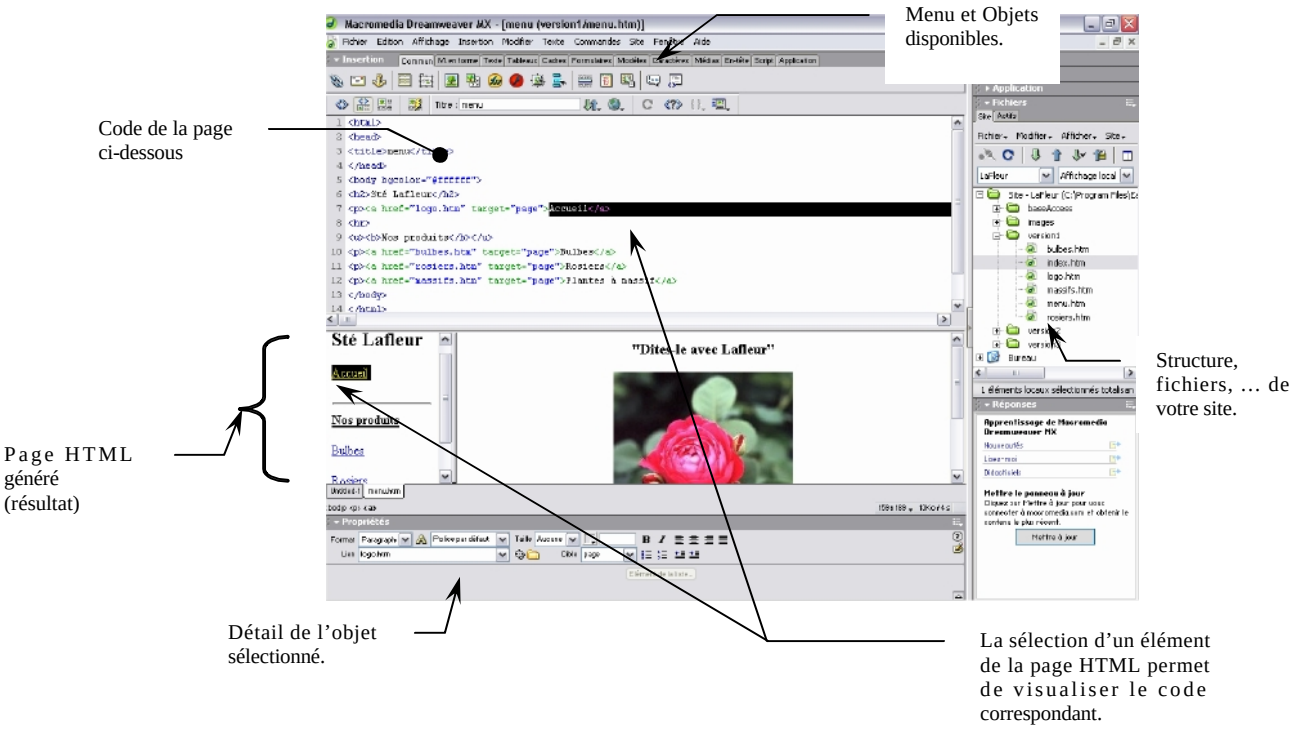
DREAMWEAVER

Dreamweaver est un logiciel de création de pages HTML d'un site Web en mode visuel. Il permet au concepteur de modifier directement le code des pages grâce à son éditeur de code intégré. Un troisième mode (appelé mixte) permet d'éditer le code tout en bénéficient de l'environnement visuel de mise en page.

Dreamweaver permet d'écrire des programmes Javascript en mode ligne de commande ou non. De plus, ce dernier intègre des fonctions de développement dynamique en utilisant les technologies (ASP, CFML, JSP), mais il permet aussi d'utiliser le couple PHP/MySql.

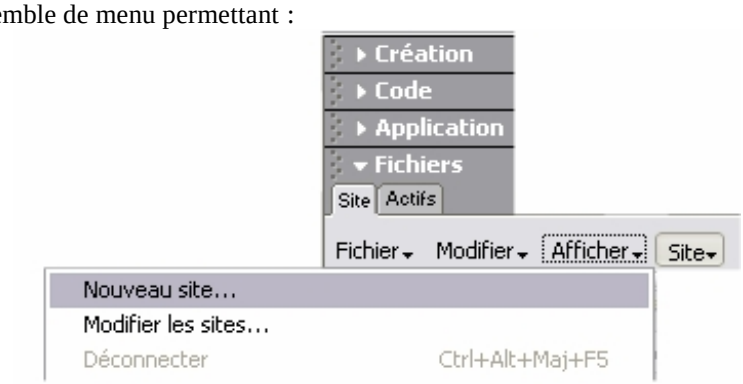
I/ INTERFACE GRAPHIQUE

Dreamweaver étant un outil très riche, nous ne vous donnerrons ici que quelques pistes permettant de vous lancer. Nous vous rappelons que tout code qui peut être généré par l'un des outils de Dreammweaver doit être compris et maîtrisé.



I.1- Gestion du site

En haut à droite, un ensemble de menu permettant :



- Création
  - ↳ Gestion des styles et des comportements.
- Code
  - ↳ Inspecteur de balise, ainsi que manuel de référence : aide en ligne.
- Application

- *Gestion des bases de données*. L'onglet base de données permet notament de sélectionner et voir la structure d'une base de données. Les onglets suivant permet de créer des liaisons, des connexions, des jeux d'enregistrements qui vous permettront de vous passer de code (automatiquement généré) pour accéder aux informations contenus dans une base de données.

- Fichiers
  - L'onglet Site permet notament de gérer l'ensemble des fichiers de votre site Web.

Travail :
Créer la structure de votre premier site :

- Donner lui un nom (ex : `laFleur`) et utilisez la technologie serveur `Php`,
- Placer le site dans le repertoire racine du serveur Web

(`ex : C:\Program Files\EasyPHP1-7\www\laFleur\`)

`EasyPhp joue ici le rôle de serveur Web, c'est donc lui qui nous donne le répertoire de base. Notons qu'il est aussi possible de créer un « Alias ».`

Vous pouvez maintenant travailler dans ce `panneau Fichiers`, comme si vous travailliez sous l'explorateur.

I.2- La barre d'outils standard et le panneau Insertion

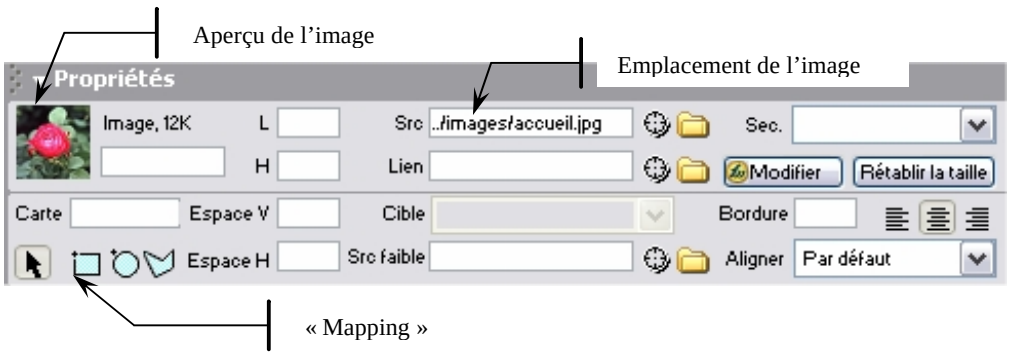
Cette barre d'outils regroupe des fonctions communes à de nombreux logiciels de l'environnement Windows. Elle permet de trouvez tous les objets ou notions liés à l'Internet. De nombreux raccourcis graphiques (icônes) permettent de supplanter les menus. Ainsi le `panneau Insertion` permet de retrouver toutes les balises étudiées dans les chapitres précedent, mais permet aussi d'utiliser plus facilement les scripts, les styles et encore les lanages (ex : php).

I.3- Le code et la Page HTML

La page `création` est au format WYSIWYG de la page `code`. Dans ce dernier mode, la touche `ctrl+espace` permet d'obtenir l'ensemble des fonctions liés à la commande en cours.

I.4- Le panneau propriétés

Le `panneau propriétés` est contextuel et affiche toutes les informations ou paramètres liés à l'objet sélectionné.
Exemple de sélection de d'une image :



EASYPHP

I/ EASYPHP

EasyPhp permet d'émuler sous Windows un serveur Apache et une base de données MySql.

Easyhp doit démarrer ses deux services afin de pouvoir les utiliser. Si EasyPhp est démaré, un  est présent dans la barre des tâches et vous permet d'accéder aux fonctionnalités :

Le mode `Administration` permet l'accès à :

Easyphp propose d'autres ouverture, telles ques :

- `PHPInfo` : qui regroupe l'ensemble informations sur le support.
- `FAQ` : une foire aux questions fournies, qui proposent de nombreux liens.

II/ APACHE

Apache est le serveur web sous linux et d'EasyPhp. Vous pouvez grâce à lui compilé et éxécuter des pages contenant du php, de l'asp ou autres.

Afin de ne pas avoir des adresses de sites trop longues et difficiles à retenir, le serveur web vous permet de créer des alias. Ces derniers permettront d'accéder directement à la première page de votre site, quelque soit sont emplacement physique. Attention, une fois l'alias créer vous ne devrez en aucun cas déplacer physiquement votre site, sous peine de ne plus pouvoir y accéder via l'alias.

Travail : Création d'un alias pour notre site.

1- Le site doit physiquement exister (dans un repertoire : `c:\monsite\`).
2- Cliquer sur ajouter, puis renseigner les champs :

3- Tester en cliquant sur l'alias créé, puis sur la page d'accueil :

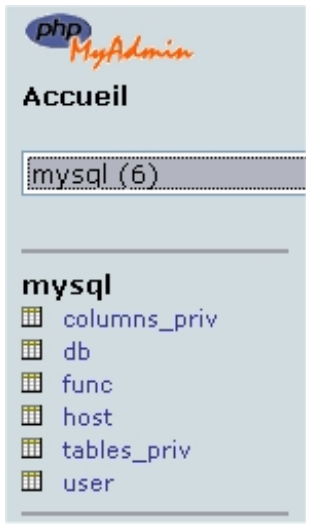
4- Vous obtenez :

5- En cliquant sur : `Affichage\Source` de votre navigateur, vous obtiendrez le code source de la page affichée. Remarquez qu'il n'y a aucune ligne de php !

III/ PHPMYADMIN

PhpMyAdmin, vous permet d'administrer graphiquement une base de données MySQL.

PhpMyAdmin posséde par défaut une base `mysql`, qui permet de gérer les droits d'accès et la sécurité.



Travail à faire :

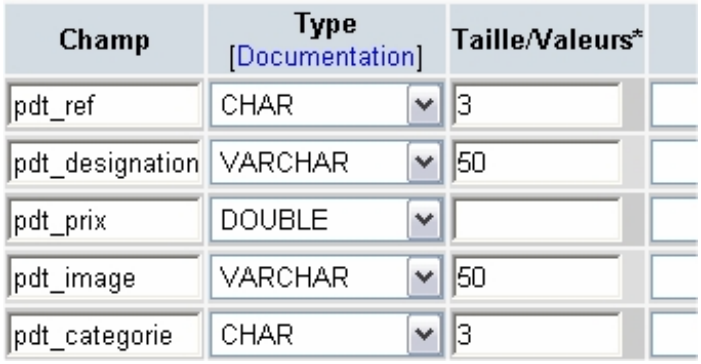
1. Créer votre propre base de données : ex : `lafleur`

Une fois créé, vous aurez les possibilités suivantes :

- `Structure` : permet la gestion de la structure de la base de données : LDD, LMD et LID.
- `SQL` : permet d'éxécuter des instructions SQL en direct.
- `Exporter` : permet d'exporter votre base de données dans un format spécifique. Choisissez de préférence un `.sql`, reconnaissable par tous les SGBD et choissiez bien vos options.
- `Rechercher et requete` : permettent de recherchee des informations dans votre base de données soit dans un format texte soit par sélection d'objets.
- `Supprimer` : permet de supprimer la base de données.

2. Dans l'onglet `Structure`, créer une table : `produit`

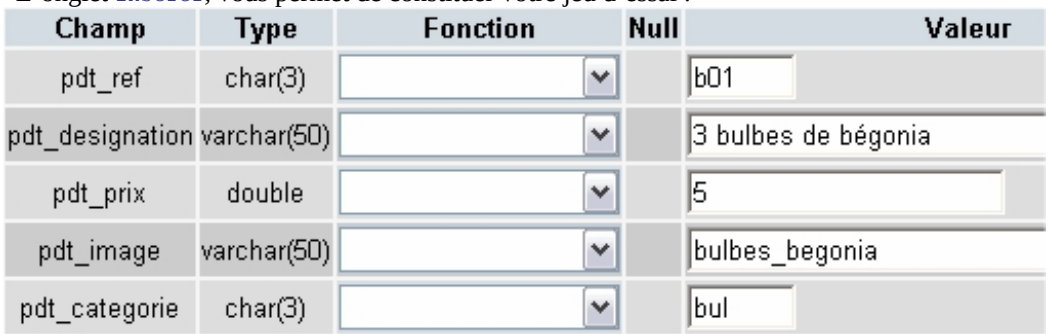
3. Puis renseigner les `champs`, ainsi que leurs `types` :



En cliquant sur la table, vous avez maintenant la possibilité de :

- `Structure` : permet la gestion de la structure de la table.
- `Afficher` : permet d'afficher le contenu de votre table.
- `SQL` : permet d'éxécuter des instructions SQL en direct.
- `Sélectionner` : permet de sélectionner des occurrences de manière graphique.
- `Insérer` : permet d'insérer un jeu d'essai de manière graphique.
- `Exporter` : permet d'exporter votre table dans un format spécifique. Choisissez de préférence un `.sql`, reconnaissable par tous les SGBD et choissiez bien vos options.
- `Opérations` : manipulation diverses sur la table.
- `Vider` : permet de vider la table.
- `Supprimer` : permet de supprimer la table.

4. L'onglet `insérer`, vous permet de constituer votre jeu d'essai :



5. et ainsi de suite, pour l'ensemble de votre base de données.

PHP

I/ INTRODUCTION

PHP (Hypertext Preprocessor) est un langage de scripts généraliste et Open Source, spécialement conçu pour le développement d'applications web. Il peut être intégré facilement au HTML.

Exemple :

```
<html> <head>
<title>Exemple</title>
</head>
<body>
  <?php
    echo "Bonjour, je suis un script PHP!";
  ?>
</body> </html>
```

I.1- Que peut faire PHP?

Il y a trois domaines différents où PHP peut s'illustrer :

- Langage de script coté serveur. Utilisation la plus traditionnelle, nécessitant trois composants : un analyseur PHP, un serveur web et un navigateur web.
- Langage de programmation. Ecrire des scripts PHP et les exécuter en ligne de commande, sans l'aide du serveur web et d'un navigateur.
- Ecrire des applications clientes graphiques. PHP n'est probablement pas le meilleur langage pour écrire des applications clientes graphiques, mais il dispose de fonctionnalités avancées à l'aide de PHP-GTK.

Avec PHP, vous n'êtes pas limités à la production de code HTML. Les capacités de PHP lui permettent de générer aussi bien des images, des fichiers PDF, des animations Flash (avec l'aide des librairies libswf et Ming), générés à la volée, du texte, du code XML ou XHTML.

De plus, une des grandes forces de PHP est le support de nombreuses bases de données : Ada, Dbase, IBM DB2, Informix, Ingres, MySQL, ODBC, Oracle, PostgreSQL, Sybase, …

I.2- Votre première page PHP

Créez un fichier appelé `bonjour.php` avec le contenu suivant :

```
<html> <head>
<title>Test PHP</title>
</head>
<body>
  <? php echo '<p>Bonjour le monde</p>'; ?>
</body> </html>
```

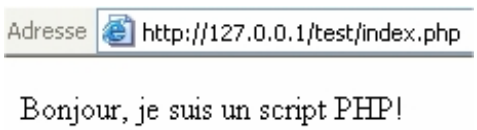
Utilisez votre navigateur pour accéder au fichier via votre serveur web (cf cours précédant) :

`http://localhost/<alias>/bonjour.php` ou encore `http://127.0.0.1/<alias>/bonjour.php`

Le code interprété par le navigateur (`Affichage/source`) sera :

```
<html> <head>
<title>Test PHP</title>
</head>
<body>
  <p>Bonjour le monde</p>
</body> </html>
```

et le résultat sera :



I.3- Deuxième exemple : avec des variables

Réalisons maintenant quelque chose de plus puissant. Nous allons vérifier le type de navigateur que le visiteur de notre site utilise.

Pour cela, nous allons accéder aux informations que le navigateur du visiteur nous envoie, lors de sa requête `HTTP`. Cette information est stockée dans une variable. Les variables sont faciles à repérer, car elles commencent toutes par un signe `$`. La variable qui nous interesse ici est `$_SERVER['HTTP_USER_AGENT']`.

Pour afficher cette variable, nous pouvons faire :

```
<? php echo $_SERVER['HTTP_USER_AGENT']; ?>
```

Un résultat possible du script pourra alors être :

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

Il y a de nombreux types de variables disponibles en PHP que nous verons par la suite. $_SERVER est juste une variable réservée qui est automatiquement disponible dans votre script.

Vous pouvez ajouter plusieurs commandes PHP dans une balise PHP, et créer de petits blocs de code qui réalisent des opérations plus complexes qu'un simple affichage. Par exemple, si nous voulons vérifier que le navigateur est bien de la famille des Internet Explorer, nous pouvons faire cela :

```
<?php
  if (strpos($_SERVER['HTTP_USER_AGENT'], 'MSIE') !== FALSE) {
    echo 'Vous utilisez Internet Explorer<br />';
  }
?>
```

Le résultat de ce script, si vous utilisez Internet Explorer, sera :

```
Vous utilisez Internet Explorer<br />
```

Ici, nous introduisons de l'algorithmique (`la structure if`), et des méthodes de php (`la fonction strpos`).

Nous pouvons maintenant progresser et vous montrer comment utiliser le mode PHP, au milieu du code HTML :

```
<?php
if (strpos($_SERVER['HTTP_USER_AGENT'], 'MSIE') !== FALSE) {
?>
  <h3>strpos() n'a pas retourné FALSE</h3>
  <p>Vous utilisez Internet Explorer</p>
<?php
} else {
?>
  <h3>strpos() a retourné FALSE</h3>
  <p>Vous n'utilisez pas Internet Explorer</p>
<?php
}
?>
```

Le résultat obtenu de ce script est :

```
<h3>strpos() n'a pas retourné FALSE</h3>
<p>Vous utilisez Internet Explorer</p>
```

I.4- Troisième exemple : avec un formulaire

L'un des points forts de PHP est sa capacité à gérer les formulaires. Il est important à comprendre est que tous les champs d'un formulaire seront automatiquement disponibles dans le script PHP d'action.

Un simple formulaire HTML :

```
<form action="action.php" method="post">
  <p>Votre nom : <input type="text" name="nom" /></p>
  <p>Votre age : <input type="text" name="age" /></p>
  <p><input type="submit" value="OK"></p>
</form>
```

Lorsque le visiteur remplit le formulaire, et clique sur le bouton `OK`, le fichier `action.php` est appelé :

```
Bonjour, <?php echo $_POST['nom']; ?>.
Tu as <?php echo $_POST['age']; ?> ans.
```

Voici le résultat que vous pourriez obtenir :

```
Bonjour Jean.
Tu as 29 ans.
```

Les variables $_POST['nom'] et $_POST['age'] sont automatiquement créés par PHP. $_POST est une variable auto-globale qui contient toutes les données envoyées par la méthode POST.

Si vous avions utilisé la méthode GET, les informations auraient été palcées dans la variable $_GET. La variable $_REQUEST peut être utilisée si vous ne souhaitez pas vous embarrasser de la méthode utilisée.

I.5- Utiliser des codes anciens avec les nouvelles versions de PHP

Pour la plupart, les développeurs de PHP ont tâché d'assurer la compatibilité ascendante, ce qui fait que de nombreux scripts écrits pour les anciennes versions sont aussi valables pour les nouvelles versions de PHP, idéalement sans modifications.

En pratique, deux modifications récentes (les plus importantes) doivent être apportées :

- Les anciennes variables $HTTP_*_VARS (qui devaient être indiquées comme globales pour être utilisées dans une fonction ou une méthode) sont obsolètes. Les nouveaux tableaux auto-globaux ont été introduits en PHP 4.1.0. Ce sont les variables suivantes : $_GET , $_POST , $_FILES , $_COOKIE , $_SERVER , $_ENV , $_REQUEST et $_SESSION. Les vieux tableaux $HTTP_*_VARS, tels que $HTTP_POST_VARS existent toujours depuis PHP 3. Depuis PHP 5.0.0, les tableaux prédéfinis PHP peuvent être désactivés avec l'option de configuration register_long_arrays.

- Les variables externes ne sont plus enregistrées dans l'environnement par défaut. En d'autres termes, depuis PHP 4.2.0, la directive PHP register_globals vaut off par défaut dans le php.ini . La méthode recommandée pour accéder à ces valeurs, est via les tableaux auto-globaux mentionnés ci-dessus.

I.6- Le fichier de configuration

Le fichier de configuration (appelé php.ini dans la version 4) est lu par PHP au démarrage. Si vous avez compilé PHP en module, le fichier n'est lu qu'une seule fois, au lancement du démon HTTP.

Extrait du php.ini :

```
; tout texte sur une ligne, situé après un point-virgule  ";" est ignoré
; [php] : les marqueurs de section (texte entre crochets) sont aussi ignorés
; Les valeurs booléenne peuvent être spécifiées comme ceci :
; true, on, yes
; ou false, off, no, none
register_globals = off
track_errors = yes

; vous pouvez placer les chaînes de caractères entre guillemets
include_path = ".:/usr/local/lib/php"

; Les anti-slash sont traités comme n'importe quel caractère
include_path = ".;c:\php\lib"
```

II/ REFERENCE DU LANGAGE

II.1- La syntaxe de base

2 jeux de balises servent à délimiter des blocs de code PHP :
<?php. . .?> et <script language="php">. . .</script>

C'est ce mécanisme qui vous permet d'inclure du code PHP dans des pages HTML : tout ce qui est placé hors des balises PHP est affiché sans modification, tandis que le contenu est exécuté.

II.1.a) Le séparateur d'instructions

Les instructions sont séparées par un point virgule à chaque fin d'instruction, comme en langage C. La balise de fin (?>) implique la fin d'une instruction, et donc ajoute implicitement un point virgule.

II.1.b) 4.1.3 Commentaires

PHP supporte les commentaires comme en C++ :

```php
<?php
  echo "Ceci est un test"; // Ceci est un commentaire sur une ligne comme en C++
  /* Ceci est un commentaire sur plusieurs lignes,
  comme en C et C++ */
  echo "Ceci est encore un test";
  echo "Enfin, le test final"; # Ceci est un commentaire comme en Shell Unix
?>
```

II.2- Les types

PHP supporte les huit types basiques :

- 4 types scalaires : booléen, entier , nombre à virgule flottante, chaîne de caractères.
- 2 types composés : tableau, objet.
- 2 types spéciaux : ressource et null.

Commentaires :
Si vous voulez vérifier le type et la valeur d'une variable ou d'une expression , utilisez la fonction : `var_dump`
Si vous souhaitez simplement une représentation lisible, utilisez la fonction : `gettype`
Pour vérifier la présence de certains types, utilisez les fonctions : `is_ type` (i.e. : `is_int, is_string, …`)
Si vous voulez forcer une variable à être convertie en un certain type, vous devez transtyper (`cast`) la variable ou utiliser la fonction : `settype.`

II.2.a) Les chaînes de caractères

Les chaînes de caractères sont des séquences de caractères. Le moyen le plus simple de spécifier une chaîne de caractères est d'utiliser les guillemets simples : `'`

Pour spécifier un guillemet simple littéral, vous devez l'échapper avec un anti-slash (`\`). Si un anti-slash doit apparaître dans votre chaîne ou bien en fin de chaîne, il faudra le doubler.

Si la chaîne est entourée de guillemets doubles, PHP va comprendre certaines séquences de caractères :

```
\n    Nouvelle ligne
\r    Retour à la ligne
\t    Tabulation horizontale
\\    Anti-slash
\$    Caractère $
\"    Guillemets doubles
\[0-7]{1,3}
      Séquence de caractères qui permet de rechercher un nombre en notation octale.
\x[0-9A-Fa-f]{1,2}
      Séquence de caractères qui permet de rechercher un nombre en notation hexadécimale.
```

Le plus important pour les chaînes à guillemets doubles est le fait que les variables qui s'y trouvent seront remplacées par leur valeur.

Syntaxe simple d'accès aux variables

Dès qu'un signe dollar `$` est rencontré, l'analyseur PHP va lire autant de caractère qu'il peut pour former un nom de variable valide. Entourez le nom de la variable avec des accolades pour indiquer explicitement son nom.

```php
<?php
  $boisson = 'vin';
  // OK, car "," n'est pas autorisé dans les noms de variables
  echo "Du $boisson, du pain et du fromage!";

  // Pas OK, car 's' peut faire partie d'un nom de variable
  echo 'Il a goûté plusieurs ' . $boissons;

  // OK
  echo "Il a goûté plusieurs ${boisson}s";
?>
```

Syntaxe complexe d'accès aux variables

Il suffit d'écrire une expression comme si elle était hors de la chaîne, puis de l'entourer d'accolades {}.

```php
<?php
  $super = 'fantastique';

  // Ne fonctionne pas. Affiche : Ceci est { fantastique}
  echo "Ceci est { $super}";
```

```
// Fonctionne. Affiche Ceci est fantastique
echo "This is {$super}";
echo "This is ${super}";

// Fonctionne
echo "Ce carré a un coté de {$square->width}00 centimètres.";
echo "Ceci fonctionne : " . $arr['foo'][3];
echo "Vous pouvez même écrire {$obj->values[3]->name}";
echo "Ceci est une valeur de variable variable : {${$name}}";
?>
```

Accès et modification des caractères d'une chaîne

Les caractères d'une chaîne sont accessibles et modifiables en spécifiant leur offset (le premier caractère est d'offset 0) entre accolade, après le nom de la variable.

```
<?php
// Lit le premier caractère de la chaîne
$str = 'Ceci est un test.';
$first = $str{0};

// Lit le dernier caractère de la chaîne
$str = 'Ceci est un test.';
$last = $str{strlen($str)-1};

// Modifie le dernier caractère de la chaîne
$str = 'Ceci est un test.';
$str{strlen($str)-1} = '!';
?>
```

Fonctions et opérateurs pratiques

La manipulation des chaînes de caractères est une opération redondante du travail de développeur, ainsi de nombreuses fonctions sont à votre disposition, en voici quelques une :

L'opérateur `'.'`	Concaténation des chaînes
`Substr ()`	Accès à une sous-chaîne d'une chaîne `string substr (string chaine, int début, [int longueur])`

`Strcmp ()`
`Strcasecmp ()`
`Strnatcmp()`
	Comparaison de chaîne
	`int strcmp (string chaîne1, string chaîne2)`

`Strlen ()`	Détermine la longueur d'une chaîne `int Strlen (string chaîne)`
`Strstr ()` `Strchr()`	Recherche d'une chaîne dans une chaîne `string strstr (string chaîne, string chaine à trouver)`
`Strpos()`	Retourne la position numérique d'une chaine dans une autre chaîne) `int strpos (string chaîne, string chaine à trouver)`
`Strval ()`	Convertion de chaîne

`Chop()`
`Ltrim ()`
`Trim ()`
	Elagage des chaînes (nettoyage des blancs)

Syntaxe Heredoc

Un autre moyen de délimiter les chaînes est d'utiliser la syntaxe dite " Heredoc " :

```
<<<, suivi d'un identifiant arbitraire, puis de la chaîne
```

Exemple :

```
<?php
$str = <<<EOD
Exemple de chaîne
s'étalant sur
plusieurs lignes
avec la syntaxe heredoc
EOD;
?>
```

L'identifiant utilisé doit suivre les mêmes règles que les étiquettes PHP : il ne doit contenir uniquement que des caractères alpha-numériques, et des soulignés ("_"), et enfin, commencer par un caractère alphabétique ou un souligné.

La ligne qui contient l'identifiant de fermeture ne doit contenir aucun autre caractère, hormis, éventuellement, un point-virgule. La syntaxe Heredoc se comporte exactement comme une chaîne à guillemets doubles, sans les guillemets doubles.

II.2.b) Les tableaux

Un tableau PHP est en fait une association ordonnée (littéralement, une map).

Une association est un type qui fait correspondre des valeurs à des clés. Ce type est optimisé de diverses façons, qui font que vous pouvez le manipuler comme un tableau à indices réels, une liste (vecteur), ou un table de hachage, dictionnaire, collection, pile, …

Création

Un tableau array peut être créé avec la fonction `array` :

```
array( Array value , ... )
```

Exemple de tableau :

```
<?php
$arr = array("foo" => "bar", 12 => true);
echo $arr["foo"];                                    // bar
echo $arr[12];                                       // 1
?>
```

Exemple de tableau associatif

```
<?php
array("untableau" => array(6 => 5, 13 => 9, "a" => 43));
echo $arr["untableau"][6];                           // 5
echo $arr["untableau"][13];                          // 9
echo $arr["untableau"]["a"];                         // 43
?>
```

Commentaires :
L'assignation de valeurs de tableau se fait en spécifiant la clé entre crochets. Si vous omettez la clé ("`$tableau[]`"), la valeur sera ajoutée à la fin du tableau.

```
<?php
$arr = array(5 => 1, 12 => 2);
$arr[] = 56; // Ceci revient à $arr[13] = 56;
?>
```

Si `$arr` n'existe pas, il sera créé. Cela en fait une alternative pour créer un tableau.

Suppression

Pour modifier une valeur, assignez-lui une nouvelle valeur. Pour supprimer une valeur, utilisez la fonction `unset` :

```
<?php
$arr["x"] = 42;         // Ceci ajoute un nouvel élément
// avec l'index "x"
unset($arr[5]);         // Ceci efface un élément du tableau
unset($arr);            // Ceci efface tout le tableau
?>
```

`Unset` ne réindexe pas le tableau.

Génération automatique de clé

Code	Affichage

`<?php`
`// Crée un simple tableau`
`$array = array(1, 2, 3, 4, 5);`
`print_r($array);`
 	
`// On efface tous les éléments`
`foreach ($array as $i => $value) {`
`  unset($array[$i]);`
`}`
`print r($array);`
	`Array (`
	`[0] => 1`
	`[1] => 2`
	`[2] => 3`
	`[3] => 4`
	`[4] => 5`
	`)`
	
	`Array ()`

`// Ajout d'une valeur : la nouvelle clé est 5`
`// et non pas 0, comme on l'attendrait.`
`$array[] = 6;`
`print_r($array);`
 	
`// Re-indexation :`
`$array = array_values($array);`
`$array[] = 7;`
`print_r($array);`
`?>`
	`Array ([5] => 6)`
	
	`Array (`
	`[0] => 6`
	`[1] => 7`
	`)`

Pourquoi est-ce que $foo[bar] est invalide?

Dans vos vieux scripts, vous pouvez avoir utilisé la syntaxe suivante :

```
<?php
  $foo[bar] = 'ennemi';
  echo $foo[bar];
?>
```

Cela est mauvais, mais ça marche. La raison est que PHP attend une constante entre crochets (`bar`) plutôt qu'une chaîne `('bar'`, notez les guillemets).

Cela ne signifie pas que vous devez toujours mettre les clés entre guillemets. Vous n'allez pas utiliser les guillemets avec les clés qui sont des constantes ou des variables , car cela empêchera PHP de les interpréter correctement.

Exemples

- Utilisation des tableaux

```
<?php
  $a = array( 'couleur' => 'rouge', 'forme' => 'rond', 4) ;   //4 sera en clé :0
// est équivalent à
  $a['couleur'] = 'rouge';
  $a['forme'] = 'rond';
  $a[] = 4;                     // cette clé sera 0
?>
```

- Utilisation des tableaux

```
<?php
  $b[] = 'a';
  $b[] = 'b';
  $b[] = 'c';
// est équivalent à
  $b = array(0 => 'a' , 1 => 'b' , 2 => 'c')
// est équivalent à
  $b = array('a' , 'b' , 'c' )
?>
```

- Utilisation de array

```
<?php
  $c = array( 10           // clé = 0
             ,5 => 6
             ,3 => 7
             ,'a' => 4
             ,11           // clé = 6 (index maximum : 5)
             ,'8' => 2     // clé = 8 (entier)
             ,'02' => 77   // clé = '02' (chaîne)
             ,0 => 12      // la valeur de la clé 10 sera remplacée par 12
  );
?>
```

- Collection

```
<?php
  $couleurs = array('rouge','bleu','vert','jaune');
  foreach ( $couleurs as $couleur ){
    echo "Aimez-vous la couleur $couleur?\n";
  }
?>
```

L'exemple ci-dessus va afficher :

```
Aimez-vous la couleur rouge?
Aimez-vous la couleur bleu?
Aimez-vous la couleur vert?
Aimez-vous la couleur jaune?
```

- Remplissage d'un tableau

```
<?php
// remplit un tableau avec les noms de fichiers d'un dossier
  $handle = opendir('.');
  while ( $file = readdir($handle) ) {
    $files[] = $file;
  }
  closedir($handle);
?>
```

- Tri de tableaux

```
<?php
  sort($files);
  print_r($files);
?>
```

- Tableaux multi-dimensionnels et récursifs

```
<?php
$fruits = array ("fruits => array ("a" => "orange", "b" => "banane", "c" => "pomme"),
                 "nombre" => array (1, 2, 3, 4, 5, 6,),
                 "trou" => array ("premier", 5 => "second", "troisième"));

// Exemples d'utilisations des tableaux ci-dessus
echo $fruits["trou"][5];         // affiche "second"
echo $fruits["fruits"]["a"];     // affiche "orange"
unset($fruits["trou"][0]);       // supprime "premier"

// Créer un tableau multidimensionnel
$juices["pomme"]["vert"] = 'bon';
?>
```

- Passage par adresse

Soyez conscients que l'assignation de valeur dans un tableau entraine automatiquement la copie de valeurs. Vous devez utiliser l'opérateur de référence `&` pour copier un tableau par référence :

```
<?php
$arr1 = array(2, 3);
$arr2 = $arr1;
$arr2[] = 4;              // $arr2 est modifié, $arr1 vaut toujours array(2, 3)
$arr3 = &$arr1;
$arr3[] = 4;              // maintenant $arr1 et $arr3 sont identiques
?>
```

II.2.c) La valeur NULL

La valeur spéciale `NULL` représente l'absence de valeur. Une variable avec la valeur NULL n'a pas de valeur :

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

II.2.d) Pseudo-types utilisés

- `Mixed` : mixed indique qu'un paramètre accepte plusieurs types, mais pas forcément tous les types.
- `Number` : number indique qu'un paramètre peut être du type entier ou nombre à virgule flottante.
- `Callback` : Exemple de fonction de callback :

```
<?php
// Exemple simple de fonction de callback
function foobar() {
  echo 'Bonjour le monde!';
}
call_user_function("foobar");

// Exemple de méthode de callback
class foo {
  function bar() {
    echo 'Bonjour le monde!';
  }
}
$foo = new foo;
call_user_function(array($foo, "bar"));     // appel d'une méthode d'objet
call_user_function(array("foo", "bar"));    // appel d'une méthode statique de classe
?>
```

II.2.e) Définition du type

PHP ne nécessite pas de déclaration explicite du type d'une variable. Le type d'une variable est déterminé par le contexte d'utilisation.
Par exemple, si vous assignez une `chaîne de caractères` à la variable `var` , `var` devient une chaîne de caractère. Si vous assignez un nombre `entier` à `var` , elle devient un `entier`.

Transtypage

La conversion de type en PHP fonctionne de la même manière qu'en C : le nom du type désiré est écrit entre parenthèses devant la variable à transtyper ("`cast`").

```
<?php
$foo = 10;                  // $foo est un entier
$bar = (double) $foo;       // $bar est un double
?>
```

Les conversions autorisées sont :

```
(int) , (integer)             - type entier
(bool) , (boolean)            - booléen
(real) , (double) , (float)   - type double
(string)                      - type chaîne
(array)                       - type tableau
(object)                      - type objet
```

Transtypage en chaîne :

```
<?php
$foo = 10;                  // $foo est un entier
$str = "$foo";              // $str est une chaîne
$fst = (string) $foo;       // $fst est aussi une chaîne
```

II.3- Les variables

II.3.a) Généralités

En PHP, les variables sont représentées par un signe dollar "`$`" suivi du nom de la variable. Le nom est sensible à la casse et un nom de variable valide doit commencer par une lettre ou un souligné (_).

En PHP 3, les variables sont toujours assignées par valeur. C'est-à-dire, lorsque vous assignez une expression à une variable, la valeur de l'expression est recopiée dans la variable.

PHP 4 permet aussi d'assigner les valeurs aux variables par référence. Cela signifie que la nouvelle variable ne fait que référencer ("pointe sur") la variable originale. Les modifications de la nouvelle variable affecteront l'ancienne, et vice versa.

Pour assigner par référence, ajoutez simplement un `&` au début de la variable qui est assignée :

```
<?php
$foo = 'Pierre';              // Assigne la valeur 'Pierre' à $foo
$bar = &$foo;                 // Référence $foo avec $bar.
$bar = "Mon nom est Pierre";  // Modifie $bar...
echo $foo;                    // $foo est aussi modifiée
echo $bar;
?>
```

II.3.b) Variables prédéfinies

Depuis la version PHP 4.2.0, la valeur par défaut de la directive PHP `register_globals` est off. Ceci est une évolution majeure de PHP, ainsi

- pour lire `DOCUMENT_ROOT` vous devez utiliser `$_SERVER['DOCUMENT_ROOT']` au lieu de `$DOCUMENT_ROOT`,
- il faut lire `$_GET['id']` dans l'URL http://www.example.com/test.php?id=3 au lieu de `$id`,
- ou encore `$_ENV['HOME']` au lieu de `$HOME`.

Depuis la version 4.1.0, PHP fournit un jeu de tableaux prédéfinis, contenant les variables du serveur (si possible), les variables d'environnement et celle d'entrées. Ces nouveaux tableaux sont automatiquement globaux : ils sont automatiquement disponibles dans tous les environnements d'exécution, sans avoir à utiliser le mot réservé `global` . Pour cette raison, ils sont dits '`auto-globaux`' ou bien encore '`superglobaux`'.

- `$GLOBALS` : Contient une référence sur chaque variable qui est actuellement disponible dans l'environnement d'exécution global. Les clés de ce tableau sont les noms des variables globales.

- `$_SERVER` : Les variables fournies par le serveur web, ou bien directement liées à l'environnement d'exécution du script courant. (Nouvelle version de `$HTTP_SERVER_VARS`).

- `$_GET` : Les variables fournies par le protocole HTTP en méthode GET. (Nouvelle version de `$HTTP_GET_VARS`).

- `$_POST` : Les variables fournies par le protocole HTTP en méthode POST. (Nouvelle version de `$HTTP_POST_VARS`)

- `$_COOKIE` : Les variables fournies par le protocole HTTP, dans les cookies. (Nouvelle version de `$HTTP_COOKIE_VARS`)

- `$_FILES` : Les variables fournies par le protocole HTTP, suite à un téléchargement de fichier. (Nouvelle version de `$HTTP_POST_FILES`)

- `$_ENV` : Les variables fournies par l'environnement. (Nouvelle version de `$HTTP_ENV_VARS`)

- `$_REQUEST` : Les variables fournies au script par n'importe quel mécanisme d'entrée et qui ne doit recevoir q'une confiance limitée.

- `$_SESSION` : Les variables qui sont actuellement enregistrées dans la session attachée au script. (Nouvelle version de `$HTTP_SESSION_VARS`)

II.3.c) Portée des variables

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 :

```
<?php
$a = 1;              // portée globale
function test() {
  echo $a;           // portée locale
}
test();
?>
```

Le script n'affichera rien à l'écran, car la fonction `echo` utilise la variable locale `$a`, et celle-ci n'a pas été assignée préalablement dans la fonction. Vous pouvez noter que ce concept diffère un petit peu du langage C dans lequel une variable globale est automatiquement accessible dans les fonctions, à moins d'être redéfinie localement dans la fonction.

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

Redéfinition

```
<?php
  $a = 1;
  function test() {
    global $a;
    echo $a;
  }
  test();
?>
```

ou

Tableau associatif

```
<?php
  $a = 1;
  function test() {
    echo $GLOBALS['a'];
  }
  test();
?>
```

Le mot clé global

Le tableau `$GLOBALS` est un tableau associatif avec le nom des variables globales comme clef et les valeurs des éléments du tableau comme valeur des variables. Notez que `$GLOBALS` existe dans tous les contextes, car `$GLOBALS` est un superglobal.

Utilisation des variables static

Une autre caractéristique importante de la portée des variables est la notion de variable `static` . Une variable statique a une portée locale uniquement, mais elle ne perd pas sa valeur lorsque le script appelle la fonction.

```
<?php
function test() {
  $a = 0;
  echo $a;
  $a++;
}?>
```

Cette fonction est un peu inutile car à chaque fois qu'elle est appelée, elle initialise `$a` à 0 et affiche "0". L'incrémentation de la variable `($a ++)` ne sert pas, car dès que la fonction est terminée la variable disparaît.

Pour faire une fonction de comptage utile, c'est-à-dire qui ne perdra pas la trace du compteur, la variable `$a` est déclarée comme une variable statique :

```
<?php
function test() {
  static $a = 0;
  echo $a;
  $a++;
}?>
```

Maintenant, à chaque fois que la fonction `Test()` est appelée, elle affichera une valeur de `$a` incrémentée de 1.

Les variables statiques sont essentielles lorsque vous faites des appels récursifs à une fonction. Une fonction récursive est une fonction qui s'appelle elle-même.

Test d'existence

Pour tester si une variable a déjà été affecté, il est possible d'utiliser la fonction : `isset($variable)` ou son contraire : `empty ($variable)`.

Exemple :

```
if ( !isset ($variable)) $variable ="";
```

II.3.d) Les variables dynamiques

Il est pratique d'avoir parfois des noms de variables qui sont variables. C'est-à-dire un nom de variable qui est affectée et utilisée dynamiquement. Une variable classique :

```
<?php
  $a = 'bonjour';
?>
```

Une variable dynamique prend la valeur d'une variable et l'utilise comme nom d'une autre variable. Dans l'exemple ci-dessous, `bonjour` peut être utilisé comme le nom d'une variable en utilisant le `"$$"` précédent la variable. Les variables variables :

```
<?php
  $$a = 'monde';
?>
```

A ce niveau, deux variables ont été définies et stockées dans l'arbre des symboles PHP :

- `$a` avec comme valeur `"bonjour"`,
- `$bonjour` avec comme valeur `"monde"`.

Alors, l'instruction :

```
<?php
  echo "$a ${$a}";
?>
```

produira le même affichage que :

```
<?php
  echo "$a $bonjour";
?>
```

c'est-à-dire :

```
bonjour monde
```

Commentaire :

Afin de pouvoir utiliser les variables dynamiques avec les tableaux, vous avez à résoudre un problème ambigu. Si vous écrivez `$$a[1]`, l'analyseur a besoin de savoir si vous parler de la variable qui a pour nom `$a[1]` ou bien si vous voulez l'index `[1]` de la variable `$$a`. La syntaxe pour résoudre cette ambiguité est la suivante : `${$a[1]}` pour le premier cas, et `${$a}[1]` pour le deuxième.

II.3.e) Variables externes à PHP

Formulaires HTML (GET et POST)

Lorsqu'un formulaire est envoyé à un script PHP, toutes les variables du formulaire seront automatiquement disponibles dans le script. Par exemple, considérons le formulaire suivant :

```
<form action="foo.php" method="post">
  Nom : <input type="text" name="username"><br />
  Email: <input type="text" name="email"><br />
  <input type="submit" name="submit" value="Envoi!">
</form>
```

Pour accéder aux variables de formulaires POST, vous pouvez utiliser les syntaxes suivantes :

```
echo $_POST['username'];
```

ou

```
echo $_REQUEST['username'];
```

Utiliser un formulaire de type `GET` est similaire, (hormis le fait que vous deviez utiliser `$_GET`).
GET s'applique aussi à la `QUERY_STRING` (les informations disponibles après le '`?`' dans une URL). De ce fait, `http://www.example.com/test.php?id=3` contient les données de `GET`, qui sont accessibles via `$_GET['id']` ou `$_REQUEST` (et import_request_variables).

Exemple d'un formulaire qui se poste à lui-même des données, et les affiche :

```
<?php
if ($_POST['action'] == 'submitted') {
  print '<pre>';
  print_r ($_POST);
  echo '<a href="'. $_SERVER['PHP_SELF'] .'">Essayez à nouveau</a>';
  echo '</pre>';
} else {
?>
  <form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
    Nom : <input type="text" name="personal[name]"><br />
    Email : <input type="text" name="personal[email]"><br />
    Biere : <br />
    <select multiple name="vin[]">
      <option value="bordeaux">bordeaux</option>
      <option value="beaujolais">beaujolais</option>
      <option value="loire">loire</option>
    </select><br />
    <input type="hidden" name="action" value="submitted">
    <input type="submit" name="submit" value="submit me!">
  </form>
<?php
}
?>
```

Commentaire :
Lors de la soumission d'un formulaire, il est possible d'utiliser une image au lieu d'un bouton standard, comme ceci :

```
<input type="image" src="image.gif" name="sub">
```

Lorsque l'utilisateur clique sur cette image, le formulaire associé est envoyé au serveur, avec deux données supplémentaires, `sub_x` et `sub_y` . Elles contiennent les coordonnées du clic de l'utilisateur dans l'image.

Passage de variable par le lien

Les methodes `Post` et `Get` permettent de récupérer des variables (ansi que leurs valeurs) passées par des objets (zone de saisie, liste, …) d'un formulaire.

Mais il est aussi possible de passer une variable sans avoir à créer d'objets, il suffit pour cela de passer la variable directement dans le lien.
Le nom de la page à ouvrir (présent dans la méthode `action`), sera suivi d'un `?` et de la liste des variables avec leurs valeurs :

```
<page.php>?<nom_variable>=<valeur> [&& <nom_variable2>=<valeur>]
```

Exemple :

```
<form action="foo.php?test=vrai" >
```

La valeur de la variable sera récupérée à l'aide de `$_REQUEST (<nom_variable>)`.

II.4- Les constantes

Tous comme les superglobals , les constantes sont accessibles de manière globale. Vous pouvez la définir n'importe où, et y accéder depuis n'importe quelle fonction.

```
<?php
define("CONSTANTE", "Bonjour le monde.");    //Définition d'une constante
echo CONSTANTE;                              //Utilisation d'une constante
?>
```

Une fois qu'une constante est définie, elle ne peut jamais être modifiée, ou détruite.

Constantes magiques

PHP fournit un grand nombre de constantes magiques. Il y a cinq constantes magiques qui changent suivant l'emplacement où elles sont utilisées :

```
__LINE__        La ligne courante dans le fichier.
__FILE__        Le chemin complet et le nom du fichier courant.
__FUNCTION__    Le nom de la fonction.
__CLASS__       Le nom de la classe courante.
__METHOD__      Le nom de la méthode courante.
```

II.5- Les opérateurs

Les opérateurs arithmétiques

Signe	Signification
+	addition
-	soustraction
*	multiplication
/	par division
%	reste de la division par
=	assignation

Les opérateurs associatifs

Signe	Description
+=	plus égal
-=	moins égal
*=	multiplié égal
/=	divisé égal
.=	Assignation et concaténation

Les opérateurs de comparaison

Signe	Signification
==	égal
===	égal et de type identique
<	inférieur
<=	inférieur ou égal
>	supérieur
=<	supérieur ou égal
!= ou <>	différent

Les opérateurs logiques

Signe	Description
&&	et
||	ou

Les opérateurs d'incrémentation

Signe	Description
x++	incrémentation (x++ est le même que x=x+1)
x--	décrémentation (x-- est le même que x=x-1)

II.5.a) Opérateur de contrôle d'erreur

PHP supporte un opérateur de contrôle d'erreur : `@`.

Lorsque cet opérateur est ajouté en préfixe d'une expression PHP, les messages d'erreur qui peuvent être générés par cette expression seront ignorés.

Si l'option `track_errors` est activée, les messages d'erreurs générés par une expression seront sauvés dans la variable globale `$php_errormsg` . Cette variable sera écrasée à chaque erreur.

```
<?php
/* Erreur intentionnelle (le fichier n'existe pas): */
$mon_fichier = @file ('non_file') or die ("Impossible d'ouvrir : '$php_errormsg'");

// Cela fonctionne avec n'importe quelle expression, pas seulement les fonctions
```

```
$value = @$cache[$key];
// la ligne ci-dessus n'affichera pas d'alerte si la clé $key du tableau n'existe pas
?>
```

L'opérateur `@` ne fonctionne qu'avec les expressions : si vous pouvez prendre la valeur de quelque chose, vous pouvez le préfixer avec `@`.

II.5.b) Opérateur d'exécution

PHP supporte un opérateur d'exécution : guillemets obliques ("``"). PHP essaiera d'exécuter le contenu de ces guillemets obliques comme une commande shell. Le résultat sera retourné. Utilisez les guillemets obliques revient à utiliser la fonction `shell_exec` :

```
<?php
$output = `ls -al`;
echo "<pre>$output</pre>";
?>
```

II.5.c) Opérateurs de tableaux

Signe	***Signification***
+	Union
==	Egalité sans importance dans l'ordre
===	Egalité dans le même ordre
!= ou <>	Inégalité

II.5.d) Opérateur de type

PHP a un seul opérateur de type : `instanceof,` qui sert à déterminer si un objet est d'une classe donnée.

```
<?php
class A { }
class B { }
$thing = new A;
if ($thing instanceof A) {
  echo 'A';
}
if ($thing instanceof B) {
  echo 'B';
}
?>
```

II.6- Les structures de contrôle

Si… alors… Sinon…	`if (expression) { instr1 } else { instr2 }`
Si… alors… Sinon…	`(expression) ? inst1 : inst2`
Selon exp cas 1 : instr 1 … cas n : instr n Sinon : instr sinon FinSelon	`switch ($exp) { case 1: instr 1; break; … case n: instrs n; break; default : instr default; } ?>`
Pour i de <val init> à <cond> pas <prog> Faire …	`for (val init ; cond ; prog) { instructions; }`
Tant Que <condition> Faire …	`while (condition vraie){ instructions }`
Répéter … Jusqu'à <condition>	`Do { instructions } while (condition vraie) ;`

Pour chaque <elt> de <ensemble_elt> Faire
…
	`foreach (ensemble_elt as $elt) instr`
	ou
	`foreach (ensemble_elt as $key => $elt) instr`

Interruption volontaire de boucle	`Break ;`
Saut d'instructions	`Continue`

II.6.a) Foreach

Le foreach travaille sur une copie du tableau spécifié, et pas sur le tableau lui-même. Par conséquent, les modifications faites dans le tableau ne seront pas prises en compte dans le tableau original.

Exemple avec Foreach et While :

```
<?php
$arr = array("un", "deux", "trois");
reset($arr);

while (list(, $value) = each ($arr)) {
  echo "Valeur : $value<br />\n";
}

foreach ($arr as $value) {
  echo "Valeur : $value<br />\n";
}
?>
```

Exemple pour Foreach

- la valeur seulement

```
$a = array (1, 2, 3, 17);
foreach ($a as $v) {
  print "Current value of \$a : $v.\n";
}
```

- la valeur et sa clé d'index

```
$a = array (1, 2, 3, 17);
$i = 0;                          /* uniquement pour l'illustration */
foreach($a as $v) {
  print "\$a[$i] => $v.\n";
  $i++;
}
```

- la clé et la valeur

```
$a = array ("un" => 1, "deux" => 2, "trois" => 3, "dix-sept" => 17);
foreach($a as $k => $v) {
  print "\$a[$k] => $v.\n";
}
```

- tableaux multi-dimensionnels

```
$a[0][0] = "a";
$a[0][1] = "b";
$a[1][0] = "y";
$a[1][1] = "z";
foreach($a as $v1) {
  foreach ($v1 as $v2) {
    print "$v2\n";
  }
}
```

- tableaux dynamiques

```
foreach(array(1, 2, 3, 4, 5) as $v) {
  print "$v\n";
}
```

II.6.b) Require

`Require` inclut et exécute un fichier PHP. La commande `require` se remplace elle-même par le contenu du fichier spécifié, tel que décrit dans la documentation de la fonction `include`.
`Require et include` sont identiques, sauf dans leur façon de gérer les erreurs. `Include` produit une `Alerte ( warning )` tandis que `require` génère une `erreur fatale`.

```
<?php
require 'prepend.php';
require $somefile;
require ('unfichier.txt');
?>
```

Il existe une vairante : `require_once`.

II.6.c) Include

La fonction `include` inclut et exécute le fichier spécifié en argument. Les fichiers à inclure sont d'abord recherchés dans le dossier désigné par `include_path`, relativement au dossier courant, puis dans `include_path`, relativement au dossier de travail du script.
Exemple, soit : `vars.php`

```
<?php
  $couleur = 'verte';
  $fruit = 'pomme';
?>
```

Utilisation : `test.php`

```
<?php
  echo "Une $couleur $fruit";          // Une
  include 'vars.php';
  echo "Une $coleur $fruit";           // Une verte pomme
?>
```

Il existe une vairante : `include_once`.

Le `require` et le `include` permettent de simuler des entêtes ou des peids de pages de site par exemple.

II.7- Les fonctions

Une fonction peut être définie en utilisant la syntaxe suivante :

```
<?php
function foo($arg_1, …, $arg_n) {
  echo "Exemple de fonction.\n";
  return $retval;
}?>
```

II.7.a) Les arguments de fonction

PHP supporte le passage d'arguments par valeur (méthode par défaut) et par référence.

Passage d'arguments par référence

Par défaut, les arguments sont passés à la fonction par valeur, si vous voulez que vos fonctions puissent changer la valeur des arguments, vous devez passer ces arguments par référence.

Si vous voulez qu'un argument soit toujours passé par référence, vous pouvez ajouter un ' `&` ' devant l'argument dans la déclaration de la fonction :

```
<?php
function add_some_extra(&$string) {
  $string .= ', et un peu plus.';
}
$str = 'Ceci est une chaîne';
add_some_extra($str);
echo $str;                      // affiche 'Ceci est une chaîne, et un peu plus.'
?>
```

Valeur par défaut des arguments

Vous pouvez définir comme en C++ des valeurs par défaut pour les arguments de type scalaire :

```
<?php
function servir_cafe ($type = "cappuccino") {
  return "Servir un $type.\n";
}
echo servir_cafe();
echo servir_cafe("espresso");
?>
```

L'exemple ci-dessus va afficher :

```
Servir un cappuccino.
Servir un espresso.
```

La valeur par défaut d'un argument doit obligatoirement être une constante, et ne peut être ni une variable, ni un membre de classe, ni un appel de fonction.

Les valeurs de retour

Tous les types de variables peuvent être renvoyés, tableaux et objets compris.

```
<?php
function carre ($num) {
  return $num * $num;
```

```
}
echo carre (4);                          // affiche '16'.
?>
```

Commentaire :
Vous ne pouvez pas renvoyer plusieurs valeurs en même temps, mais vous pouvez obtenir le même résultat en renvoyant un tableau.

III/ L'ACCES AUX BASES DE DONNEES

Avant d'entamer ce chapitre, vous devez avoir compris le chapitre sur les « outils » et le « fonctionnement des sites web », ainsi que le chapitre sur « easyphp » et la « construction d'une base de données ».

Nous verons la connexion à une base `MySql` (le plus classique), que nous découperons en deux catégories. Les requêtes : `LMD-LDD`, qui permettent d'ajouter, modifier ou supprimer des informations dans la base et les requêtes : `LID`, qui permettent de récupérer des informations de la base dans le but de les utiliser dans nos pages HTML.

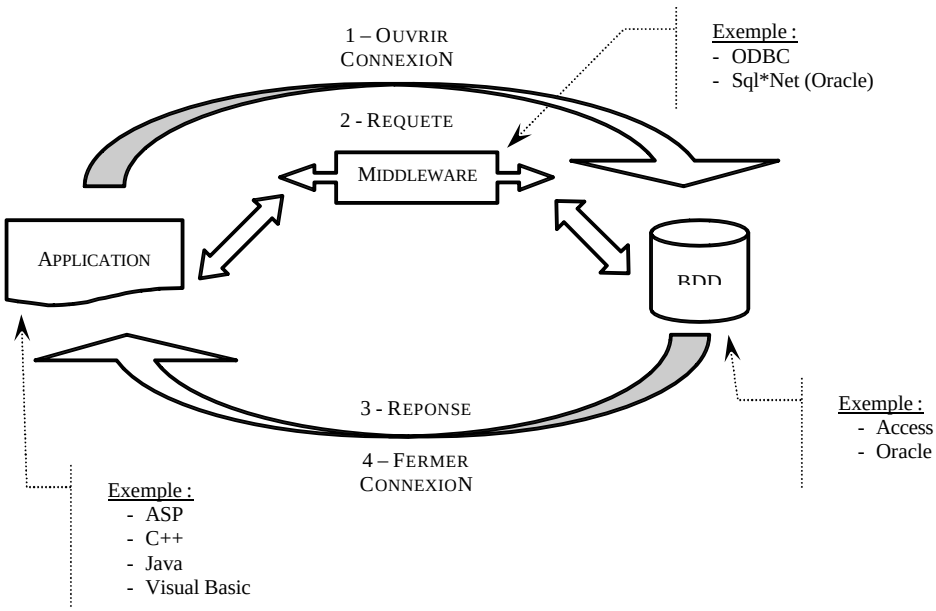
De plus, nous verrons l'accès à toutes les autres bases de données via `le middelware ODBC`.

III.1- Le middleware

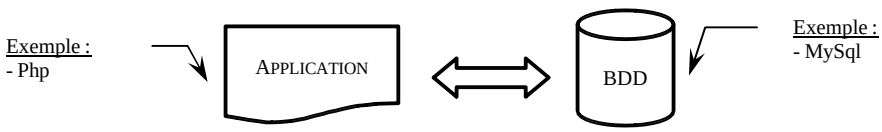
III.1.a) Principe

Le middleware (ou médiateur) va vous permettre de faire une liaison entre un langage et une base de données. Il permet de transcrire le message envoyé par le langage, pour être compréhensible par le SGBD. Il existe différents types de middleware (propriétaire, éditeur ou libre). Ceci implique que les médiateurs peuvent être obligatoires ou facultatifs.

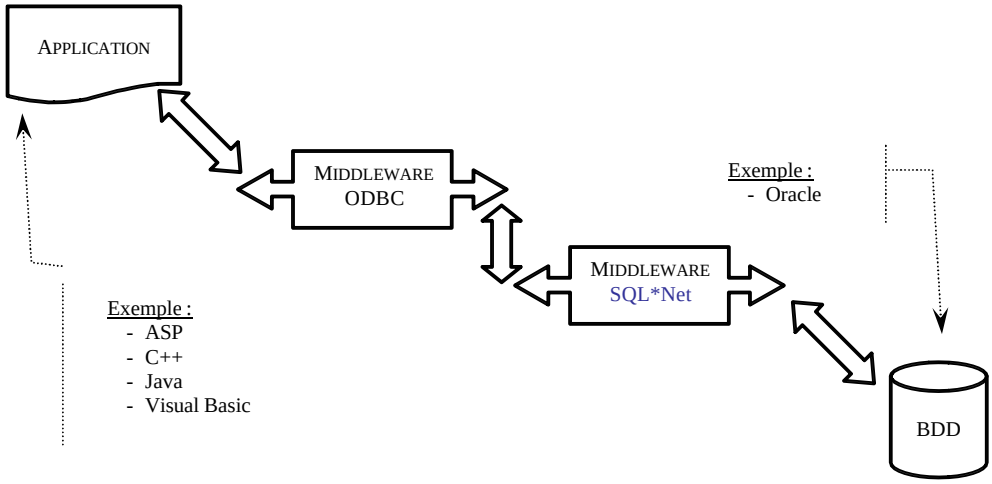
Principe de fonctionnement :



Connexion avec middleware non obligatoire : `vers MySql`



Connexion avec middleware obligatoire : vers Oracle

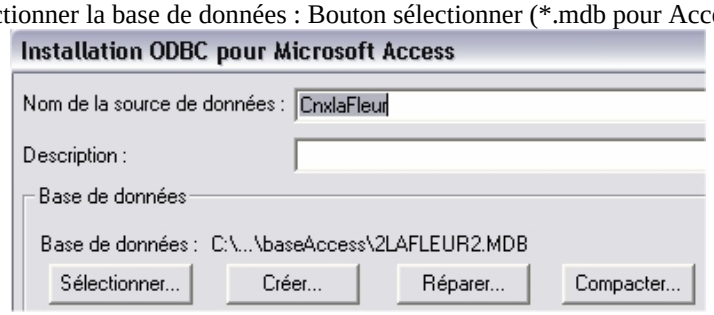


III.1.b) Implémentation d'ODBC

Le plus populaire des middlewares sera étudié en S2.

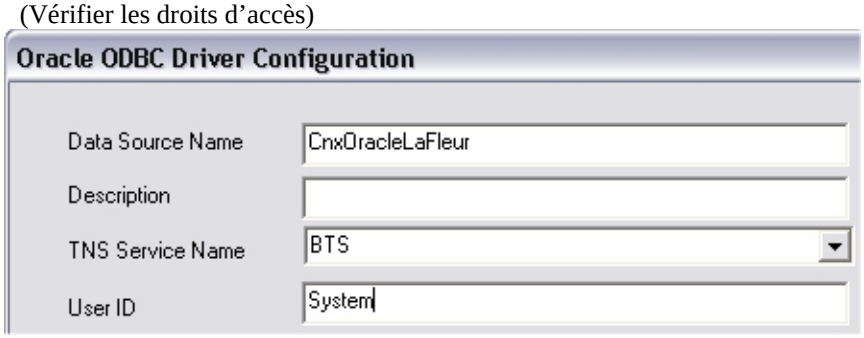
Création d'un middleware ODBC (ex : Base Access)

- Ajouter un DSN Système
- Choisir le type de la base : Access
- Nommer le lien : CnxLaFleur
- Sélectionner la base de données : Bouton sélectionner (*.mdb pour Access)



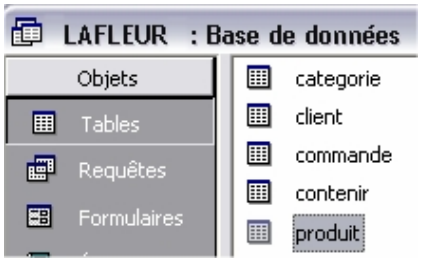
Création d'un middleware ODBC (ex : Base Oracle 9i)

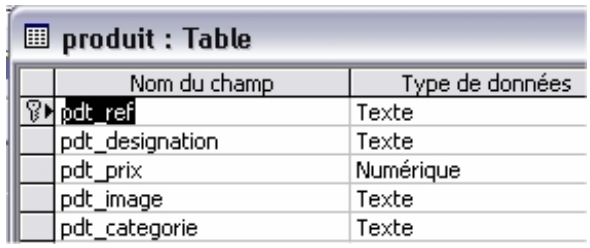
- Ajouter un DSN Système
- Choisir le type de la base : Oracle
  (de préférence le middleware du propriétaire et non celui de Microsoft)
- Nommer le lien : CnxLaFleur
- Nommer le Service Name : BTS
  (Nom de la base Oracle où se trouvent vos tables)
- Donner un nom d'utilisateur : System
  (Vérifier les droits d'accès)



III.2- La base de données

III.2.a) Exemple de base Access





III.2.b) Exemple de base MySql

Cf cours sur EasyPhp.

III.3- Php - MySQL

III.3.a) Exemple : Requête du LMD ou LDD : Ajout, Suppression ou Modification

Ces pages doivent récupérer de l'information, afin de pouvoir ajouter, modifier ou supprimer des éléments de votre base de données. Vous devez donc avant l'appel à la base de données, avoir récupérer les informations qui vous semblent utilent.

1. Conception de la fenêtre de saisie : `formSaisieProduit.php`

```
<html> <head> <title>menu</title> </head>
<body>
  <h2>Sté Lafleur</h2>
  <form name="formSaisieProduits" action="ajoutProduit.php" method="POST">
    Entrez une référence : <input type="text" size="5" name="reference"><p>
    Entrez une désignation : <input type="text" size="20" name="designation"><p>
    Entrez un prix : <input type="text" size="10" name="prix"><p>
    Entrez un nom d'image  : <input type="text" size="10" name="image"><p>
    Entrez une catégorie : <input type="text" size="10" name="categorie"><p>
    <input type="submit" value="envoyer">
    <input type="reset" value="annuler">
  </form>
</body> </html>
```

Commentaire :

- Servez vous du javascript pour vérifier la validité des saisies.
- Le click sur le bouton `envoyer` permet d'appeler la page `ajoutProduit.php`.

2. Conception de la page d'ajout dans la base : `ajoutProduit.php`

```
<html> <head> <title>Ajout Produit</title> </head>
<body>
  <h2>Sté Lafleur</h2>
  <p><b>Nos produits</b></P>
  <?
```

```
    //Inclusion du fichier de connexion
    include ('Connexion_base.php');
    //Connexion au serveur
    $idconnexion = BDD_Connect();
    //Si la connexion est ok
    if ($idconnexion) {
      //Requête
      $requete = "insert into produit values ('".$_POST['reference']."','"
                          .$_POST['designation']."',".$_POST['prix'].",'".$_POST['image']
                                                    ."','".$_POST['categorie']."')";
      //exécution de la requête et test de réussite
      $Resultat = mysql_query($requete);
      if ($Resultat) {
        echo 'Insertion OK !';
      } else {
        echo 'Insertion RATE !';
      }
    }
    //Fermeture des objets pour libération mémoire
    mysql_close ($idconnexion);
  ?>
  <a href="formSaisieProduit.php"> retour </a>
</body> </html>
```

3. La connexion est placée dans le fichier : `Connexion_base.php`

```
<?php
$hote = 'localhost';
$utilisateur = 'root';
$mdp ='';
$base = 'lafleur';

function BDD_Connect() {
  global $hote, $utilisateur, $mdp, $base;
  // Connexion à la base de données
  $result = mysql_connect($hote, $utilisateur, $mdp) or die("Connection Impossible");
  if (!$result) {
      return false;
  } else {
    //Sélection de la base
    $result2 = mysql_select_db($base) or die("Base inexistante");
    if (!$result2) {
      return false;
    } else {
      return $result;
    }
  }
}
```

Commentaire :

- Les ordres de connexion sont placés dans un fichier externe, afin de pouvoir les réutiliser plus facilement. (action répétitive et maintenance aisée en cas de modification des paramètres de connexion).
- Le fichier est appellé à l'aide l'instruction : `include`

Trame d'une requete LMD (sans tests)

```
<?php
// Connexion à la base de données
mysql_connect(<hote>, <utilisateur>, <mot_de_passe>) or die("Connection Impossible");
//Sélection de la base
mysql_select_db(<base>) or die("Base inexistante");
/Requete : 3 méthodes de récupération des variables en fonction du formulaire
$requete = "insert into <table> values ('" .$_POST['reference'] …
ou  $requete = "insert into <table> values ('" .$_GET['reference'] …
ou  $requete = "insert into <table> values ('" .$_REQUEST['reference'] …
//exécution de la requête
$Resultat = mysql_query($requete);
//Fermeture des objets pour libération mémoire
mysql_close ($idconnexion);
?>
```

III.3.b) Requête du LID : Affichage

Ces pages servent à afficher des données de la base. Vous pouvez avoir besoin d'informations (récupérées par un formulaire), qui affineront votre recherche et amélioreront la rapidité de vos traitements.

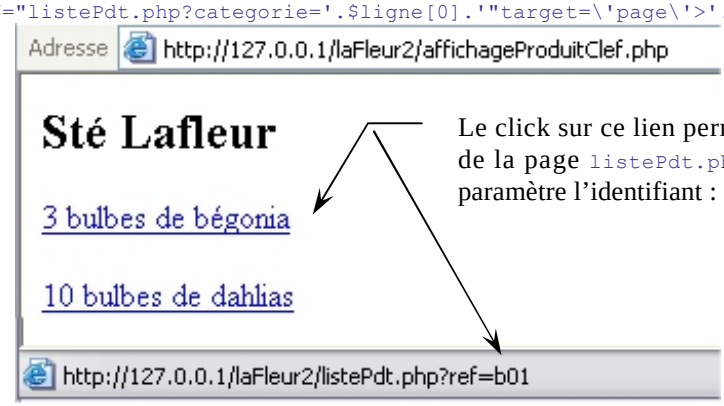
1. Affichage de la table produit (sans présentation) : `AffichageProduit.php`

```
<html> <head> <title>menu</title> </head>
<body>
  <h2>Sté Lafleur</h2>
  <?
    //Inclusion du fichier de connexion
    include ('Connexion_base.php');
    //Connexion au serveur
    $idconnexion = BDD_Connect();
    //Si la connexion est ok
    if ($idconnexion) {
      //Requête
      $requete = 'select * from produit';
      //Récupération du résultat de la requête dans jeuResultat
      $jeuResultat = mysql_query($requete);
      //Pour chaque ligne de jeuResultat (pris comme un tableau)
      while ($ligne = mysql_fetch_array ($jeuResultat)) {
        //Construction de la page HTML
        echo '<p>'.$ligne[0].' - '.$ligne[1];
      }
    }
    //Fermeture des objets pour libération mémoire
    mysql_free_result($jeuResultat);
    mysql_close ($idconnexion);
  ?>
</body> </html>
```

2. Affichage de la table produit (avec lien sur la clef) : `AffichageProduitClef.php`

Construction de la ligne par :

```
echo '<p><a href="listePdt.php?categorie='.$ligne[0].'"target=\'page\'>'.$ligne[1].'</a>';
```



Le click sur ce lien permet l'ouverture de la page `listePdt.php` avec comme paramètre l'identifiant : `reference`.

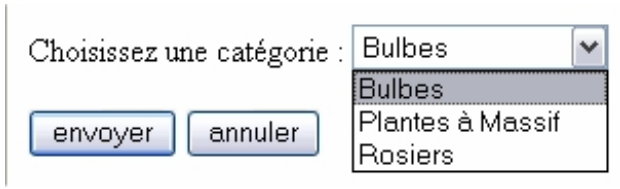
3. Affichage de la table produit (dans un tableau) : `AffichageProduitTableau.php`

```
  echo '<table width="95%" border="1">';
  while ($ligne = mysql_fetch_array ($jeuResultat)) {
    //Construction de la page HTML
    echo '<tr>';
      echo '<td align="center"><img src="/laFleur/images/'.$ligne[3].'.jpg"></td>';
      echo '<td>'.$ligne[0].'</td>';
      echo '<td>'.$ligne[1].'</td>';
      echo '<td align="right">'.$ligne[2].' €</td>';
    echo '</tr>';
  }
  echo '</table>';
```

4. Affichage de la table produit (dans une liste d'option) : `formSaisieProduitListe.php`

```
…
  Choisissez une catégorie :
    <FORM NAME="form5">
      <SELECT NAME="categorie">
    <?
```

```
    //Connexion au serveur
    …
    $requete = 'select * from categorie';
    $jeuResultat = mysql_query($requete);
    //Pour chaque ligne de jeuResultat (pris comme un tableau)
    while ($ligne = mysql_fetch_array ($jeuResultat)) {
      //Construction de ligne d'option
        echo '<OPTION VALUE="' .$ligne[0] . '">' .$ligne[1];
    }
    …
  ?>
  </SELECT>
```



Trame d'une requete LID (sans tests)

```
<?php
  // Connexion à la base de données
  mysql_connect(<hote>, <utilisateur>, <mot_de_passe>) or die("Connection Impossible");
  //Sélection de la base
  mysql_select_db(<base>) or die("Base inexistante");
  //Requête
  $requete = 'select * from …';
  //Récupération du résultat de la requête dans jeuResultat
  $jeuResultat = mysql_query($requete);
  //Pour chaque ligne de jeuResultat (pris comme un tableau)
  while ($ligne = mysql_fetch_array ($jeuResultat)) {
    //Construction du HTML
    echo $ligne[0] . '-' .$ligne[1];
  }
  //Fermeture des objets pour libération mémoire
  mysql_free_result($jeuResultat);
  mysql_close ($idconnexion);
?>
```

Liste non exhaustive de commandes disponibles

`mysql_connect()` `resource mysql_connect (string server,string user,string pwd,bool newlink,int cltflags)`	Ouvre une connexion à un serveur MySQL
`mysql_close()` `bool mysql_close (resource link_identifier)`	Ferme la connexion MySQL
`mysql_fetch_row()` `array mysql_fetch_row (resource result)`	Retourne une ligne de résultat MySQL sous la forme d'un tableau
`mysql_fetch_array()` `array mysql_fetch_array (resource result , int result_type)`	Retourne une ligne de résultat MySQL sous la forme d'un tableau associatif, d'un tableau indexé, ou les deux
`mysql_num_fields()` `Int mysql_num_fields (resource result)`	Retourne le nombre de champs d'un résultat MySQL
`mysql_db_name()` `string mysql_db_name (resource result , int row , mixed field)`	Lit les noms des bases de données

III.4- MySql - ODBC unifié

En plus du support de l'ODBC normal, l'ODBC unifié de PHP vous donne accès à diverses bases de données qui ont emprunté la sémantique des API ODBC pour implémenter leur propres API. Au lieu de maintenir de multiples pilotes qui sont similaires, ces pilotes ont été rassemblés dans un jeu de fonctions ODBC uniques.

Les bases de données suivantes sont supportées par l'ODBC unifié : IBM DB2, iODBC, Sybase SQL, …

Php supporte aussi l'utilisation d'une base de données générique : `PEAR::DB`.

Liste non exhaustive de commandes disponibles

`odbc_connect()`	Connexion à une source de données
`resource odbc_connect ( string dsn , string user , string password , int cursor_type )`	
`odbc_close()`	Ferme une connexion ODBC
`void odbc_close ( resource connection_id )`	
`odbc_fetch_row()`	Lit une ligne de résultat
`bool odbc_fetch_row ( resource result_id , int row_number )`	
`odbc_fetch_array()`	Lit une ligne de résultat dans un tableau associatif
`array odbc_fetch_array ( resource result , int rownumber )`	
`odbc_num_rows()`	Nombre de lignes dans un résultat
`Int odbc_num_rows ( resource result_id )`	

Et aussi :

`odbc_execute()`	Exécute une requête SQL préparée.
`bool odbc_execute ( resource result_id , array parameters_array )`	
`odbc_num_fields()`	Nombre de colonnes dans un résultat
`Int odbc_num_fields ( resource result_id )`	

III.4.a) Requête du LMD-LDD : Ajout, Suppression ou Modification

1. Conception de la page de récupération des informations : `ODBCFormSaisieProduit.php`

On utilisera : `FormSaisieProduit.php`, à laquelle il ne faudra pas oublier de modifier la méthode action.

2. Conception de la page d'ajout du produit : `ODBCajoutProduit.php`

```
<html> <head> <title>Ajout Produit</title> </head>
<body>
  <h2>Sté Lafleur</h2>
  <p><b>Nos produits</b></P>
  <?
    //Connexion au serveur
    $idconnexion=odbc_connect("CnxlaFleur","","");
    //Si la connexion est ok
    if ($idconnexion) {
      //Requête
      $requete = "insert into produit values ('".$_POST['reference']."','"
                          .$_POST['designation']."',".$_POST['prix'].",'".$_POST['image']
                                                    ."','".$_POST['categorie']."')";
      //exécution de la requête
      $Resultat = odbc_do($idconnexion,$requete);
      if ($Resultat) {
        echo 'Insertion OK !';
      } else {
        echo 'Insertion RATE !';
      }
    }
    //Fermeture des objets pour libération mémoire
    odbc_close ($idconnexion);
  ?>
<a href="ODBCformSaisieProduit.php"> retour </a> </body> </html>
```

III.4.b) Requête du LID : Affichage

1. Affichage de la table produit (sans présentation) : `AffichageProduit.php`

```
<html> <head> <title>menu</title> </head>
<body>
  <h2>Sté Lafleur</h2>
  <?
    //Connexion au serveur à la base lafleur via le middleware
    $idconnexion=odbc_connect("CnxlaFleur","","");
    //Si la connexion est ok
    if ($idconnexion) {
      //Requête
      $requete='select * from produit';
      //Récupération du résultat de la requête dans jeuResultat
      $jeuResultat=odbc_do($idconnexion,$requete);
        while (odbc_fetch_into($jeuResultat, $ligne)) {
          echo '<p>'.$ligne[0].' - '.$ligne[1];
        }
      }
      odbc_close($idconnexion);
    ?>
</body> </html>
```

IV/ CONTROLE DE SESSION

Le contrôle de session vise à permettre le suivi d'un utilisateur tout au long d'une session sur un site.

L'ouverture d'une session permettra d'afficher sélectivement le contenu d'un site (en fonction du niveau d'autorisation ou des préférences personnelles), l'implémentation de panier, …

IV.1- Vocabulaire

Une session est caractérisée par un numéro d'identification unique (nombre aléatoire crypté). Cet ID généré par PHP est enregistré côté client pendant toute la durée de la session.

Un ID joue le rôle d'une clé permettant d'enregistrer des variables particulières appellées « variables de session ». Le contenu de ces variables est conservé sur le serveur. L'ID est la seule information visible coté client et est accessible via un cookie ou via l'URL.

IV.2- Le cookie

Un cookie est une information enregistrée sur le disque du client, qui permet de mémoriser de l'information et de conserver cette information entre deux connexions.

Un cookie possède une durée de vie, c'est-à-dire que son effacement du disque est programmé dès qu'il y est inscrit. Par ailleurs, chaque cookie est la propriété d'un site qui est le seul a en avoir connaissance. En pratique, tous les cookies d'un site sont transmis au serveur à chaque requête du client.

Un cookie :

```
SetCookie : nom=valeur ; [expires=Date ;] [path=Chemin ;] [domain=Nom_Domaine ;] [secure]
```

ou

```
int setcookie (string nom [,string valeur [,int expiration [,string chemin [,string domain
[,int secure]]]]])
```

Ces en-têtes crées un cookie appelé `nom` avec la valeur `valeur`. Un cookie possède éventuellement une date d'expiration (`expires`), un lieu spécifique (`path` et `domain`), ainsi que la possibilité ou non de l'envoyer via une connexion sécurisée (`secure`).

Exemple : `test.php`

```
<?php
//Création du cookie validité permanente
setcookie ("MonCookie","antony");
//Visualisation du contenu du cookie, ne fonctionne qu'au chargement suivant
echo $_COOKIE['MonCookie'];
?>
```

Le cookie ne sera accessible que lorsque la page sera rechargé.

Un appel : `setcookie ("MonCookie","", <t-1>)` permet de supprimer le cookie.
Un appel : `setcookie ("MonCookie[]", "test", time()+3600)` permet d'avoir un cookie d'une heure.

Les cookies sont utilisé pour enregistrer l'ID de la session courante.

IV.3- L'URL

L'ID de session est enregistré dans la constante SID :

```
<a href="link.php?<?=SID?>">
```

IV.4- Contrôle de session simple

- ✓ **Démarrage d'une session** : avant d'exploiter les fonctionnalité relative aux sessions, nous devons ouvrir une session : `session_start ()`.

- ✓ **Enregistrement des variables de la session** : Les variables d'une session sont enregistrés dans le tableau `$_SESSION`.
- ✓ **Utilisation des variables de la session** : via `$_SESSION ['NomVariable']`.
- ✓ **Désenregistrement des variables et suppression de la session** : `unset` et `session_destroy ()`.

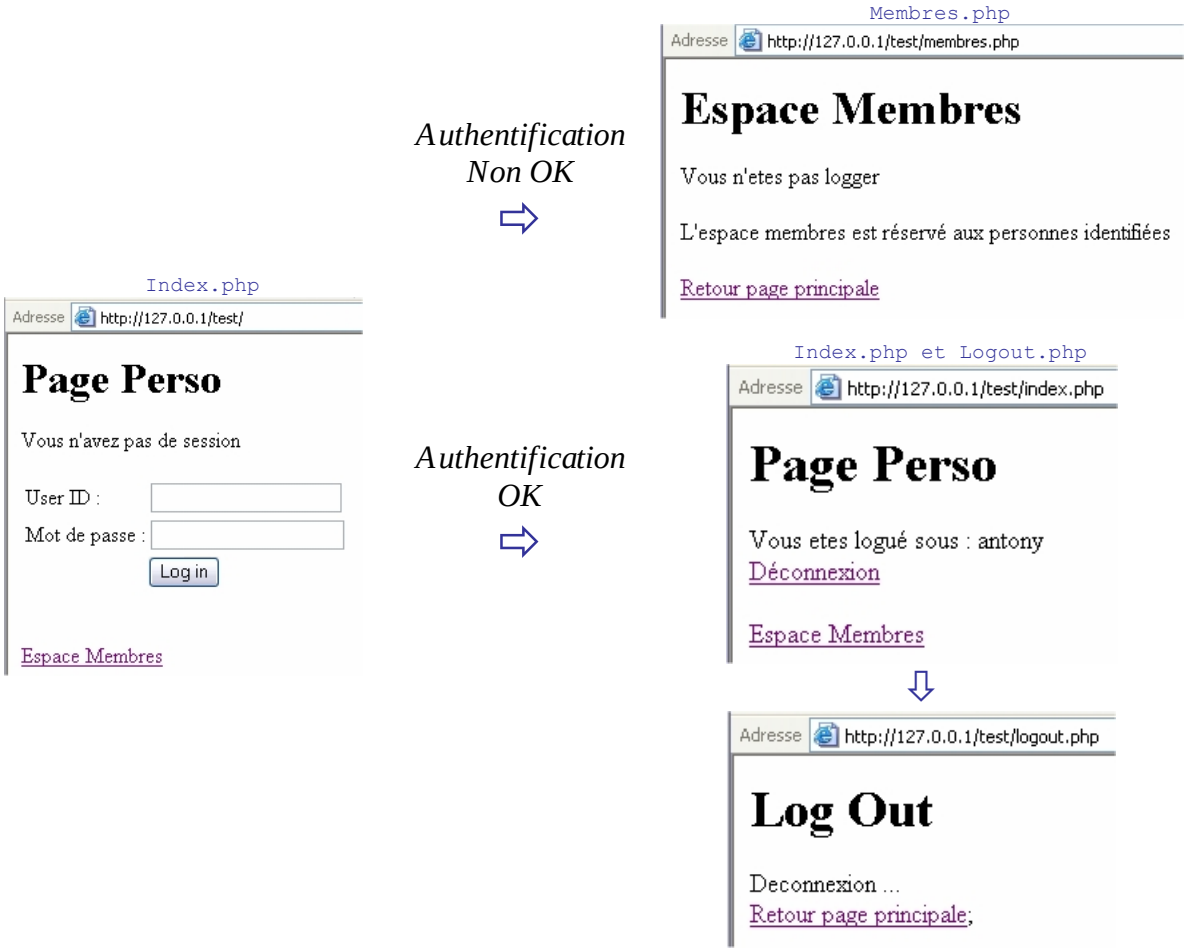
Exemple :

```
<?php
//ouvre une session et détermine un ID
session_start ();
//Crée et initialise une variable
$_SESSION['MaVar'] = 'coucou';
//Retourne le contenu de la variable si elle n est pas vide
if (! isset($_SESSION['MaVar'])) {
  echo "Variable Session : vide";
} else {
  echo "Variable Session : ".$_SESSION['MaVar'];
}
//Désenregistre les variables
unset ($_SESSION['MaVar']);
//supprime l'ID de la session
session_destroy();
?>
```

IV.5- Configuration du contrôle de session

Votre fichier `php.ini` contient un ensemble d'options de configuration à définir pour le contrôle de sessions.

IV.6- Contrôle de session avec authentification



- ✓ **Index.php**

```
<?php
//ouvre une session et détermine un ID
```

```
session_start ();
//Retourne le contenu de la variable si elle n est pas vide
if (isset($_POST['userid']) && isset ($_POST ['pwd'])) {
  //Si l'utilisateur a essayé d'ouvrir une session
  $userid = $_POST['userid'];
  $pwd = $_POST['pwd'];
  //Inclusion du fichier de connexion et connexion
  include ('Connexion_base.php');
  $idconnexion = BDD_Connect();
  if ($idconnexion) {
    //Requête sur base mysql du system avec variables php
    $requete = "select * from user where user = '$userid' and password = '$pwd'";
    $Result = mysql_query($requete);
    //S'il y a un résultat
    if (mysql_num_rows ($Result) > 0) {
      //Création de la variable session
      $_SESSION['validuser'] = $userid;
    }
  }
}
?>
<html><body><h1>Page Perso</h1>
<?
  if (isset ($_SESSION['validuser'])) {
    echo 'Vous etes logué sous : '.$_SESSION['validuser'].'<br/>';
    echo '<a href="logout.php"> Déconnexion </a><br/>';
  } else {
    if (isset ($userid)) {
      //Tentative d'ouverture avortée
      echo 'Vous ne pouvez vous logger';
    } else {
      //L'utilisateur n'a pas de session
      echo "Vous n'avez pas de session";
    }
    //Formulaire d'ouverture de session
    echo '<form method="post" action="index.php">';
    echo '<table>';
    echo '<tr><td>User ID :</td>';
    echo '<td><input type="text" name="userid"></td></tr>';
    echo '<tr><td>Mot de passe :</td>';
    echo '<td><input type="password" name="pwd"></td></tr>';
    echo '<tr><td colspan="2" align="center">';
    echo '<input type="submit" value="Log in"></td></tr>';
    echo '</table></form>';
  }
?>
<br><a href="membres.php"> Espace Membres</a></body></html>
```

✓ Membres.php

```
<?
session_start();
echo '<H1> Espace Membres </h1>';
//Vérification variable session
if (isset ($_SESSION['validuser'])) {
    echo '<p>Vous etes logué sous : '.$_SESSION['validuser'].'</p>';
    echo '<p> Bienvenue !</p>';
} else {
    echo "Vous n'etes pas logger <p>";
    echo "L'espace membres est réservé aux personnes identifiées<p>";
}
echo '<a href="index.php"> Retour page principale</a>';
?>
```

✓ Logout.php

```
<?php
session_start ();
$olduser = $_SESSION['validuser'];
unset ($_SESSION['validuser']);
session_destroy();
?>
<html>
<body>
  <h1>Log Out</h1>
  <?
  if (!empty ($olduser)) {
    echo 'Deconnexion ... <br/>';
  } else {
    //Si l'utilisateur n'avait ouvert de session mais qu'il était tout de même parvenu ici
```

```
    echo "Vous n'etiez pas connecté, mais vous etes déconnecté";
  }
  ?>
  <a href="index.php"> Retour page principale</a>;
</body>
</html>
```

IV.7- Affichage sélectif

Mettre l'ID de l'utilisateur dans la variable session permet notamment de faire de l'affichage sélectif :

✓ **Membres.php**

```
include ('Connexion_base.php');
$idconnexion = BDD_Connect();
if ($idconnexion) {
  //Requête sur base mysql du system avec variables php
  $requete = "select * from user";
  $Result = mysql_query($requete);
  echo "<table border=1>";
  echo "<tr><td>User ID</td><td>Localhost</td><td>Nb Connections*</td>";
  while ($ligne = mysql_fetch_array ($Result)) {
    //Informations publiques destinées à tous
    $lg = "<tr><td>".$ligne[1]."</td><td>".$ligne[0]."</td><td>";
    //Informations privées à sélectionner
    if (strcmp ($ligne[1],$_SESSION['validuser']) == 0) {
      //Informations personnelles à n'afficher que pour l'utilisateur
      $lg = $lg.$ligne[30];
    } else {
      //Sinon mettre « confidentiel »
      $lg = $lg."Confidentiel";
    }
    $lg = $lg."</td>";
    echo $lg;
  }
  echo "</table>";
}
```

IV.8- Un peu de sécurité …

Attention, le code proposé ci-dessus est « sensible » à `l'injection SQL`. Il est nécessaire d'apporter des modifications de sécurité, telles que :

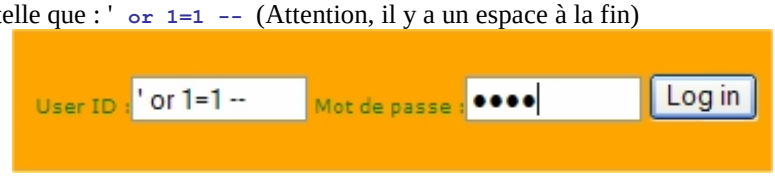
- tester plus précisément les champs SQL retournés et notamment leurs contenus,
- ou tester le contenu des zones de saisies par des expréssions régulières.

✓ **L'injection SQL**

Le problème vient de notre requête `SQL`, cette dernière ressemble à ceci :

```
$requete = "select * from client where clt_code = '$userid' and clt_motPasse = '$pwd'";
```

Un utilisateur malveillant et connaissant le `SQL`, insèrera à la place de la variable `$userid` (via l'interface), une ligne de code SQL telle que : `' or 1=1 --` (Attention, il y a un espace à la fin)



Cette instruction SQL va venir modifier notre requête SQL comme ceci :

```
select * from client where clt_code = '' or 1=1 -- $userid' and clt_motPasse = '$pwd'";
```

A noter que les `--` constituent des commentaires et donc notre requête devient :

```
select * from client where clt_code = '' or 1=1
```

Bien sûr, il ne trouvera personne via la selection : `clt_code = ''`, mais l'instruction : `or 1=1` étant toujours vrai, elle lui permettra de retourner l'ensemble des lignes de la table.

Il pourra ainsi :

- soit les afficher,
- soit détouner notre code pour entrer :

```
//Récupération du résultat de la requête dans jeuResultat
$Result = mysql_query($requete);
//Si le nombre de lignes retournées est supérieur à 0
if (mysql_num_rows ($Result) > 0) {
```

LES BALISES META ET LE REFERENCEMENT

Véritable "sac à malice" du concepteur de sites Web, ces mystérieuses balises META feront de votre site une gentille œuvre d'amateur ou lui donneront la signature d'un site de "pro".

Conformément à la philosophie du Html, les éléments relatifs au document sont dissociés du corps du document lui-même et repris dans la zone d'en-tête (c-à-d entre les balises <HEAD> et </HEAD>). Les balises META apportent une série d'informations relatives à la teneur de votre page Web et sont donc toujours comprises entre les balises <HEAD>.

Les informations ainsi apportées intéressent :

- **principalement le référencement de votre site et les moteurs de recherche**.
- un peu vos lecteurs (avertis bien entendu).
- et accessoirement le browser ou le navigateur.

Quand on apprend que plus de la moitié des utilisateurs du Web passent par des moteurs de recherche, il importe de soigner le référencement de son site pour y apparaître de façon correcte et si possible en rang utile.

I/ LES INDISPENSABLES

<META NAME="keywords" CONTENT="mot-clé1,mot-clé2,mot-clé3,...">

Tous [ou presque] les moteurs de recherche l'utilisent.

Quelques commentaires s'imposent cependant :

- *Les mots-clés doivent être séparés par une virgule*.
- L'auteur préconise de ne pas mettre d'espaces, et de mettre tous les mots clés en miniscules.
- On parle dans la littérature d'une limitation à 1000 mots-clés.
- La tentation est grande de répéter un certain nombre de fois un même mot-clé pour espérer un meilleur classement, mais ce truc a vécu et est maintenant pénalisé [spam] par les moteurs de recherche.

Tout l'art consiste à trouver les bons mots-clés relatifs au contenu de votre site. Mettez-vous à la place de vos lecteurs potentiels. Quels mots, quels synonymes, quelles alternatives peuvent être utilisés pour décrire votre site?

<META NAME="description" CONTENT="une brève description de la page">

Cette description sera pertinente, attirante et brève.
En effet, selon les moteurs de recherche seuls les 150 à 240 premiers mots seront repris.

<TITLE> ... </TITLE>

Les moteurs de recherche tiennent fortement compte des titres des documents. Il faut impérativement en mettre sur toutes les pages d'un site. Même pour les pages qui n'apparaissent pas directement comme les sous-pages d'une page de frames !

On dit également que le fait de reprendre un ou des mot(s)-clé(s) dans le titre de toutes les pages d'un site est très favorable pour un meilleur classement.

II/ LES UTILES

<META NAME="author" CONTENT="nom de l'auteur">

Cette balise est d'une utilité discutable car rares sont les moteurs de recherche en tiennent compte. Mais il n'empêche qu'il est d'une légitime fierté de signer son œuvre. N'est-il pas ?

<META NAME="Copyright" CONTENT="Copyright © date nom">

Nous ne discuterons pas de discuter ici de la valeur juridique et pratique d'une mention de copyright sur le Web. Mais une règle élémentaire de la Netiquette est de respecter le copyright.

"Le gratuit n'est pas forcément sans valeur".

Et aussi

```
<META NAME="Distribution" CONTENT="Global ou Local">
```

Cette balise indique la destination de l'information de la page. Soit qu'elle est "Global" et donc destinée à être largement diffusée soit qu'elle est "Local" et donc à diffusion restreinte.

```
<META NAME="Rating" CONTENT="Destination de votre audience">
```

Permet de définir le contenu de votre site. Les appréciations sont General ou Mature ou Restricted ou 14 years pour respectivement tout public, adulte, accès restreint ou 14 ans.

```
<META NAME="Robots" CONTENT="instructions pour les robots">
```

Par cette balise vous pouvez indiquez aux robots de recherche automatique si vous souhaitez que votre site soit ou ne soit pas indexé par eux.

Les instructions sont :

- `All` (défaut) permet aux robots d'indexer vos pages et de suivre les liens hypertextes d'une page à l'autre.
- `None` dira aux robots de ne pas indexer vos pages et de ne pas suivre les liens.
- `Index` indique que vos pages peuvent être indexées par les robots.
- `NoIndex` pour que le robots ne procèdent à aucune indexation.
- `Follow` donne la permission aux robots de suivre les liens hypertextes des pages.
- `NoFollow` pour le contraire.

```
<META HTTP-EQUIV="Content-language" CONTENT="fr">
```

Cette balise déclare la langue utilisée dans le document Html.

```
<META HTTP-EQUIV="Reply-to" CONTENT="votre adresse e-mail">
```

Cette balise permet au lecteur averti de connaître votre adresse e-mail si elle n'apparaît pas sur la page qu'il consulte.

```
<META HTTP-EQUIV="Reply-to" CONTENT="URL de votre page d'accueil">

<META NAME="Rating" CONTENT="Destination de votre audience">
```

III/ LES EVENTUELLES

```
<META HTTP-EQUIV="Refresh" CONTENT="x;URL="adresse">
```

Cette balise est fréquemment utilisée pour rediriger automatiquement un visiteur dans le cas où l'adresse de votre site à été modifiée.

```
<META HTTP-EQUIV="Page-Enter" content="revealTrans(Duration=1.0,Transition=23)">
<META HTTP-EQUIV="Page-Exit" content="revealTrans(Duration=1.0,Transition=23)">
```

De très jolis effets de transition style PowerPoint sont possibles avec simplement une ligne de code. Mais autant vous le dire tout de suite, cela ne fonctionne que sous Explorer 4 et plus.

Quelques explications :

- `Page-Enter` et `Page-Exit` signifie que l'effet de transition se produira à l'entrée de la page ou à la sortie de celle-ci.
- `Duration` détermine la durée de la transition en secondes. Elle est dans l'exemple de 1 seconde.
- `Transition` est un nombre de 1 à 23 pour l'effet de transition choisi. Le chiffre 23 donne une transition aléatoire.

```
<META HTTP-EQUIV="expires" CONTENT="Wed, 22 Dec 2004 19:00:02 GMT">
<META HTTP-EQUIV="expires" CONTENT="0">
```

Cette balise `META` "dit" au navigateur la date à laquelle la page Html doit être considérée comme périmée.

Il faut noter que les robots de recherche peuvent retirer ces pages dites périmées de leur base de donnée.

```
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
```

C'est une autre façon de contrôler le cache du navigateur. Avec ce META, vous pouvez demander au browser de ne pas tenir la page dans le cache.

IV/ TRUCS ET ASTUCES DE REFERENCEMENT

Alors que les balises META font partie du Html, il existe des tas de trucs et astuces - plus ou moins vérifiables ou vérifiés - qui sont sensés améliorer le rang de votre site parmi les centaines renvoyés lors d'une recherche par un mot-clé.

IV.1- Référencez-vous

Cela peut paraître une évidence mais si vous attendez qu'un robot de recherche vienne visiter votre site, vous risquez de devoir attendre longtemps...

Sur la page d'accueil de chaque moteur de recherche, vous trouverez "Ajouter un site", "Référencement" ou quelque chose du même acabit. En cliquant sur le lien, vous ouvrirez un formulaire, plus ou moins long, à remplir pour référencer directement votre site.

N'oubliez pas que certains moteurs de recherche style annuaire (comme Yahoo) se réservent de droit de reprendre ou de ne pas reprendre votre site, qu'il faut compter généralement 2 à 3 semaines avant d'apparaître dans la base de donnée et un bon mois avant que les internautes commencent à s'intéresser à votre site.

IV.2- De l'importance des mots-clés

Pour obtenir un bon classement, il faut non seulement définir ces mots-clés dans la balise <META NAME="keywords... , mais il est aussi recommandé :

- que le ou les mots-clés soi(en)t repris dans le premier paragraphe de la page.
- que le ou les mots-clés soi(en)t repris dans le titre de la page (<TITLE>).
- que le ou les mots-clés revienne(nt) plus fréquemment que les autres mots de la page.

IV.3- La page d'accueil en frames

Les frames ne sont que modérément appréciés par les moteurs de recherche qui ne les tiennent pas ou difficilement en compte, car les moteurs qui ne suivent pas les liens dans les frames se rabattent sur le contenu des balises <NOFRAMES> ... </NOFRAMES>.

Ne pas oublier de mettre les balises META !

IV.4- La page d'accueil avec une image ou une image mapée

Cette situation est assez inconfortable pour le robot à la recherche de mots-clés car il n'a alors aucun texte (et donc de mots) pour faire son référencement...

A éviter …

IV.5- Et encore (A voir)

Sont ou seraient bénéfiques pour un bon référencement :
- le fait que votre site soit référencé par d'autres sites.
- le fait que votre site soit mis à jour régulièrement.

IV.6- Désolé...

Désolé, mais les trucs suivants ne fonctionnent plus et seraient même pénalisés [spam] par les moteurs de recherche :

- texte invisible dans la même couleur que le fond de la page.
- texte repris en commentaire (balises <!-- ... -->).
- les mêmes mot-clé repris indéfiniment dans la balise <META NAME="keywords" ....>.
- le texte dans l'élément de formulaire <INPUT type="hidden"...> (éléments cachés).

FTP & PUBLICATION

I/ LES HEBERGEURS

Pour « mettre en ligne » un site Internet et permettre au monde de découvrir votre travail, vous devez placer ce dernier sur une machine à laquelle tout le monde aura accès.

Deux possibilités vous sont offertes :

- Soit vous disposez d'une adresse IP. Ex : les universités, les grands groupes, etc …
- Soit vous en « louer » une à un fournisseur d'accès.

Les services des fournisseurs d'accès ont explosés ces dernières années. Ils vous permettent notamment :

- d'avoir un ou plusieurs emails,
- d'avoir un espace web,
- de tchater,
- …

Démarche

1. Pour un premier site, choissisez un fournisseur d'accès gratuit de préférence (ex : free).

2. Demamdez lui (sur son site : www.free.fr), un accès gratuit à Internet.

Cet accès vous donnera droits à de nombreux services et notamment un espace web.

3. Vous recevrez alors des paramètres :
   - Un nom d'utilisateur,
   - Un mot de passe,
   - Des paramètres tels que :

Serveur POP3	pop.free.fr	DNS primaire	212.27.32.5
Serveur SMTP	smtp.free.fr	DNS secondaire	213.228.0.168
Serveur de News	news.free.fr	Domaine	free.fr
Proxy http/FTP	proxy.free.fr (port 3128)	Point d'accès	

Ceux sont ces paramètres qui vous permettront d'accéder à un répertoire spécifique (voir ci-dessous) dans lequel vous n'aurez plus qu'à déposer votre site.

4. Vous aurez peut-être besoin d'activer votre site, la base de données, … (lire les conditions spécifiques à chaque hébergeurs). Ce qui vous permettra d'obtenir le nom du site :
   http://<nom_utilisateur>.free.fr

Remarque sur les bases de données :
Les hébergeurs proposent des interfaces de gestion de bases de données. (renseignez vous)

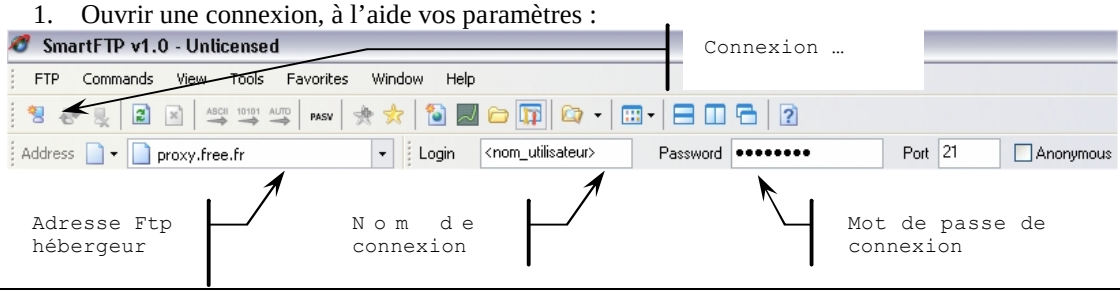
5. Déplacez vos fichiers via un logiciel FTP (File Transfert Protocol) **et admirez !**

6. N'oubliez pas les options : référencement, nom de domaine, …

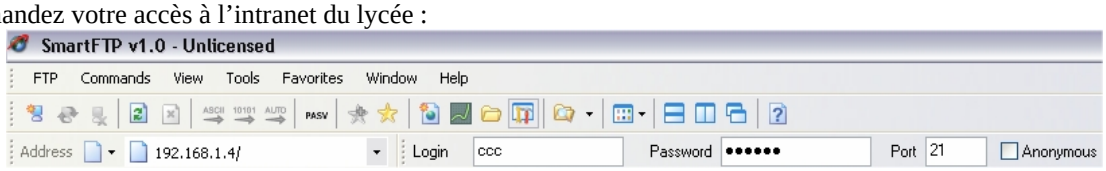
II/ SMARTFTP

SmartFtp est un logiciel de transfert de fichier, vous devez :

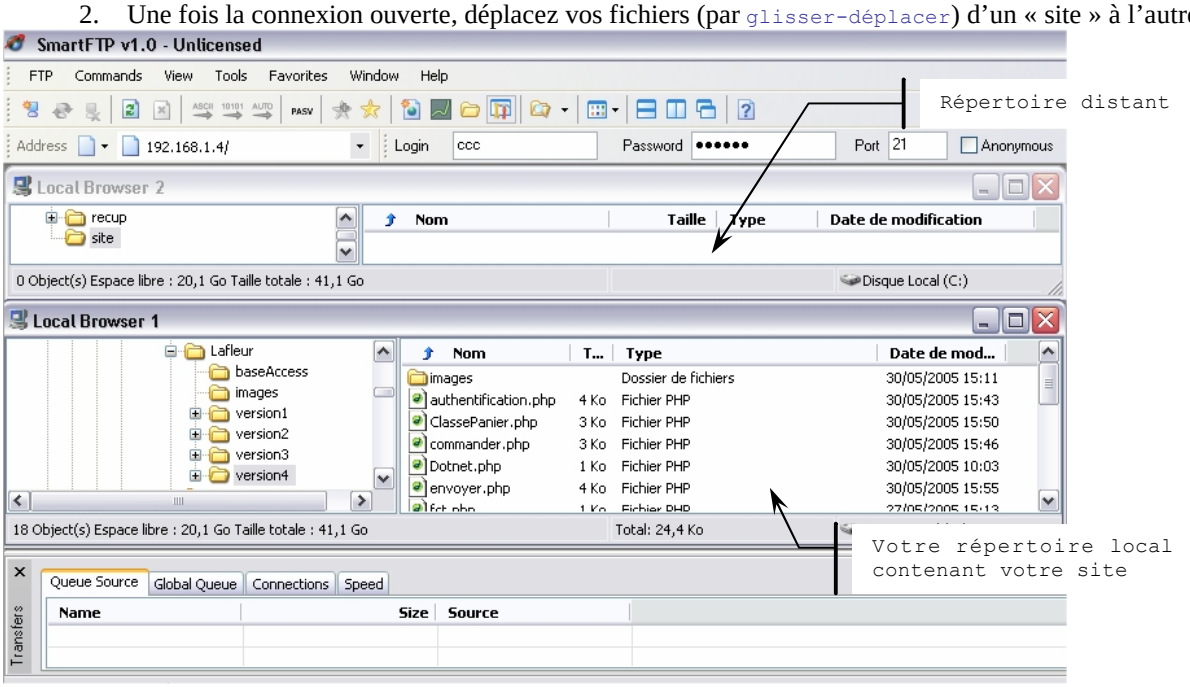
1. Ouvrir une connexion, à l'aide vos paramètres :



Demandez votre accès à l'intranet du lycée :



2. Une fois la connexion ouverte, déplacez vos fichiers (par `glisser-déplacer`) d'un « site » à l'autre.



3. Pensez à bien refermer la connexion.

ANNEXE 1
LISTE DES PRINCIPALES BALISES HTML

Mise en forme des caractères

`<B>...</B>`	Texte en gras
`<BIG>...</BIG>`	Agrandissement de la taille des caractères
`<BLINK>...</BLINK>`	Texte clignotant (Netscape seul)
`<EM>...</EM>`	Texte en italique
`<FONT color="#XXXXXX">...</FONT>`	Texte en couleur où XXXXXX est une valeur hexadécimale
`<FONT size=X>...</FONT>`	Taille des caractères où X est une valeur de 1 à 7
`<I>...</I>`	Texte en italique
`<NOBR>...</NOBR>`	Empêche les ruptures automatiques de ligne des navigateurs
`<PRE>...</PRE>`	Texte pré formaté (affichage de tous les espaces et sauts de ligne)
`<SMALL>...</SMALL>`	Réduction de la taille des caractères
`<STRONG>...</STRONG>`	Mise en gras du texte
`<SUB>...</SUB>`	Texte en indice
`<SUP>...</SUP>`	Texte en exposant
`<U>...</U>`	Texte souligné

Mise en forme du texte

`<!--...-->`	Commentaire ignoré par le navigateur
` `	A la ligne
`<BLOCKQUOTE>...</BLOCKQUOTE>`	Citation (introduit un retrait du texte)
`<CENTER>...</CENTER>`	Centre tout élément compris dans le tag
`<DIV align=center> ...</DIV>`	Centre l'élément encadré par le tag
`<DIV align=left> ...</DIV>`	Aligne l'élément à gauche
`<DIV align=right> ...</DIV>`	Aligne l'élément à droite
`<Hx>...</Hx>`	Titre où x a une valeur de 1 à 7
`<Hx align=center>...</Hx>`	Titre centré
`<Hx align=left>...</Hx>`	Titre aligné à gauche
`<Hx align=right>...</Hx>`	Titre aligné à droite
`<P>...</P>`	Nouveau paragraphe
`<P align=center>...</P>`	Paragraphe centré
`<P align=left>...</P>`	Paragraphe aligné à gauche
`<P align=right>...</P>`	Paragraphe aligné à droite

Listes

`<UL>`
`<LI>`
`</UL>`
	Liste non numérotée (dite à puces)
	Elément de liste

`<OL>`
`<LI>`
`</OL>`
	Liste numérotée
	Elément de liste

`<DL>`
`<DT>...</DT>`
`<DD>...</DD>`
`</DL>`
	Liste de glossaire
	Terme de glossaire (sans retrait)
	Explication du terme (avec retrait)

`<HR>`	Ligne de séparation Trait horizontal (centré par défaut)
`<HR width="x%">`	Largeur du trait en %
`<HR width=x>`	Largeur du trait en pixels
`<HR size=x>`	Hauteur du trait en pixels
`<HR align=center>`	Trait centré (défaut)
`<HR align=left>`	Trait aligné à gauche
`<HR align=right>`	Trait aligné à droite
`<HR noshade>`	Trait sans effet d'ombrage

Hyperliens

`<A href="http://...">...</A>`	Lien vers une page Web
`<A href="mailto:...">...</A>`	Lien vers une adresse Email
`<A href="fichier.htm">...</A>`	Lien vers la page locale fichier.htm située dans le même dossier
`<A name="xyz">...</A>`	Définition d'une ancre
`<A href="xyz">...</A>` `<A href="fichier#xyz">...</A>`	Lien vers une ancre

Images

`<IMG src="xyz.gif">` `<IMG src="xyz.pjg>`	Insertion d'une image au format Gif ou Jpg (voir liens pour l'adressage)
`<IMG ... width=x height=y>`	Mise à l'échelle de l'image en pixels
`< IMG ... border=x>`	Définition de la bordure d'une image avec lien
`<IMG ... alt="votre texte">`	Texte alternatif lorsque l'image n'est pas affichée
`<IMG ... align=bottom>`	Aligne l'image en bas
`<IMG ... align=middle>`	Aligne l'image au milieu
`<IMG ... align=top>`	Aligne l'image en haut
`<IMG ... align=left>`	Aligne l'image à gauche
`<IMG ... align=right>`	Aligne l'image à droite
`<IMG ... hspace=x>`	Espacement horizontal entre l'image et le texte
`<IMG ... vspace=y>`	Espacement vertical entre l'image et le texte

Tableau

`<TABLE>...</TABLE>`	Définition d'un tableau
`<TABLE width="x%">`	Largeur du tableau en %
`< TABLE width=x>`	Largeur du tableau en pixels
`<TABLE border=x>`	Largeur de la bordure
`<TABLE cellpadding=x>`	Espace entre la bordure et le texte
`<TABLE cellspacing=x>`	Epaisseur du trait entre les cellules
`<TR>...</TR>`	Ligne du tableau
`<TD>...</TD>`	Cellule du tableau
`<TD bgcolor="#XXXXXX">`	Couleur d'une cellule de tableau
`<TD width="x%">`	Largeur de colonne en %
`<TD width=x>`	Largeur de colonne en pixels
`<TD align=center>`	Texte dans la cellule centré
`<TD align=left>`	Texte dans la cellule aligné à gauche
`<TD align=right>`	Texte dans la cellule aligné à droite
`<TD valign=bottom>`	Alignement vers le bas du contenu d'une cellule
`<TD valign=middle>`	Centrage vertical du contenu d'une cellule
`<TD valign=top>`	Alignement vers le haut du contenu d'une cellule
`<TD colspan=x>`	Nombre de cellules à fusionner horizontalement
`<TD rowspan=x>`	Nombre de cellules à fusionner verticalement

Frames

`<framset>...</framset>`	Définit une structure de frames (remplace alors le tag `BODY`)
`<framset rows="x%,y%,...">`	Division horizontale de la fenêtre en %
`<framset cols="x%,y%,...">`	Division verticale de la fenêtre en %
`<FRAME src="fichier.htm">`	Fichier affiché dans une fenêtre de frames
`<NOFRAMES>...</NOFRAMES>`	Contenu pour les browsers non prévus pour les frames

Fichier Html

`<HTML>...</HTML>`	Début et fin de fichier Html
`<HEAD>...</HEAD>`	Zone d'en-tête d'un fichier Html
`<TITLE>...</TITLE>`	Titre affiché par le browser (élément de `HEAD`)
`<BODY>...</BODY>`	Début et fin du corps du fichier Html
`<BODY bgcolor="#XXXXXX">`	Couleur d'arrière-plan (en hexadécimal)
`<BODY background="xyz.gif">`	Image d'arrière-plan

ANNEXE 2
LISTE DES PROPRIETES DES FEUILLES DE STYLES

La liste complète (et officielle) des propriétés et recommandations concernant les feuilles de style version 1.
http://www.w3.org/pub/WWW/TR/REC-CSS1

En voici une sélection :

1- Les styles de police

`font-family`
définit un nom de police ou une famille de police `<nom>` ou `<famille>`
police précise (Arial, Times, Helvetica...) ou famille (serif, sans-serif, cursive, fantasy, monospace)
`ex : H3 {font-family: Arial}`

`font-style`
définit le style de l'écriture : normal ou italique ou oblique
`ex : H3 {font-style: italic}`

`font-weight`
définit l'épaisseur de la police : normal, bold, bolder, lighter ou valeur numérique (100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900)
`ex : P {font-weight: bold}`

`font-size`
définit la taille de la police xx-small, x-small, small, médium, large, x-large, xx-large, larger, smaller
ou taille précise en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
`ex : P {font-size: 12pt}`

`font-variant`
définit une variante par rapport à la normale : normal ou small-caps
`ex: P {font-variant: small-caps}`

`font`
raccourci pour les différentes propriétés de police
`ex : P {font: bold italic}`

2- Les styles du texte

`text-align`
définit l'alignement du texte : left, center ou right
`ex : H1 {text-align: center}`

`text-indent`
définit un retrait dans la première ligne d'un bloc de texte souvent utilisé avec `<P>`
spécifié en inches (in), en centimètres (cm) ou en pixels (px)
`ex : P {text-indent: 1cm}`

`text-decoration`
définit une décoration (?) du texte, soit barré, clignotant, blink, underline, line-through, overline ou none
`ex : A:visited {text-decoration: blink}`

`text-transform`
définit la casse du texte : uppercase (majuscule), lowercase (minuscule) ou capitalize (premier caractère en majuscule)
`ex : P {text-transform: uppercase}`

`color`
définit la couleur du texte
`ex : H3 {color: #000080}`

`word-spacing`
définit l'espace entre les mots en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
`ex : P {word-spacing: 5pt}`

letter-spacing
définit l'espace entre les lettres spécifié en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
```
ex : P {letter-spacing: 2pt}
```

line-height
définit l'interligne soit l'espace entre les lignes du texte en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
```
ex : P {line-height: 10pt}
```

width
détermine la longueur d'un élément de texte ou d'une image en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
```
ex : H1 {width: 200px}
```

height
détermine la hauteur d'un élément de texte ou d'une image en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
```
ex : H1 {heigh: 100px}
```

white-space
espace ou blanc : normal ou pre ou nowrap
```
ex : PRE {white-space: pre}
```

3- Les arrière-plans

background-color
définit la couleur de l'arrière-plan : couleur (par exemple en hexadécimal) ou transparent
```
ex: H1 {background-color: #000000}
```

background-image
définit l'image de l'arrière-plan : URL de l'image
```
ex : BODY {background-image: image.gif}
```

background-repeat
définit la façon de répéter l'image d'arrière-plan : repeat ou no-repeat ou repeat-x (x = nombre de répétitions horizontales) ou repeat-y (y = nombre de répétitions verticales
```
ex : P {background-image: image.gif; background-repeat: repeat-4}
```

background-attachment
spécifie si l'image d'arrière-plan reste fixe avec les déplacements de l'écran : scroll ou fixed
```
ex : BODY {background-image: image.gif; background-attachement: fixed}
```

background-position
spécifie la position de l'image d'arrière-plan par rapport au coin supérieur gauche de la fenêtre {1, 2}, {top ou center ou bottom , left ou center ou right}, ou en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
```
ex: BODY {background-image: img.gif; background-position: right top}
```

background
raccourci pour les différentes propriétés d'arrière-plan
```
ex : P {background: image.gif fixed repeat}
```

4- Les marges

margin-top
détermine la valeur de la marge supérieure : en unité de longueur ou auto
```
ex : { margin-top: 5px }
```

margin-right
détermine la valeur de la marge droite : en unité de longueur ou auto
```
ex : { margin-right: 5px }
```

margin-bottom
détermine la valeur de la marge inférieure : en unité de longueur ou auto
```
ex : { margin-bottom: 5px }
```

margin-left
détermine la valeur de la marge gauche : en unité de longueur ou auto
ex : { margin-left: 5px }

margin
regroupe les différentes propriétés de la marge

5- Les bords et les "enrobages"

border-top-width
donne l'épaisseur du bord supérieur : thin, medium, thick, spécifié par l'auteur
ex : H3 {border-top-width: thin}

border-right-width
donne l'épaisseur du bord droit : thin, medium, thick, spécifié par l'auteur
ex : H3 {border-right-width: medium}

border-bottom-width
donne l'épaisseur du bord inférieur : thin, medium, thick, spécifié par l'auteur
ex : H3 {border-bottom-width: thick}

border-left-width
donne l'épaisseur du bord gauche : thin, medium, thick, spécifié par l'auteur
ex : H3 {border-left-width: 0.5cm}

border-width
regroupe les différentes propriétés de border-width

border-color
détermine la couleur de la bordure
ex : H3 {border-color: yellow}

border-style
détermine le style du trait de la bordure : none, solid, dotted, dashed, double, groove, ridge, inset ou outset
ex : {border-style: solid dashed}

border
regroupe toutes les propriétés des bords

padding-top
valeur de remplissage haut entre l'élément et le bord : en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
ex : H3 {padding-top: 3px}

padding-right
valeur de remplissage droite entre l'élément et le bord : en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
ex : H3 {padding-right: 3px}

padding-bottom
valeur de remplissage bas entre l'élément et le bord : en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
ex : H3 {padding-bottom: 3px}

padding-left
valeur de remplissage gauche entre l'élément et le bord : en points (pt), inches (in), centimètres (cm), pixels (px) ou pourcentage (%)
ex : H3 {padding-left: 3px}

padding
regroupe les différentes propriétés de remplissage

6- Les listes

list-style-type

détermine le type de puces ou de numérotation : disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha ou upper-alpha

```
ex : OL {list-style-type: square}
```

`list-style-image`

permet de remplacer les puces par une image : url ou none

```
ex : OL {list-style-image: image.gif}
```

`list-style-position`

spécifie si les puces sont à l'intérieur ou à l'extérieur du texte : inside ou outside

```
ex : UL {list-style-position: inside}
```

`list-style`

regroupe toutes les propriétés de liste

`<ol type=i>`	numérotation type : i,ii,iii
`<ol type=I>`	numérotation type : I,II,III
`<ol type=a>`	numérotation type : a,b,c
`<ol type=A>`	numérotation type : A,B,C

ex :

```
ol {
  font-size: 24px;
  list-style-type : lower-alpha ;
}
```

et second niveau

```
ol ol {
  list-style-type : circle ;
}
```

7- Tableaux

`border-collapse`

Fusion des bordures des cellules (collapse) ou non (separate)

`border-spacing`

Espacement des cellules

`caption-side top, bottom, left ou right`

Positionnement de la légende du tableau

`empty-cells show ou collapse`

Affichage (show) ou masquage (collapse) des cellules vides

`table-layout fixed (indépendamment du contenu des cellules) ou auto (selon le contenu des cellules)`

Largeur fixe ou variable

`speak-headers alwats (systématiquement avant chaque cellule) ou once (une seule fois)`

Propriété pour sourds et malentendants indiquant le comportement lors de la lecture des cellules d'en-tête d'un tableau

8- Mise en page

`@page (size: < auto, landscape ou portrait >)`

Définit la mise en page de l'impression. Peut se décliner en `@page :left {…}` (marges pour pages de gacuhe), `@page :right {…}` (marges pour pages de droite) et `@page :first {…}` (marges pour première page).

Ex :

```
<style>
  @page {size :landscape ;}
</style>
```

ou

```
<style>
  @page {size :21.0cm 29.7cm ;}
</style>
```

ou

```
@page {
  size :21.0cm 29.7cm ;
  margin-top :2cm ;      /* marge du haut*/
  margin-bottom :2cm ;   /* marge du bas*/
  margin-left :2cm ;     /* marge de gauche*/
  margin-right :2cm ;    /* marge de droite*/
}
```

ou

```
@page {
  size :21.0cm 29.7cm ;
  margin :2cm ;          /* les 4 marges*/
}
```

`margin-top <valeur>`
Marge supérieure

`margin-right <valeur>`
Marge de droite

`margin-bottom <valeur>`
Marge inférieure

`margin-left <valeur>`
Marge de gauche

`marks <crop (traits de coupe), cross (repère de montage), none (pas de marque)>`
Traits de coupe et repères de montage

`page-break-before <Always, avoid Force>`
Le saut de page avant un élément

`avoid`	: ne jamais insérer de saut de page avant/après
`always`	: toujours insérer de saut de page avant/après
`left`	: insérer de saut de page, élément actuel/suivant sur la prochaine page de gauche
`right`	: insérer de saut de page, élément actuel/suivant sur la prochaine page de droite
`inherit`	: prendre la mention saut de page de l'élément parent
`auto`	: pas de mention de saut de page (par défaut)

Ex :

```
img {
  page-break-before : avoid ;
}
```

`page-break-after <Always, avoid Force>`
Le saut de page après un élément

`orphans <valeur>`
Evite les lignes orphelines en fin de page. Définit le nombre de ligne minimal à partir un renvoi en page suivante est effectué.

`widows <valeur>`
Evite les lignes veuves en début de page. Définit le nombre de ligne minimal à partir un renvoi en page précédente est effectué

8- Média de sortie

Les feuilles de style se déclarent à l'aide de la balise `<link>`

```
<link rel= "stylesheet" media="screen" href="style.css">
```

`Media="all"`	la feuille CSS s'appliquent à tous les medias
`Media="aural"`	la feuille CSS s'appliquent à des systèmes de restitution vocale assisté par ordinateur
`Media="braille"`	la feuille CSS s'appliquent à des périphériques de sortie comportant une « ligne Braille »
`Media="embossed"`	la feuille CSS s'appliquent à des imprimantes en braille
`Media="handheld"`	la feuille CSS s'appliquent à l'affichage sur des ordinateurs de poche
`Media="screen"`	la feuille CSS s'appliquent à l'affichage à l'écran
`Media="tty"`	la feuille CSS s'appliquent à des médias de sortie non graphiques

`Media="print"` la feuille CSS s'appliquent à l'impression sur papier