

Conception et développement web

Filière ID S2

Objectifs du cours:

Comprendre et connaître les principes des technologies utilisées dans le développement d'applications liées au web.

Programme général:

Programmation en HTML, CSS, JavaScript, PHP et MySQL

- la structure d'un fichier HTML et ses notions de base: Les listes, styles de caractères, les liens, les images cliquables, les tableaux, les cadres, les formulaires...
- les Avantages du CSS et ses règles : syntaxe, type, Sélecteur (id, classe, Pseudo-classes, ...), notion d'héritage et de cascade....
- les limites d' HTML
- Application interactive avec du Javascript: le DOM
- le modèle client/serveur
- les pages statiques / dynamiques
- PHP : Historique et version, Fonctionnalités, le fonctionnement, Syntaxe, types de données, Variables prédéfinies, Structures de contrôle, Boucles, Fonctions, Fonctions sur les tableaux (each() et list() ...), Gestion des fichiers, fonction explode()... expressions régulières en PHP, gestion de sessions et les cookies
- Interfaçage avec les bases de données MySQL

Conception et développement web

Filière ID S2

Dates importantes:

- De la semaine du 27 Janvier à la semaine du 24 Février: Cours.
- La semaine du 24 Février: Lancement des travaux pratiques.
- **La semaine du 27 Avril: Examen Cours et TP**



Oumayma BANOUAR
oumayma.banouar@gmail.com

Le serveur de fichier représente juste l'espace de stockage (ou d'hébergement), il contient l'ensemble des documents constitutants votre site web.

Requête (URL, Lien...)

Le serveur d'application analyse d'abord le fichier.

Si celui là contient des scripts côté serveur (dits CGI) ils seront d'abord exécutés.

- ✓ Le serveur de base de données renferme la/les base de données té
- ✓ qui alimente le site Web en contenu. Cette base de données est it
- ✓ interrogée par les scripts CGI.

Serveur
WEB

Appel de la page

Base de
données

extraction
de données

Script
PHP

exécution

Document
HTML

Serveur de base
de données

Serveur
d'application

Serveur de
fichiers

Chapitre 1: Le langage HTML

Langage HTML (Hypertext markup Language)

1. HTML C'est quoi?
2. Comment écrire une page HTML?
3. Balises, attributs et caractères spéciaux
4. Balises de base
5. Balises de formatage du texte
6. Paragraphes, conteneurs (ou containers) et listes
7. Images et liens hypertextes
8. Les tableaux
9. Les formulaires
10. Balises spéciales
11. Les balises META
12. Validation et compatibilité HTML - Les moteurs de rendu
13. Exercice - Présentation d'une page HTML basique

Chapitre 1: Le langage HTML

HTML (HyperText Markup Language) est un langage de description (dit de marquage) de pages Web. Il permet de présenter les documents hypertextes destinés à être affichés sur le navigateur. Il s'agit d'un langage côté client (tout comme CSS et Javascript). Il est supporté et développé par W3C*.

L'origine du HTML remonte au début du Web. En effet, il a été inventé vers les années 1989 afin qu'il puisse présenter les documents qui circulent sur la toile et établir des liens entre eux à travers les liens hypertextes (ou hyperliens).

- Versions du langage

En 1989 HTML1.0 est apparu, il se basait sur les spécifications du standard SGML (Standard Generalized Markup Language) qui est aussi un langage de description à balises. Puis HTML2.0 a vu le jour en 1995. En 1997, HTML3 puis HTML4 sont apparus.

HTML4 est la version du langage la plus populaire, elle est toujours supportée de nos jours malgré l'apparition du standard HTML5.

* Le World Wide Web Consortium, abrégé par le sigle W3C, est un organisme de standardisation chargé de promouvoir la compatibilité des technologies du World Wide Web telles que HTML5, HTML, XHTML, XML, RDF, SPARQL, CSS, PNG...

Chapitre 1: Le langage HTML

L'objectif de ce cours et d'apprendre à utiliser le langage HTML.

Les outils dont vous aurez besoin

Pour bien maîtriser le langage HTML, il faut pratiquer. Il vous faudra donc un [éditeur](#) de code et un navigateur pour tester ce que vous avez codé.

Il existe des éditeurs WYSIWYG (pour What You See Is What You Get) qui permettent de construire facilement un document HTML sans avoir besoin de connaître la syntaxe.

Mais pour devenir un bon développeur il faut essayer d'écrire le code source par soi-même.

Je vous recommande donc de vous servir d'un éditeur texte. Personnellement je préfère Notepad++. (Sublime)
N'importe quel navigateur fera l'affaire. De préférence utiliser chrome.



Chapitre 1: Le langage HTML

Format d'une page HTML

Une page HTML est un fichier texte (après tout, le code HTML est du texte) avec l'extension .html ou .htm. Il n'y a absolument aucune différence entre les deux extensions. La première est l'extension normale alors que la deuxième est plus courte et aussi plus adaptées aux anciens systèmes d'exploitation qui comprenaient seulement les extensions en 3 caractères.

Les fichiers HTML sont déposés dans un serveur de fichiers. Il ne nécessite donc aucune exécution par le serveur car HTML est un langage côté client. Par conséquent, quand vous créez votre fichier HTML, il n'y a aucun problème à le mettre n'importe où dans votre ordinateur (au bureau par exemple) car c'est le navigateur qui l'exécutera après ouverture.

Les développeurs ont pour cotume de nommer la page d'accueil de leur site Web "index", est puisque c'est un document HTML alors son nom complet sera "index.html". Essayez de faire de même.

Chapitre 1: Le langage HTML

A quoi ressemble un fichier HTML?



```
index.html
1 <html>
2   <head>
3     <title>Ma première page HTML</title>
4     <meta http-equiv="content-type" content="text/html; charset=utf-8">
5   </head>
6   <body>
7     <h1>Il s'agit d'un exemple de HTML</h1>
8     <div>
9       <font color="#DD7700"><b>HTML</b></font>
10      est un langage de description de pages Web.
11    </div>
12  </body>
13 </html>
```

Il s'agit d'un exemple de HTML

HTML est un langage de description de pages Web.

Un document HTML contient du texte, mais vous avez certainement remarqué la présence des symboles < et >. Ces deux chevrons (ouvrant et fermant) avec le mot qu'ils contiennent constituent ce que l'on appelle une balise.

Un document HTML est structuré sous forme de balises, chaque balise a un sens que le navigateur comprend et exécute pour que le document soit présenté tel que les balises le dictent.

Chapitre 1: Le langage HTML

Balises HTML

Les **balises** constituent l'élément de base du langage HTML. Chaque balise a une signification que le navigateur connaît et applique au contenu. C'est comme ça que la présentation du document est effectuée.

Une balise est représentée par: **le chevron ouvrant + le nom de la balise + le chevron fermant**. Exemple **** ou **<body>**.

Il existe deux types de balises: des balises ouvrantes et des balises fermantes. Elles vont souvent ensemble de telle sorte que la balise ouvrante applique un effet au contenu qui la suit et la balise fermante cesse l'effet en question. La balise fermante se distingue par un slash (/) qui vient juste avant le nom de la balise. Exemple **** est la balise fermante de **** et **</body>** est la balise fermante de **<body>**.

Néanmoins, il existe des balises ouvrantes qui n'ont pas de balises fermantes associées. On les appelle des balises **auto fermantes** ou des balises **orphelines**. Ce sont des balises qui s'ouvrent et se ferment au même moment et elles appliquent leur effet à l'endroit où elles sont déclarées (et pas au contenu qui les suit comme les balises qu'on a vu ci haut). Exemple: **
**. (Saut de ligne)

Chapitre 1: Le langage HTML

Imbrication des balises

Un contenu peut subir l'effet de plusieurs balises à la fois. Si c'est le cas, alors une règle s'impose: En cas d'imbrication de plusieurs balises, la première balise ouverte est la dernière à fermer. Il faut donc fermer les balises dans l'ordre inverse de leur ouverture.

Le code ressemblera à ceci:

```
<balise1><balise2><balise3>Contenu</balise3></balise2></balise1>
```

Les attributs

La plupart des balises HTML peuvent appliquer des effets différents selon les préférences du développeur. C'est comme si on peut les personnaliser à notre guise. C'est là où les attributs interviennent.

Les attributs sont déclarés dans la balise ouvrante. Ils représentent des paramètres qui personnalisent la balise où ils sont définis et ils possèdent des valeurs. Une balise peut renfermer plusieurs attributs à la fois, chacun avec sa valeur.

```
<balise attribut1="valeur1" attribut2="valeur2" attribut3="valeur3"...>Contenu</balise>
```

`HTML
est un langage de description de pages Web.`

HTML est un langage de description de pages Web.

Chapitre 1: Le langage HTML

Pour que l'écriture du code HTML soit correcte il faut vérifier les points suivants:

- Il ne doit pas y avoir d'espace entre le chevron d'ouverture et le nom de la balise
- Il doit impérativement y avoir un espace (ou plus) entre le nom de la balise et l'attribut
- Il doit impérativement y avoir un espace (ou plus) entre les attributs successifs
- La valeur de l'attribut doit être déclarée entre des guillemets (doubles quote ou simple quote). Cependant les navigateurs peuvent tolérer l'absence des guillemets, mais votre code reste invalide du point de vue de W3C(*)

The screenshot shows a code editor on the left and a browser window on the right. The code editor displays the file 'index.html' with the following content:

```
index.html
1 <html>
2   <head>
3     <title>Ma première page HTML</title>
4     <meta http-equiv="content-type" content="text/html; charset=utf-8">
5   </head>
6   <body>
7     <h1>Il s'agit d'un exemple de HTML</h1>
8     <br>
9     <div>
10       <font color="#DD7700"><b>HTML</b></font>
11       est un langage de description de pages Web.
12     </div>
13
14     <br>
15     <br>
16
17     <font color="red" face="arial">Bonjour à tous</font>
18
19
20   </body>
21 </html> |
```

The browser window shows the rendered HTML with the following content:

Ma première page HTML

Il s'agit d'un exemple de HTML

HTML est un langage de description de pages Web.

Bonjour à tous

Chapitre 1: Le langage HTML

Les caractères spéciaux

HTML est constitué principalement de balises. Mais ce n'est pas tout car il y a des séquences, autres que les balises, que le navigateur reconnaît et remplace par leurs significations. On appelle ces séquences: caractères spéciaux.

Le code HTML des caractères spéciaux commence par le symbole "&" et fini par ";". Par exemple © signifie "©".

Carctère	Code HTML
"	"
&	&
<	<
>	>
oe	œ
Espace	&nbsp
£	£
©	©

®	®
±	±
µ	µ
½	½
ç	Ç
æ	æ

Chapitre 1: Le langage HTML

Les balises de bases

Les trois balises HTML minimales sont: <html>, <head> et <body>.

```
index.html
1 <html>
2   <head>
3     <title>Ma première page HTML</title>
4     <meta http-equiv="content-type" content="text/html;charset=utf-8">
5   </head>
6   <body>
7     <h1>Il s'agit d'un exemple de HTML</h1>
8     <br>
9     <div>
10       <font color="#DD7700"><b>HTML</b></font>
11       est un langage de description de pages Web.
12     </div>
13
14
15
16
17     <font color="red" face="arial">Bonjour à tous</font>
18
19
20   </body>
21 </html>
```

Chapitre 1: Le langage HTML

Les balises de bases

Les trois balises HTML minimales sont: <html>, <head> et <body>.

- Balise <html>

La balise <html> est la balise de premier niveau. Elle renferme tout le contenu de la page Web. Elle est déclarée au début du document. La balise fermante </html> vient clôturer le document en la déclarant à la fin.

- Balise <head>

La balise <head> représente l'entête du document. Elle contient des données supplémentaires qui ne sont pas nécessairement affichées sur le navigateur, mais elle peuvent fournir des informations complémentaires à l'exécution de la page par celui-ci, ou des informations utiles aux outils de recherche en vu de référencement.

Une des balises les plus utilisées dans la balise <head> est la balise <title>. Cette balise indique le titre du document. Celui-ci sera affiché sur la barre de titre du navigateur. C'est une balise qui doit être fermée pour indiquer la fin du titre.

On trouve aussi la balise <meta> qui fournit des informations supplémentaires sur le contenu.

Chapitre 1: Le langage HTML

Les balises de bases

Les trois balises HTML minimales sont: <html>, <head> et <body>.

- Balise <body>

C'est la plus importante parmi les balises vues précédemment. En effet, la balise <body> renferme le contenu du corps. C'est ce qui sera réellement visible sur le navigateur par les internautes.

La balise <body> doit aussi être fermée pour indiquer la fin du corps du document. Elle peut contenir des attributs, mais la plupart d'entre elles ne sont pas compatibles.

Cependant <body> peut contenir un attribut plus au moins utilisé, il s'agit de bgcolor qui permet de colorer l'arrière-plan de la page Web en entier. La balise est donc déclarée ainsi:

```
<body bgcolor="orange"></body>
```

La couleur d'arrière plan sera donc orange. Il suffit de mentionner le nom de la couleur en anglais pour que celle-ci soit reconnue en HTML. Cependant il existe une manière plus efficace pour déclarer les couleurs qu'on va voir après.

Chapitre 1: Le langage HTML

Les balises de bases

The image shows a dual-pane interface. On the left is a code editor with tabs for "index.html" and "index2.html". The "index2.html" tab is active, displaying the following HTML code:

```
1 <html>
2   <head>
3     <title>Titre de ma page Web</title>
4   </head>
5   <body bgcolor="orange">
6     ...
7   </body>
8 </html>
```

On the right is a web browser window titled "Titre de ma page Web". The address bar shows "Fichier | file:///C:/Users/oumay/Desktop/index2.html". The browser displays a solid orange page, which is the visual representation of the "bgcolor" style applied in the HTML code.

Chapitre 1: Le langage HTML

Balises de formatage

- **Balises de formatage sans attributs:**

Balise **** et ****

La balise **** met le texte qu'elle contient en gras (bold). C'est utile pour les textes importants ou les titres.

Par exemple:

```
<b>Ce texte est écrit en gras</b>
```

Ce qui donne:

Ce texte est écrit en gras

La balise **** met aussi le texte en gras, elle peut remplacer la balise ****, bien que celle ci est plus populaire.

Chapitre 1: Le langage HTML

Balises de formatage

- **Balises de formatage sans attributs:**

Balise <u>

La balise <u> (pour underline) souligne le texte qu'elle entoure. C'est utile pour attirer l'attention à un mot ou une phrase.

Par exemple:

```
<u>Texte souligné</u>
```

Ce qui donne:

Texte souligné

Chapitre 1: Le langage HTML

Balises de formatage

- **Balises de formatage sans attributs:**

Balise *<i>*

La balise *<i>* (pour italic) met le texte en italique. C'est pratique pour les citations ou les notes.

Par exemple:

```
<i>Texte en italique</i>
```

Ce qui donne:

Texte en italique

Chapitre 1: Le langage HTML

Balises de formatage

- Balises de formatage sans attributs:

Balise <s>

La balise <s> (pour strike) barre le texte. C'est pratique pour marquer un contenu obsolète ou invalide (par exemple: sur les sites E-commerce, on barre généralement les anciens prix d'articles en promotion, pour mettre en valeur le nouveau prix).

Par exemple:

```
<s>Texte barré</s>
```

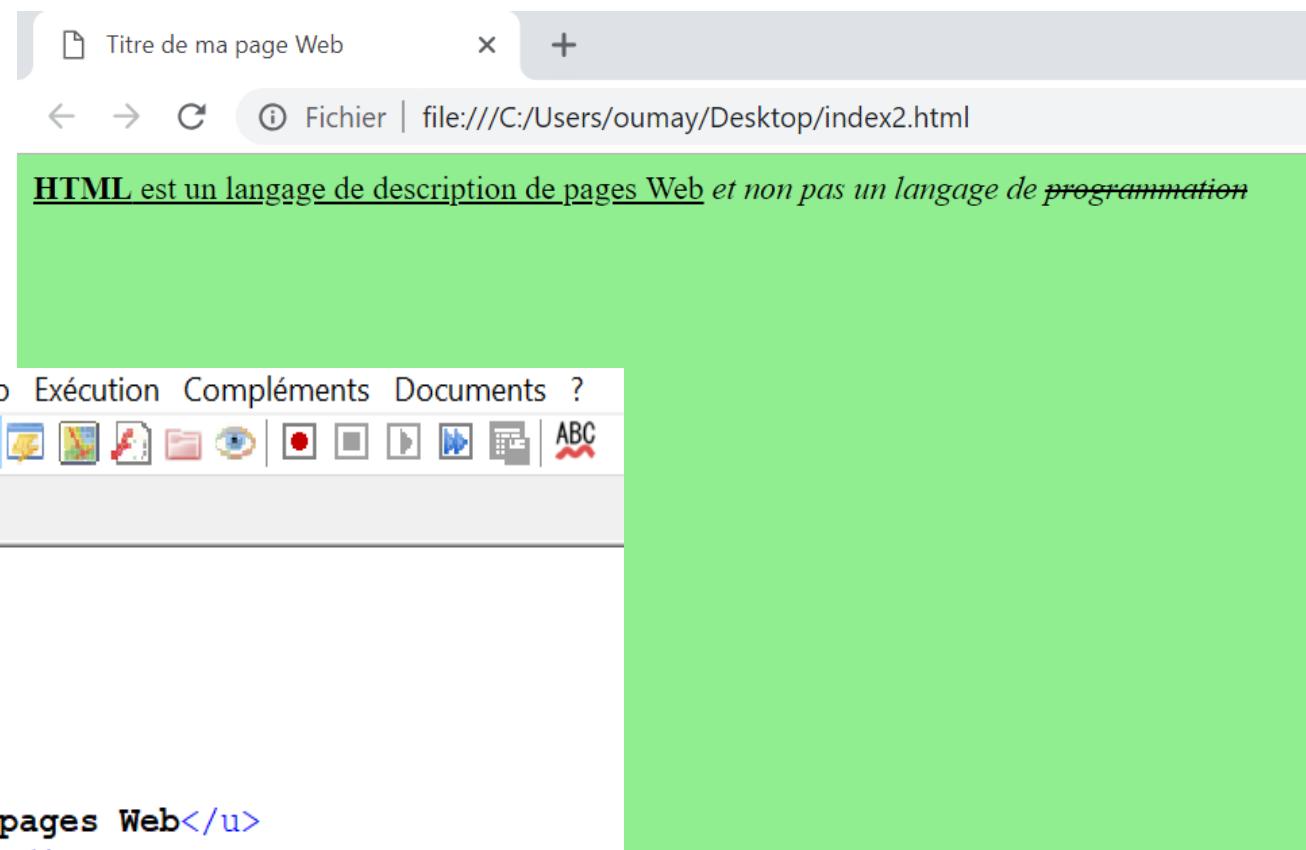
Ce qui donne:

~~Texte barré~~

Chapitre 1: Le langage HTML

Balises de formatage

- Balises de formatage sans attributs:



The screenshot shows a web browser window titled "Titre de ma page Web". The address bar indicates the file is located at "file:///C:/Users/oumay/Desktop/index2.html". The browser displays the text: "HTML est un langage de description de pages Web et non pas un langage de programmation".

The browser interface includes a menu bar with French options: Fichier, Édition, Recherche, Affichage, Encodage, Langage, Paramètres, Outils, Macro, Exécution, Compléments, Documents, and a help icon. Below the menu is a toolbar with various icons. The bottom left shows two tabs: "index.html" and "index2.html", with "index2.html" currently active.

```
1 <html>
2   <head>
3     <title>Titre de ma page Web</title>
4   </head>
5   <body bgcolor="lightgreen">
6
7     <u><b>HTML</b> est un langage de description de pages Web</u>
8     <i>et non pas un langage de <s>programmation</s></i>
9
10    </body>
11  </html>
```

Chapitre 1: Le langage HTML

Balises de formatage

- Balises de formatage sans attributs:

Balises **<h1>**, **<h2>**...**<h6>**

La balise **<h1>** (heading) désigne un titre de premier niveau. Le texte qu'elle entoure est de grande taille et mis en gras. Il est utile pour créer les grands titres d'une pages. La balise **<h2>** désigne un titre de deuxième niveau, elle applique le même effet que la balise précédente mais avec une taille de caractères légèrement plus petite. C'est pratique pour les sous-titres. Il existe aussi **<h3>**, **<h4>**, **<h5>** et **<h6>** avec une taille de caractères de plus en plus petite.

Par exemple:

```
<h1>Titre de premier niveau</h1>
<h2>Titre de deuxième niveau</h2>
<h3>Titre de troisième niveau</h3>
<h4>Titre de quatrième niveau</h4>
<h5>Titre de cinquième niveau</h5>
<h6>Titre de sixième niveau</h6>
```

Ce qui donne:

Titre de premier niveau

Titre de deuxième niveau

Titre de troisième niveau

Titre de quatrième niveau

Titre de cinquième niveau

Titre de sixième niveau

Chapitre 1: Le langage HTML

Balises de formatage

- Balises de formatage sans attributs:

Balises `<sup>` et `<sub>`

La balise `<sup>` met un texte en exposant et la balise `<sub>` le met en indice.

Par exemple, le code suivant:

```
x<sup>2</sup>
```

donne:

x^2

et ce code

```
H<sub>2</sub>O
```

donne

H_2O

Chapitre 1: Le langage HTML

Balises de formatage

- Balises de formatage sans attributs:

Retour à la ligne: Balise

Pour retourner à la ligne à n'importe quel moment, il suffit de déclarer la balise `
`. Il s'agit d'une balise orpheline (donc pas de `</br>`).

Exemple:

```
Ligne 1 <br>
Ligne 2 <br>
Et ligne 3
```

Ce qui donne le résultat suivant:

```
Ligne 1
Ligne 2
Et ligne 3
```

Chapitre 1: Le langage HTML

Balises de formatage

- **Balises de formatage sans attributs:**

Commentaire HTML <!-- -->

Un commentaire HTML est déclaré comme ceci: **<!-- Commentaire -->**. Il est visible dans le code source mais ignoré par le navigateur. Il sert à marquer un bloc de code pour que celui-ci soit facilement trouvé et compris lors de sa prochaine manipulation.

Exemple:

```
<!-- Ceci est un commentaire -->
<i>Le commentaire n'est pas visible sur le navigateur.</i>
```

Le résultat eu sur le navigateur est:

Le commentaire n'est pas visible sur le navigateur.

Chapitre 1: Le langage HTML

Balises de formatage

- Balises de formatage sans attributs:

Balise `<pre>`

Nous avons vu précédemment que pour intégrer un retour à la ligne il faut déclarer la balise `
`, et pour mettre plusieurs espaces ou tabulations il faut passer par le caractère spécial `&nbsp`. Cependant, il existe une méthode très simple pour afficher un contenu tel qu'il a été édité dans le code. C'est à dire, si vous mettez une tabulation dans le code, celle ci sera automatiquement reprise dans le navigateur, et si vous sautez une ligne, ce saut de ligne sera aussi visible sur le navigateur. La solution magique s'appelle la balise `<pre>`.

Exemple:

```
<pre>
  Là-haut sur la montagne, l'était un vieux chalet.
  Murs blancs, toit de bardeaux,
  Devant la porte un vieux bouleau.
  Là-haut sur la montagne, l'était un vieux chalet.
</pre>
```

Le résultat du code est le suivant:

```
Là-haut sur la montagne, l'était un vieux chalet.
Murs blancs, toit de bardeaux,
Devant la porte un vieux bouleau.
Là-haut sur la montagne, l'était un vieux chalet.
```

Chapitre 1: Le langage HTML

Balises de formatage

- Balises de formatage avec attributs:

Avec la même balise on peut réussir plusieurs effets différents. C'est grâce aux attributs que celle ci renferme. Une des balises les plus populaires de HTML est la balise ****.

Balise ****

La balise de formatage de texte la plus connue est sans doute la fameuse ****. Elle doit sa notoriété aux effets qu'elle peut apporter aux textes qu'elle inclue.

La balise **** dispose de trois attributs principaux qui permettent d'appliquer des effets non négligeables aux textes. Ces trois attributs sont: **face**, **size**, et **color**.

L'attribut **face**:

L'attribut **face** permet de spécifier la police à utiliser pour afficher le texte inclus dans la balise ****. Par défaut c'est la police "Times New Roman" qui est appliquée par la plupart des navigateurs si aucune police n'est spécifiée. "Times New Roman" est une police de la famille "Sérif" et elle n'est pas très adaptée aux pages Web. Cependant d'autres polices pourront bien faire l'affaire comme "verdana" ou "arial" qui sont assez populaires et sont supportés par de nombreux systèmes d'exploitation.

Chapitre 1: Le langage HTML

Balises de formatage

- Balises de formatage avec attributs:

L'attribut size:

L'attribut **size** change la taille du texte. Par défaut le navigateur applique la taille 12 points (écrit 12pt) aux polices, sauf quelques exceptions comme "Netscape Navigator" qui applique par défaut la taille 10pt. (Le point est l'unité de mesure des polices).

HTML ne supporte (malheureusement) que 7 tailles de police seulement, elle sont déclarées par des indices allant de 1 à 7. Le tableau suivant détaille la valeur de chaque indice:

Indice (valeur de l'attribut size)	Valeur réelle en pt
1	8pt
2	10pt
3 (par défaut)	12pt
4	14pt
5	18pt
6	24pt
7	36pt

Chapitre 1: Le langage HTML

Balises de formatage

- **Balises de formatage avec attributs:**

L'attribut color:

Dans un texte, il n'y a pas que la police et la taille, il y a aussi la couleur. L'attribut **color** sert à appliquer une couleur au texte entouré par la balise ****. La méthode la plus simple pour définir une couleur c'est par son nom anglais (red pour rouge, yellow pour jaune...).

Un problème surgit aussitôt à la surface. Combien de couleurs peut on définir de cette manière? sans doute pas beaucoup (un peu plus de 100 couleurs). Cependant un écran peut afficher environ 16 millions de couleurs. Pour couvrir toutes ces nuances, la seule solution consiste à utiliser les codes des couleurs.

Gestion des couleurs en HTML (codes des couleurs)

Pour qu'un écran affiche une image en couleur, il combine 3 couleurs dites **primaires**. Ces couleurs sont **Rouge**, **Vert** et **Bleu**. On parle alors de la base RVB (RGB en anglais). Le fait de mélanger ces trois couleurs à des proportions différentes donne naissance aux 16 millions de couleurs dont on a parlé au paragraphe précédent.

Techniquement, chaque couleur primaire est codée sur un octet (8 bits). Les trois couleurs sont donc codées sur 24 bits et du coup on peut avoir jusqu'à 16777216 couleurs.

Chapitre 1: Le langage HTML

Balises de formatage

- Balises de formatage avec attributs:

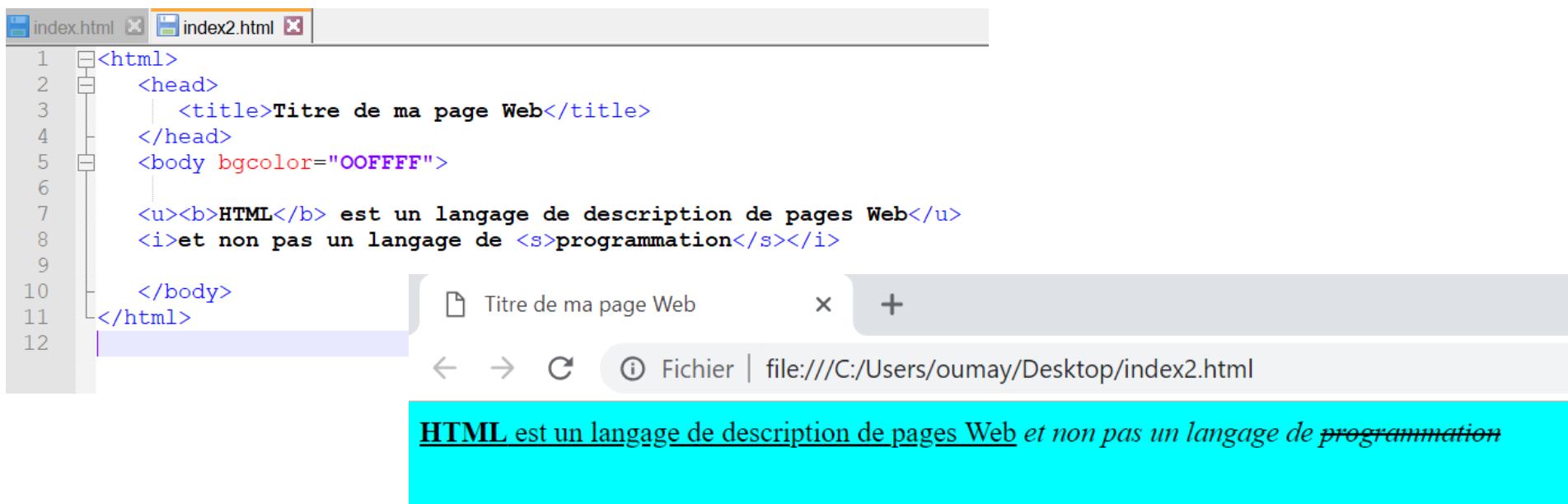
Le tableau suivant illustre les couleurs basiques:

couleur	Code HTML
Rouge	#FF0000
Vert	#00FF00
Bleu	#0000FF
Blanc	#FFFFFF
Noir	#000000
Jaune	#FFFF00
Magenta	#FF00FF
Cyan	#00FFFF
Gris moyen	#888888
Orange	#FF8800

```
<font color="#FF0000" face="verdana" size="2">  
    Ce texte a été formaté grâce à la balise font  
</font>
```

Le résultat obtenu est:

Ce texte a été formaté grâce à la balise font



```
index.html index2.html  
1 <html>  
2   <head>  
3     <title>Titre de ma page Web</title>  
4   </head>  
5   <body bgcolor="#00FFFF">  
6  
7     <u><b>HTML</b> est un langage de description de pages Web</u>  
8     <i>et non pas un langage de <s>programmation</s></i>  
9  
10    </body>  
11  </html>  
Titre de ma page Web  
Fichier | file:///C:/Users/oumay/Desktop/index2.html  
HTML est un langage de description de pages Web et non pas un langage de programmation
```

Chapitre 1: Le langage HTML

1. Le code suivant est-il correct?

```
<b><i><u>Bonjour à tous</u></b></i>
```

2. Si on appliquait la balise <u> à un texte dans la balise , est ce que celui-ci serait souligné?

3. Par défaut, combien mesure la taille de caractères d'un titre

?

- 24pt
- 36pt
- 18pt
- 20pt

4. Que fait le code suivant?

```
<font color="#0000FF" face="arial" size="2">Texte formaté</font>
```

- Affiche le texte "Texte formaté" en vert avec une taille de caractères de 2 points
- Affiche le texte "Texte formaté" en vert avec une taille de caractères de 10 points
- Affiche le texte "Texte formaté" en bleu avec une taille de caractères de 2 points
- Affiche le texte "Texte formaté" en bleu avec une taille de caractères de 10 points

5. Quelle est l'erreur commise dans ce code?

```
<html>
  <head>
  </head>
  <body>
    <b>Texte en gras</b><br>
    <u>Texte souligné</u><br>
  </body>
</html>
```

- La balise a été oubliée
- La balise ne contient pas assez de contenu
- La balise
 ne doit pas être fermée

Chapitre 1: Le langage HTML

Les paragraphes

Nous avons vu l'intérêt de la balise `
` qui sert à sauter la ligne à n'importe quel moment au milieu d'un contenu HTML. Déjà avec cette balise on peut dire qu'on arrive à séparer les paragraphes entre elles. Mais il existe une balise plus adapté à cet usage. Il s'agit de `<p>`.

Balise `<p>`

- La balise `<p>` est une balise de type **block**, c'est à dire qu'elle crée un bloc et engendre automatiquement un retour à la ligne. Elle sert à définir un paragraphe. On peut la doter de l'attribut **align** qui permet d'aligner, à sa guise, le contenu du paragraphe. Les différentes valeurs de l'attribut **align** sont:

left: C'est la valeur par défaut. Elle permet d'aligner le contenu du paragraphe à gauche de la page (ou à gauche du conteneur qui renferme la balise `<p>`).

right: Elle permet d'aligner le contenu du paragraphe à droite de la page (ou à droite du conteneur qui renferme la balise).

center: Elle permet de centrer le paragraphe.

justify: Elle permet de justifier le contenu du paragraphe (prolonge le texte pour qu'il occupe toute la ligne).

Chapitre 1: Le langage HTML

Les paragraphes

Balise <p>

```
<h1>Les paragraphes</h1>
<p align="left">
    La balise &lt;p&gt; permet de définir un paragraphe au sein d'un code HTML.
    Il engendre un retour à la ligne avant et après.
</p>
```

L'exécution du code sur le navigateur donne:

Les paragraphes

La balise <p> permet de définir un paragraphe au sein d'un code HTML. Il engendre un retour à la ligne avant et après.

Chapitre 1: Le langage HTML

Les conteneurs (ou containers)

On entend par **conteneur** (ou container) des balises qui peuvent renfermer d'autres éléments comme du textes ou des images. Les conteneurs les plus célèbres sont les balises **<div>** et ****.

Balise **<div>**

La balise **<div>** est une balise de type **block**. Elle permet de définir un conteneur ou un bloc qui contient d'autres éléments. Il sert généralement à mieux diviser la page Web pour placer le bon contenu au bon endroit. Cependant, la balise **<div>** nécessite des styles CSS pour qu'elle soit dotée de toute sa force.

En HTML cette balise ne sert pas à grand chose si ce n'était que pour aligner du texte ou définir un paragraphe (Elle se comporte à peu près comme la balise **<p>**). Les attributs dont on peut la doter sont les même que ceux de la balise **<p>**.

Exemple:

```
<h1>Les conteneurs</h1>
<div align="left">
    La balise &lt;div&gt; permet de définir un conteneur de type block au sein d'un code HTML.
    Il engendre un retour à la ligne avant et après.
</div>
```

L'exécution du code sur le navigateur donne:

Les conteneurs DIV

La balise **<div>** permet de définir un conteneur de type block au sein d'un code HTML. Il engendre un retour à la ligne avant et après.

Chapitre 1: Le langage HTML

Les conteneurs (ou containers)

Balise ``

La balise `` est aussi un container, mais de type **inline** (elle occupe juste assez d'espace pour afficher ce qu'elle contient et pas toute la ligne comme la balise `<div>`). Par conséquent elle n'engendre pas de retour à la ligne.

Exemple:

```
HTML est un <span>langage</span> de description de pages Web.
```

Sur le navigateur cela donne:

```
HTML est un langage de description de pages Web.
```

Chapitre 1: Le langage HTML

Les listes HTML

Pour afficher plusieurs entrées, l'une en dessous de l'autre, il n'y a pas mieux que les listes. Il y'en a deux types: des listes ordonnées et des listes non ordonnées.

Listes ordonnées: Balise ``

La balise `` permet d'avoir des listes ordonnées (ou numérotées). Elle doit être fermée (par ``) et elle contient une autre balise qui est ``. La balise `` permet de définir une entrée de la liste (Il y 'a donc autant de balises `` que d'entrées).

La balise `` peut avoir des attributs dont les plus importants sont:

- **type**: sert à définir le type de marqueur de la liste. Les différentes valeur de l'attribut sont:

1 pour les chiffres classiques (il s'agit d'ailleurs de la valeur par défaut),

A pour l'alphabet majuscule,

a pour l'alphabet minuscule,

I pour les chiffres romains majuscules

i pour les chiffres romains minuscule.

- **start**: indique la valeur du marqueur du début. Par défaut la liste commence par 1 (si elle est ordonnée). Si vous mettez `start="5"` alors la première entrée sera marquée par 5 et non par 1, la deuxième par 6...

Chapitre 1: Le langage HTML

Exemple:

```
Les noms des balises de bases à déclarer dans un document HTML sont:  
<ol type="1">  
    <li>HTML</li>  
    <li>HEAD</li>  
    <li>BODY</li>  
</ol>
```

On obtient sur le navigateur:

```
Les noms des balises de bases à déclarer dans un document HTML sont:  
1. HTML  
2. HEAD  
3. BODY
```

Chapitre 1: Le langage HTML

Liste non ordonnées: Balise ``

- La balise `` permet de créer des listes non ordonnées. Elle contient aussi la balise `` qui permet de définir les éléments de la liste.

Tout comme la balise ``, la balise `` possède des attributs qui permettent de personnaliser la liste. Le plus important d'entre eux est l'attribut **type** qui définit le type de marquer et qui peut avoir une des valeurs suivantes:

- **disc**: le marqueur est un cercle plein. Il s'agit de la valeur par défaut.
- **circle**: le marqueur est un cercle creux.
- **square**: le marqueur est un carré plein.

Le résultat obtenu ressemble à ceci:

On reprend l'exemple précédent avec la balise `` cette fois:

```
Les noms des balises de bases à déclarer dans un document HTML sont:  
<ul type="square">  
  <li>HTML</li>  
  <li>HEAD</li>  
  <li>BODY</li>  
</ul>
```

Les noms des balises de bases à déclarer dans un document HTML sont:

- HTML
- HEAD
- BODY

Chapitre 1: Le langage HTML

Les images

Pour afficher une image en HTML on fait appelle à la balise ``. Toutefois on peut afficher les images en tant qu'arrière plan (CSS) .

Balise ``

- La balise `` permet d'afficher une image sur le navigateur. Il s'agit d'une balise orpheline (donc pas de ``).

La balise `` peut avoir plusieurs attributs. Les plus importants sont:

- **src**: désigne la source de l'image. La valeur associée correspond au chemin vers elle (ou son emplacement). Le chemin peut être absolu ou relatif. Le chemin absolu représente l'[URL](#) de l'image (par exemple: <http://www.logistiqua.com/wp-content/uploads/2016/02/logo-EST-FES.png>). Le chemin relatif, quant à lui, désigne l'emplacement de l'image à partir de l'emplacement actuel (par exemple: images/Logo_ESTF.png).

- **border**: accepte une valeur numérique (1,2...). Cette valeur désigne l'épaisseur de la bordure qui encadre l'image. L'unité utilisée pour la bordure est le pixel (px) mais elle est implicite, seule la valeur numérique est déclarée. Par défaut la bordure est de couleur noire.

Chapitre 1: Le langage HTML

Les images

- **width** et **height**: signifient, respectivement, la largeur et la hauteur de l'image. L'unité peut être le pixel (qui est implicite) ou le pourcentage (qu'il faut expliciter à l'aide du symbole universel %). Si par exemple on donne à la largeur la valeur 100%, cela veut dire que l'image occupe toute la largeur de la page (ou du conteneur qui accueille la balise ****).

Si width ou height ne sont pas spécifiée alors l'image sera affichée en taille réelle.

Si seule la valeur d'une dimension (width ou height) est déclarée, l'autre dimension sera automatiquement recalculée par le navigateur de telle sorte à conserver les proportions originales de l'image.

- **alt**: représente le texte alternatif à l'image. Il accepte comme valeur un mot ou une phrase qui sera affichée à la place de l'image si celle ci n'est pas chargée correctement.
- **title**: il s'agit d'un message qui sera affiché dans une infobulle si on survole l'image avec la souris.

Chapitre 1: Le langage HTML



```
index.html index2.html
1 <html>
2   <head>
3     <title>Titre de ma page Web</title>
4   </head>
5   <body >
6
7     <b> Ecole Supérieure de Technologie de FES</b>
8
9     <br>
10    <br>
11      <br>
12    <center>
13      
15    </center>
16  </body>
17</html>
```

Oumayma BANOUAR
oumayma.banouar@gmail.com

Chapitre 1: Le langage HTML

Les liens hypertextes

Un lien hypertexte ou hyperlien est un objet HTML (texte, image ou autre) sur lequel on peut cliquer pour aller vers une autre page (ou un autre emplacement dans la même page). Il est considéré comme l'un des éléments les plus importants du langage HTML puisqu'il permet de relier les pages les unes aux autres pour constituer ainsi le **Web** tel que nous le connaissons.

Balise <a>

- La balise **<a>** (pour **anchor** ou ancre) permet de créer un lien hypertexte pointant vers un autre document ou autre emplacement. Un lien classique est connaissable par sa couleur bleue et un style souligné (ces propriétés peuvent être changées à l'aide du langage CSS).

La balise **<a>** dispose des attributs suivants:

- **href**: désigne le chemin vers la page de destination après avoir cliqué sur le lien. Comme pour l'attribut **src** de la balise ****, le chemin peut être absolu ou relatif.

Si par exemple la valeur de l'attribut **href** vaut "mailto:adresse@mail" alors, en cliquant sur le lien, le logiciel de messagerie par défaut installé chez le client s'ouvrira en l'invitant à écrire un mail à "adresse@mail".

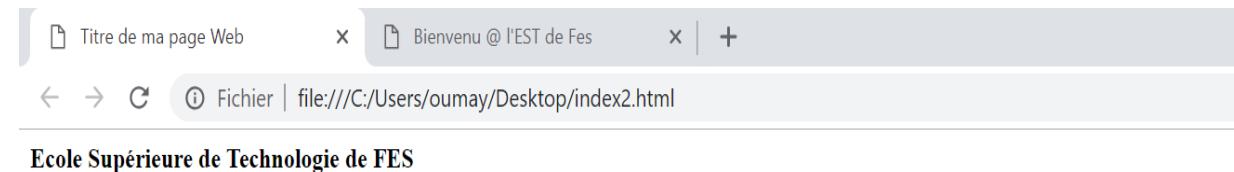
- **target**: désigne où la page de destination sera ouverte. Il peut contenir plusieurs valeurs mais une seule est fréquemment

Chapitre 1: Le langage HTML

Les liens hypertextes

- **target**: désigne où la page de destination sera ouverte. Il peut contenir plusieurs valeurs mais une seule est fréquemment utilisée il s'agit de **_blank** (n'oubliez pas le souligné avant le mot blank). La valeur **_blank** signifie que le lien sera ouvert dans une nouvelle fenêtre (ou nouvel onglet). C'est utile pour les liens pointant vers d'autres sites Web. Si l'attribut **target** n'est pas spécifié, alors la page de destination sera ouverte dans la même fenêtre.

- **title**: décrit un message qui sera affiché dans une infobulle si on survole le lien avec la souris.



```
<a href="http://www.est-usmba.ac.ma/" title="Aller vers ESTF" target="_blank">  
Lien vers le site de l'ESTF  
</a>
```

[Lien vers le site de l'ESTF](http://www.est-usmba.ac.ma/)



Oumayma BANOUAR
oumayma.banouar@gmail.com

Chapitre 1: Le langage HTML

Liens internes

Il arrive des fois qu'en cliquant sur un lien, on se déplace vers un autre emplacement dans la même page déjà ouverte. On l'appelle alors un lien interne.

Le principe est simple. Supposons que vous voulez prévoir un lien qui déplace l'internaute vers une image dans la même page. Ce qu'il faut faire c'est ajouter une **ancre** près de l'image (de préférence avant la balise ****). L'ancre aura la forme suivante:

```
<a name="toto"></a>
```

(Vous êtes libre de mettre ce que vous voulez dans l'attribut **name**). Notez que cet ancre ne sera pas visible car la balise **<a>** ne contient aucun texte. Ensuite on va créer le lien qui nous mènera vers l'ancre créé comme ceci:

```
<a href="#toto">Aller vers toto</a>
```

Notez que le **#** fait référence à la page courante. Après le **#** on déclare le nom de l'ancre (la valeur de l'attribut **name** de tout à l'heure).

En cliquant sur le texte "Aller vers toto" on sera déplacé automatiquement vers l'ancre et par conséquent vers l'image souhaitée.

Chapitre 1: Le langage HTML

1. Une page Web peut contenir un nombre illimité d'images

- Oui
- Non

2. Si on applique la balise `<i>` à une image, cette dernière sera inclinée

- Oui
- Non

3. La balise `` permet d'intégrer des images JPEG et PNG seulement

- Oui
- Non

4. Bien que l'on peut redimensionner les images avec les attributs `width` et `height`, il est peu recommandé de recourir à cette pratique.

- Oui
- Non

Chapitre 1: Le langage HTML

Les tableaux HTML

Nous avons déjà vu deux balises qui servent de conteneur à savoir `<div>` et ``. C'est à dire qu'ils peuvent contenir d'autres balises afin de donner une structure au document. Mais j'ai précisé que ces deux balises ne servent pas à grand chose si nous ne les dotons pas de styles CSS. Cependant, il existe un autre objet HTML qui peut servir de conteneur sans être obligé de faire appel au CSS. C'est le tableau HTML.

Les tableaux peuvent servir de grille qui contiennent des données structurées en ligne et en colonne. Comme par exemple une liste d'articles avec leurs prix. Mais ils peuvent servir également de conteneurs qui renferment d'autres objets HTML afin de structurer sa page Web sous forme de cellules et faciliter ainsi la mise en page.

Pour créer un tableau nous avons besoin, au moins, de trois balises de base: `<table>`, `<tr>` et `<td>`.

Balise `<table>`

- La balise `<table>` est le conteneur principal. Elle permet de déclarer un tableau et peut accueillir de nombreux attributs dont les plus fréquents sont:
 - **border:** définit l'épaisseur de la bordure du tableau (et de ses cellules). Elle est exprimée en pixels (px) mais nous nous contentons de déclarer la valeur sans l'unité. Celle-ci étant implicite et connue par le navigateur.
 - **width:** définit la largeur du tableau. Elle est exprimée en pixel ou en pourcentage. Si on entend les pixels alors on met la valeur sans l'unité. Si c'est du pourcentage alors on met la valeur suivie du symbole %.

Oumayma BANOUAR

oumayma.banouar@gmail.com

Chapitre 1: Le langage HTML

Les tableaux HTML

Balise <table>

- **height**: définit la hauteur du tableau en pixel. Mais il vaut mieux ne pas déclarer cet attribut et laisser le tableau s'étirer en fonction de la taille de son contenu.
- **bgcolor**: désigne la couleur de l'arrière plan du tableau. Cette couleur est exprimée en nom anglais ou en code de couleur.
- **cellspacing**: signifie l'espacement entre les différentes cellules du tableau. Sa valeur est exprimée en pixel (implicite).
- **cellpadding**: définit la marge interne des cellules. Elle est exprimée en pixel (implicite).

Balise <tr>

Même si vous déclarez la balise <table> dans le code, rien ne s'affichera sur le navigateur.

En effet, ce qui peut être affiché ce sont les cellules d'un tableau, et une cellule c'est au moins une ligne et une colonne.

La balise <tr> permet d'ajouter une ligne à un tableau. Elle est directement déclarée dans la balise <table>.

Il y a donc autant de balises <tr> que de lignes.

La balise <tr> peut accueillir des attributs comme **height** ou **bgcolor**, mais la plupart des développeurs préfèrent la déclarer sans attribut.

Chapitre 1: Le langage HTML

Les tableaux HTML

Balise <td>

- La balise <td> représente une colonne. C'est elle en fait le vrai conteneur, car c'est elle qui peut accueillir du contenu. Elle est déclarée dans la balise <tr>. Si vous voulez avoir un tableau avec une ligne et deux colonnes, alors vous devez déclarer une balise <tr> et dans celle ci, deux balises <td>.

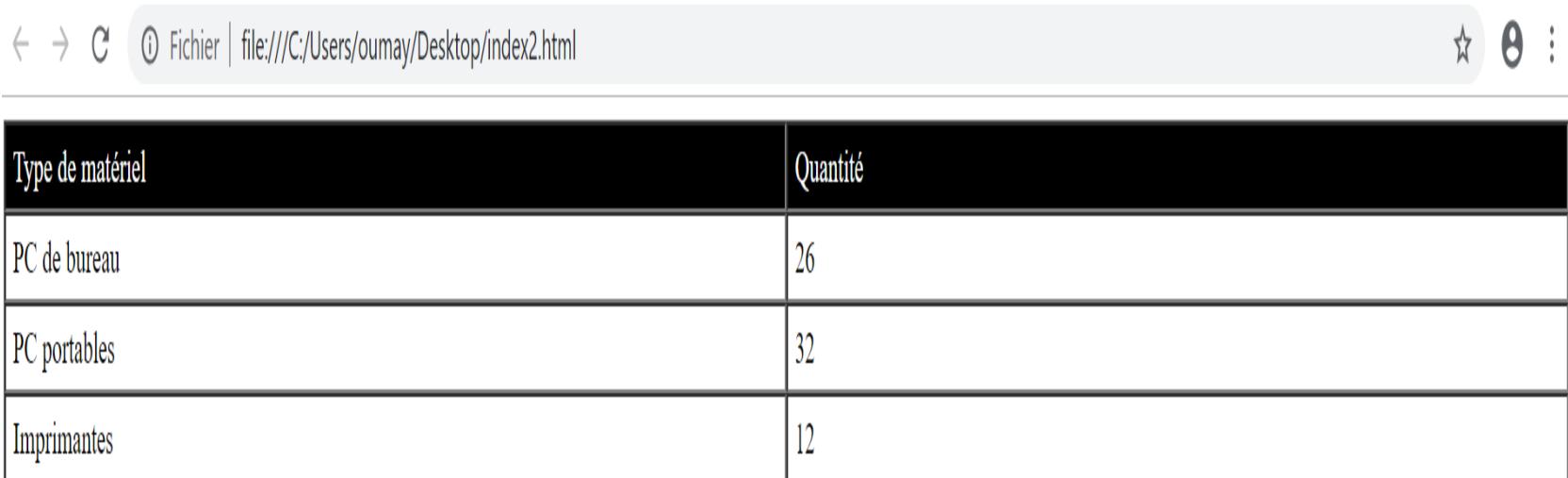
Si je vous ai conseillé de laisser la balise <tr> sans attributs, c'est parce que c'est la balise <td> qui doit en avoir. Les principaux attributs qu'on peut déclarer sur la balise <td> sont:

- **width**: définit la largeur de la colonne. Elle est exprimé en pixel ou en pourcentage par rapport à la largeur du tableau.
- **height**: définit la hauteur de la colonne en pixel.
- **bgcolor**: désigne la couleur de l'arrière plan de la colonne. Cette couleur est exprimée en nom anglais ou en code de couleur. Si la couleur d'arrière plan n'est pas déclarée, alors la colonne est transparente, et c'est la couleur de l'arrière plan du tableau qui sera visible.
- **align**: permet de spécifier l'alignement des objets contenus dans la colonne. Elle peut avoir une des valeurs: **left** (valeur par défaut), **right**, **center** ou **justify**.
- **valign**: défini l'alignement vertical des objets contenus dans la colonne. Elle peut avoir une des valeurs: **middle** (valeur par défaut), **top** ou **bottom**.

Chapitre 1: Le langage HTML

Les tableaux HTML

```
<html>
  <head>
    <title>Titre de ma page Web</title>
  </head>
  <body>
    <table border="1" width="100%" cellspacing="0" cellpadding="6">
      <tr>
        <td width="50%" bgcolor="#000000"><font color="#FFFFFF">Type de matériel</font></td>
        <td width="50%" bgcolor="#000000"><font color="#FFFFFF">Quantité</font></td>
      </tr>
      <tr>
        <td width="50%">PC de bureau</td>
        <td width="50%">26</td>
      </tr>
      <tr>
        <td width="50%">PC portables</td>
        <td width="50%">32</td>
      </tr>
      <tr>
        <td width="50%">Imprimantes</td>
        <td width="50%">12</td>
      </tr>
    </table>
  </body>
</html>
```



The screenshot shows a web browser window with the title bar "Fichier | file:///C:/Users/oumay/Desktop/index2.html". Below the title bar is a toolbar with icons for back, forward, search, and other file operations. The main content area displays a table with three rows. The first row contains the column headers "Type de matériel" and "Quantité". The second row contains the values "PC de bureau" and "26". The third row contains the values "PC portables" and "32". The fourth row contains the values "Imprimantes" and "12". The table has a black header row and white data rows.

Type de matériel	Quantité
PC de bureau	26
PC portables	32
Imprimantes	12

Chapitre 1: Le langage HTML

Les tableaux HTML

Balise <caption>

La balise **<caption>** est déclarée directement après la balise **<table>**. Elle contient le titre du tableau et elle peut accueillir l'attribut **align** qui peut prendre une des valeurs suivante: **top** (valeur par défaut) permet d'avoir le titre en dessus du tableau et **bottom** permet d'avoir le titre en dessous du tableau.

Balise <thead>

La balise **<thead>** permet de déclarer les cellules d'entête. Elle est déclarée dans la balise **<table>** et peut contenir des lignes (balise **<tr>**) qui contiennent des colonnes (balise **<td>**). Les cellules d'entête sont incluses en haut.

Balise <th>

Quand la balise **<thead>** est déclarée, il est préférable de remplacer ses balises **<td>** par **<th>**. La balise **<th>** désigne donc une légende pour les colonnes d'entête.

Balise <tbody>

Il s'agit d'une balise qui est sensée accueillir les cellules du corps du tableau. **<tbody>** est déclarée dans la balise **<table>** et peut contenir les balises **<tr>** puis **<td>**.

Balise <tfoot>

Il s'agit d'une balise qui est sensée accueillir les cellules du pied du tableau. **<tfoot>** est déclarée dans la balise **<table>** après la balise **<thead>** et peut contenir les balises **<tr>** puis **<td>**. Les cellules de la balises **<tfoot>** sont automatiquement intégrées à la fin du tableau.

Chapitre 1: Le langage HTML

```
<html>
  <head>
    <title>Titre de ma page Web</title>
  </head>
  <body>
    <table border="1" width="100%" cellspacing="0" cellpadding="6">
      <caption>Liste du matériel informatique disponible</caption>
      <thead>
        <tr>
          <th width="50%">Type de matériel</th>
          <th width="50%">Quantité</th>
        </tr>
      </thead>
      <tfoot>
        <tr>
          <th width="50%">Total</th>
          <th width="50%">70</th>
        </tr>
      </tfoot>
      <tbody>
        <tr>
          <td width="50%">PC de bureau</td>
          <td width="50%">26</td>
        </tr>
        <tr>
          <td width="50%">PC portables</td>
          <td width="50%">32</td>
        </tr>
        <tr>
          <td width="50%">Imprimantes</td>
          <td width="50%">12</td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

← → ⌂ Fichier | file:///C:/Users/oumay/Desktop/index2.html

Liste du matériel informatique disponible

Type de matériel	Quantité
PC de bureau	26
PC portables	32
Imprimantes	12
Total	70

Chapitre 1: Le langage HTML

Fusionnement de cellules d'un tableau

Si on suppose qu'on veut créer un tableau qui contient deux lignes, et si la ligne du dessus contient deux colonnes, alors ce qu'on a fait jusqu'ici ne nous permet pas d'avoir un nombre différent de colonnes dans la ligne du dessous. Alors cette dernière ne peut comporter que deux colonnes si on tient à avoir un affichage correcte. Néanmoins, il existe des cas (d'ailleurs très nombreux) où on a besoin d'avoir un nombre de cellules différent dans chaque ligne. La solution magique est le **fusionnement des cellules**.

On peut fusionner les cellules horizontalement ou verticalement. Mais le plus fréquent c'est de fusionner à l'horizontale grâce à l'attribut **colspan**.

L'attribut **colspan**

L'attribut **colspan** est inclus dans la balise de colonne **<td>**. Il indique à celle ci combien de colonnes il faut fusionner pour en avoir qu'une seule à la fin.

Un exemple sera en mesure de clarifier les idées. Alors, supposons que je veux créer un tableau qui contient deux lignes et chaque ligne contient deux colonnes. Bien entendu, y a rien de nouveau à cela. Voici le code:

Chapitre 1: Le langage HTML

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

Cellule 1	Cellule 2
Cellule 3	Cellule 4

```
<table>
  <tr>
    <td colspan="2">Cellule 1</td>
  </tr>
  <tr>
    <td>Cellule 3</td>
    <td>Cellule 4</td>
  </tr>
</table>
```

Cellule 1	
Cellule 3	Cellule 4

Chapitre 1: Le langage HTML

L'attribut rowspan

Si **colspan** fusionne les colonnes à l'horizontale, **rowspan** quant à elle, elle les fusionne à la verticale. Le principe est le même: Enlevez la (les) colonnes en plus, et sur celle qui reste mettez **rowspan="le nombre de colonnes à fusionner"**. Par exemple. avant le fusionnement:

```
<table>
  <tr>
    <td rowspan="2">Cellule 1</td>
    <td>Cellule 2</td>
  </tr>
  <tr>
    <td>Cellule 3</td>
  </tr>
</table>
```

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

Cellule 1	Cellule 2
Cellule 3	Cellule 4

Cellule 1	Cellule 2
Cellule 3	Cellule 4

Chapitre 1: Le langage HTML

1. Que fait le code suivant?

```
<table>
  <tr>
    <td colspan="2"></td>
  </tr>
  <tr>
    <td></td>
    <td></td>
  </tr>
</table>
```

- Crée un tableau à deux lignes et deux colonnes
- Crée un tableau à deux lignes, la première contient une seule colonne et la deuxième en contient deux
- Crée un tableau à deux lignes, la première contient deux colonnes et la deuxième contient une seule
- Crée un tableau à trois lignes.

2. Le code suivant est-il correct?

```
<table>
  <tr>
    <font color="red">Mon tableau</font>
    <td></td>
  </tr>
</table>
```

3. Un tableau doit impérativement contenir autant de colonnes dans chaque ligne.

4. Par défaut, le contenu de la balise `<caption>` est placé

- En dessus du tableau
- En dessous du tableau

Oui

Non

Chapitre 1: Le langage HTML

Les formulaires

Qu'est ce qu'un formulaire?

Un **formulaire HTML** permet de rendre la page Web plus interactive en la rendant capable de dialoguer avec l'internaute à travers des champs de saisie et boutons. Un formulaire permet à l'internaute de saisir du texte ou de valider un choix ou encore de sélectionner une entrée. Ces informations seront ensuite (dans la plupart des cas) envoyées au serveur pour les traiter. C'est très utile surtout quand il s'agit d'une [page Web dynamique](#).

Balise `<form>`

La balise `<form>` délimite les champs du formulaire (zones de texte, boutons...). Elle peut avoir plusieurs attributs dont voici les plus fréquents:

L'attribut `name`

L'attribut **name** permet d'identifier le formulaire par un nom. On peut mettre n'importe quel nom (à conditions qu'il ne contienne que les caractères conventionnels comme les lettres et les chiffres). Le nom du formulaire est unique. En effet, nous pouvons déclarer plusieurs formulaires dans la même page, pour les distinguer les uns des autres il suffit de leur donner des noms différents.

Chapitre 1: Le langage HTML

L'attribut **method**

L'attribut **method** permet de spécifier la méthode à utiliser pour envoyer le formulaire au serveur. En effet, l'objectif d'un formulaire c'est de transférer les données saisies par le client au serveur (on dit généralement **poster** ou **soumettre** le formulaire). En HTML, il existe deux méthodes pour envoyer un formulaire: **GET** et **POST**.

La méthode **GET**

La méthode **GET** (`method="get"`) est la valeur par défaut si on ne précise aucune valeur pour l'attribut **method**. Cette méthode permet d'envoyer les données du formulaire à travers l'URL(Uniform Resource Locator) en utilisant les symboles `?` pour préciser la suite de données et `&` pour séparer les données entre elles.

Pour mieux comprendre, imaginons que sur un site il existe une page nommée **infos.html** qui renferme un formulaire utilisant la méthode **GET** et qui contient deux zones de textes nommées respectivement **ville** et **pays**. Au chargement du formulaire, le client a saisi dans les deux champs les valeurs **Fès** et **Maroc**. Quand le client ordonnera l'envoi du formulaire, les données saisies seront envoyées via l'URL et celui ci aura la forme suivante:

<http://www.site.ma/infos.html?ville=Fès&pays=Maroc>

Chapitre 1: Le langage HTML

L'attribut **method**

La méthode **POST** (`method="post"`) permet d'envoyer les données du formulaire au serveur à travers une entête. Pour simplifier, on va dire que les données et la page ne font qu'un. Les informations ne seront donc pas visibles sur l'URL. La méthode **POST** permet aussi de poster un volume de données plus important que celui transmis par **GET**. D'ailleurs on peut même poster un fichier à travers un formulaire via cette méthode. C'est ce que l'on appelle le "Upload" (C'est comme si vous "uploadiez" une vidéo sur Youtube, ou votre photo de profil sur Facebook).

L'attribut **action**

Quand un formulaire est rempli par l'internaute, il est posté au serveur pour le traitement. Il faut donc désigner la page qui se chargera du traitement des données. Cette page est spécifiée dans l'attribut **action**.

Si l'attribut **action** est vide ou n'est pas mentionné, alors c'est la page courante (celle qui contient le formulaire) qui sera la page **action**.

Chapitre 1: Le langage HTML

Les formulaires

Supposons que nous voulons créer un formulaire nommée "inscription" utilisant la méthode "post" et faisant appel à la page "inscription.html" pour assurer le traitement après envoi. Le code HTML ressemblera à ceci:

```
<form name="inscription" method="post" action="inscription.html">  
</form>
```

Si vous exécutez ce code sur le navigateur, rien ne s'affichera. En effet, ce qui sera visible c'est ce que l'on intégrera dans la balise **<form>**.

Que contient donc la balise **<form>**? Elle contient ce que l'on appelle **les champs de formulaire**. Ce sont des objets comme des zones de texte ou des boutons. Nous allons maintenant voir comment intégrer ces champs.

Chapitre 1: Le langage HTML

Les formulaires

Balise `<input>`

La **zone de texte** est le champ de formulaire le plus célèbre. Il permet d'écrire un texte sur une seule ligne (comme le nom ou le login par exemple). Pour créer une zone de texte on fait appel à la balise `<input>`. Il s'agit d'une balise orpheline (pas de `</input>`).

Notez que la plupart des champs de formulaire existants sont déclarés aussi à l'aide de la même balise `<input>`. Pour distinguer une zone de texte d'un bouton par exemple, on fait appel à l'attribut **type**. Détaillons les attributs les plus utilisées pour les **inputs**.

Attribut **type**

L'attribut **type** permet de définir le type de champ à intégrer. Voici les valeurs à mettre pour cet attribut et qui permettent d'avoir les champs les plus fréquents:

- **text**: Permet d'avoir une zone de texte normale (adapté pour les noms, emails, login...)
- **password**: Permet d'avoir une zone de mot de passe où les caractères sont masqués.

Chapitre 1: Le langage HTML

- **radio**: Pour créer un bouton radio. Il s'agit d'un petit cercle qu'on peut cocher et qui permet de faire un choix unique. Le fait de cocher un autre bouton radio décoche le premier.
- **checkbox**: Permet de créer une case à cocher. C'est un petit carré qu'on peut cocher ou décocher. Il est destiné à faire des choix multiples.
- **file**: Permet de créer le champ de chargement de fichier (comme une pièce jointe d'un email).
- **hidden**: Pour créer un champ caché. C'est comme une zone de texte mais qui n'est pas visible sur le navigateur. Il sert à stocker provisoirement des données jusqu'à la phase de traitement.
- **button**: Permet de créer un bouton simple. A la base, ce bouton ne fait aucune opération. Il faut donc le programmer (généralement avec Javascript).
- **reset**: Permet de créer un bouton d'annulation. Si le formulaire contient des champs qu'on a déjà saisi, ce bouton permet de tout initialiser.
- **submit**: C'est le bouton d'envoi. C'est lui qui permet de poster le formulaire vers la page définie dans l'attribut **action** de la balise **<form>**.

Chapitre 1: Le langage HTML

Attribut name

L'attribut **name** permet de donner un nom au champ de formulaire. Le nom sert d'identifiant du champ et doit être unique.

Attribut value

L'attribut **value** sert à définir la valeur par défaut d'un champ. S'il s'agit d'une zone de texte, le fait de déclarer l'attribut **value** force le navigateur à initialiser ce champ avec le texte faisant office de valeur de l'attribut. Si le champs est un bouton, alors le texte de l'attribut **value** constitue l'étiquette (ou label). C'est le texte visible sur le bouton. Pour les boutons radio ou les cases à cocher, le texte de l'attribut **value** n'est pas visible sur le navigateur mais il est très utile (si on fait appel à des programmes comme Javascript ou PHP).

Attribut size

L'attribut **size** contient un entier comme valeur. Il définit la largeur du champ en caractères. Par défaut, une zone de texte mesure 20 caractères (cela veut dire qu'on peut voir jusqu'à 20 caractères à la fois même si le texte est plus grand).

Chapitre 1: Le langage HTML

Attribut **tabindex**

L'attribut **tabindex** permet de définir l'ordre de tabulation. En pratique, si on veut saisir un grand formulaire, on se déplace d'un champ à un autre en appuyant sur la touche "tabulation" du clavier. Par défaut le curseur se déplace dans l'ordre où sont déclarés les champs. On peut rompre cet ordre en définissant les valeurs de **tabindex**. La valeur est un entier qui commence de 1. Le curseur se déplacera dans l'ordre croissant de **tabindex**.

Attribut **checked**

L'attribut **checked** est un attribut booléen. Explicitement il est déclaré comme ceci: **checked="true"**, mais il suffit de déclarer **checked** tout seul pour que le navigateur sache qu'il est activé. Cet attribut s'applique uniquement sur les boutons radio (**type="radio"**) et les cases à cocher (**type="checkbox"**). S'il est déclaré alors le bouton est automatiquement coché au chargement de la page.

Chapitre 1: Le langage HTML

Balise <textarea>

La balise **<textarea>** permet de définir un espace de texte. Un espace de texte est une grande zone de texte qui permet d'écrire des paragraphes entiers.

Les attributs **name** et **tabindex** sont aussi applicables sur l'espace de texte mais, en plus, il dispose des attributs suivants:

- **cols**: définit la largeur en caractères de l'espace de texte. Elle désigne combien de caractère une ligne peut-elle contenir.

Le fait de dépasser la largeur définie entraîne un retour automatique à la ligne.

- **rows**: définit la hauteur de l'espace de texte. La hauteur désigne combien de ligne on peut voir à la fois. Si le texte dépasse le nombre de lignes défini, alors une barre de défilement qui permet d'atteindre le reste des lignes apparaît.

Chapitre 1: Le langage HTML

Les formulaires

```
<form name="inscription" method="post" action="">
    civilité<br>
    <input type="radio" name="civilite"> Mlle<br>
    <input type="radio" name="civilite"> Mme<br>
    <input type="radio" name="civilite"> M.<br>
    Nom et prénom<br>
    <input type="text" name="nom" value=""><br>
    Email<br>
    <input type="text" name="email" value=""><br>
    Login<br>
    <input type="text" name="login" value=""><br>
    Mot de passe<br>
    <input type="password" name="pass" value=""><br>
    Vous être<br>
    <select name="niv">
        <option>Etudiant</option>
        <option>Fonctionnaire</option>
        <option>Employé au secteur privé</option>
    </select><br>
    <input type="button" name="envoyer" value="S'inscrire">
</form>
```

Civilité

- Mlle
- Mme
- M.

Nom et prénom

Email

Login

Mot de passe

Vous être

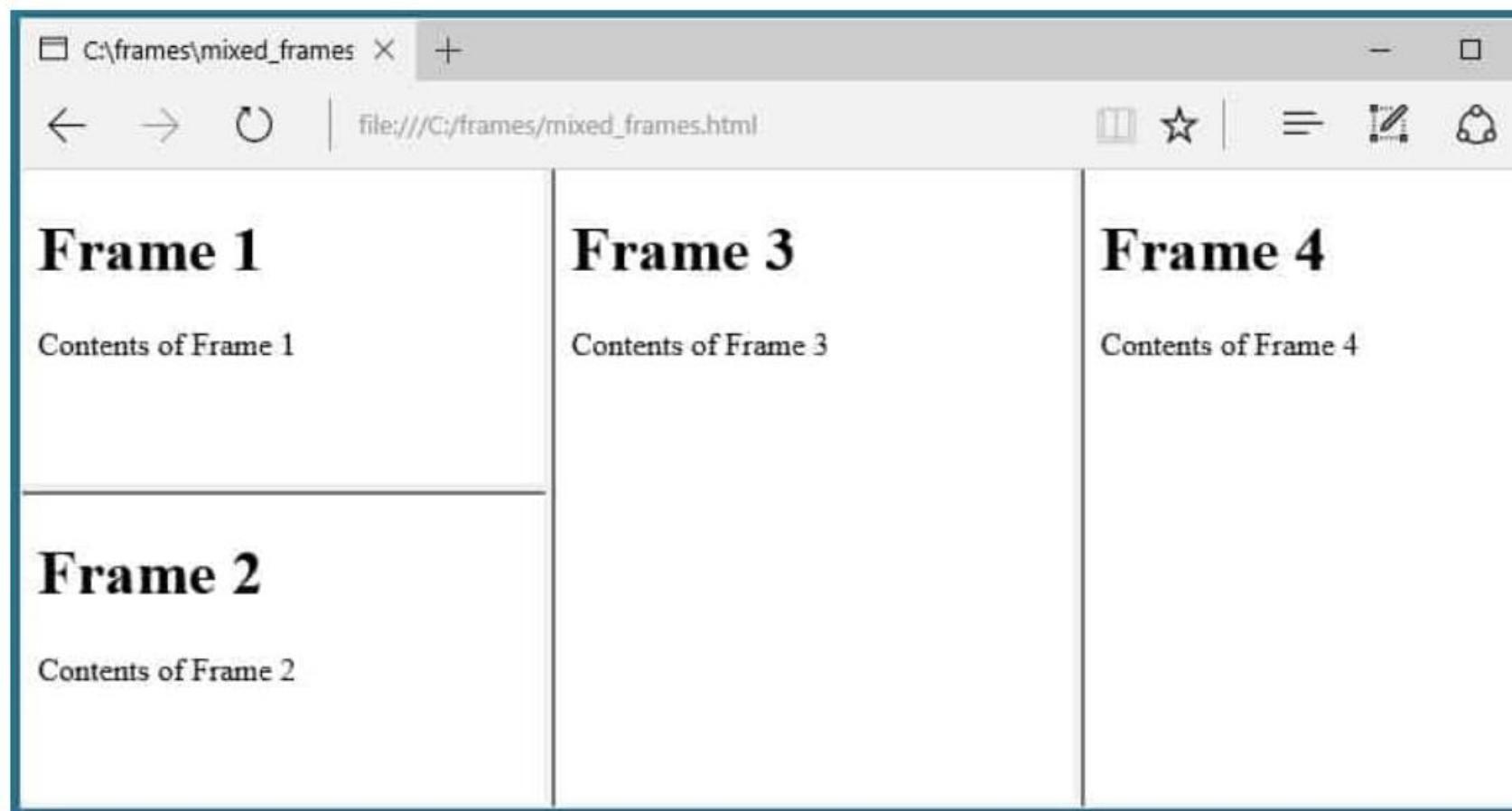
▼

S'inscrire

Chapitre 1: Le langage HTML

```
<frameset cols="*,*,*">
  <frameset rows="*,*">
    <frame src="frame_1.html">
    <frame src="frame_2.html">
  </frameset>
  <frame src="frame_3.html">
  <frame src="frame_4.html">
</frameset>
```

Les framesets



Chapitre 1: Le langage HTML

Frame 1

Contents of Frame 1

Frame 2

Contents of Frame 2

- [Load frame_1.html](#)
- [Load frame_2.html](#)
- [Load frame_3.html](#)
- [Load frame_4.html](#)

Frame 3

Contents of Frame 3

Frame 4

Contents of Frame 4

```
<frameset rows="150px,*">
  <frame noresize src="frame_1.html" marginheight="15">
  <frameset cols="20%,*,20%">
    <frame src="frame_2.html" frameborder="0">
    <frame src="frame_3.html" name="mid_col" frameborder="0">
    <frame src="frame_4.html" frameborder="0">
  </frameset>
</frameset>
```

Chapitre 1: Le langage HTML

```
<!DOCTYPE html>
<html>
<body>
    <h1>Frame 2</h1>
    <p>Contents of Frame 2</p>
    <ul>
        <li><a href="frame_1.html" target="mid_col">Load frame_1.html</a></li>
        <li><a href="frame_2.html" target="mid_col">Load frame_2.html</a></li>
        <li><a href="frame_3.html" target="mid_col">Load frame_3.html</a></li>
        <li><a href="frame_4.html" target="mid_col">Load frame_4.html</a></li>
    </ul>
</body>
</html>
```

Chapitre 2: Le langage CSS

Langage CSS (Cascading Style Sheets)

1. CSS C'est quoi?
2. Comment intègre-t-on du code CSS?
3. Couleurs de textes et propriétés agissant sur le fond
4. Les polices
5. Styles de textes, marges, dimensions
6. Les bordures
7. Gestion des positions et transformation des éléments
8. Masquage, opacité et débordement
9. Sélecteurs spéciaux
10. Pseudo-classes et pseudo-éléments
11. Les Transformations CSS
12. Les transitions

Chapitre 2: Le langage CSS

Limites du HTML

Dans le cours de [HTML](#) nous avons vu que ce langage sert à présenter le contenu de telle façon à ce qu'il soit adapté pour le Web sans pour autant pouvoir le gratifier d'un bon design. Bien qu'il existe des balises, associées à certains de leurs attributs, qui peuvent appliquer des retouches modestes et mettre ainsi fin au rendu par défaut appliqué par HTML.

Cela nous mène à croire que HTML a des limites puisqu'il ne peut pas appliquer un visuel attrayant au contenu qu'il est capable d'intégrer. Or, W3C a toujours été clair sur ce point: "HTML est un langage de description et de présentation de contenu Web", c'est tout. Il faut donc une alternative si on veut avoir une page Web réussie et qui plait aux internautes. C'est là où intervient le CSS.

Avant le CSS

Le langage HTML est né avec l'apparition du Web vers 1989. A l'époque, CSS n'existe pas et donc les créateurs de pages Web étaient contraints de tout faire avec seulement du HTML, à savoir la création du contenu et la présentation du design. Après un moment, un problème commence à surgir, le code des pages Web devient de plus en plus complexe, car il fallait mettre beaucoup de balises pour l'intégration d'une part et de mise en forme d'une autre part. Le CSS est alors apparu.

Chapitre 2: Le langage CSS

CSS c'est quoi?

CSS désigne **Cascading Style Sheets** (pour Feuilles de style en cascade). Il s'agit d'un langage de style dont la syntaxe est extrêmement simple mais son rendement est remarquable. En effet, le CSS s'intéresse à la mise en forme du contenu intégré avec du HTML.

On peut créer une page Web entière avec HTML seulement, même si le design ne serait pas au niveau de nos attentes. Par contre on ne peut absolument pas réussir une page Web avec CSS seulement. CSS ne fait que mettre en forme le contenu décrit par HTML.

Comme pour HTML, CSS a commencé petit et grandit au fil des versions. La première version a vu le jour vers 1996. Il s'agissait de CSS1, suivie de CSS2 qui était le plus populaires jusqu'à l'apparition de HTML5 qui a intégré de nouvelles fonctionnalités qui ont emmené à développer la version 3 de CSS (dite CSS3) qui est le standard le plus utilisé actuellement.

Chapitre 2: Le langage CSS

CSS c'est quoi?

CSS est un langage coté client, c'est à dire que sa syntaxe est comprise par le navigateur qui l'exécute avec le HTML. Ceci peut conduire à des problème de compatibilité, puisque parfois, les navigateurs ne comprennent pas toujours certains codes CSS de la même manière.

Chapitre 2: Le langage CSS

La syntaxe CSS à quoi ressemble-t-elle?

La syntaxe CSS est tellement simple et intuitive que le simple fait de lire le code nous donne une idée sur ce qu'il peut faire même si nous étions des novices. Il faut savoir qu'il y a plusieurs façons d'intégrer un style CSS. Dans l'exemple qui suit je vais vous montrer à quoi ressemble du code CSS en général:

```
form{
    margin:0;
    padding:0;
}
input{
    outline:none;
}
input[name="trecherche"]{
    border:none;
    font-family:eras, "Century Gothic", verdana, sans-serif;
    font-size:14pt;
    margin-bottom:3px;
    margin-left:3px;
}
#tdrecherche{
    background-image:url("../images/recherche_02.png");
    width:161px;
    background-repeat:no-repeat;
}
```

Chapitre 2: Le langage CSS

Comment déclarer un style CSS?

En règle générale, une instruction CSS est déclarée sous la forme:

propriété:valeur.

Il y a aussi le point virgule (;) qui sert à séparer deux instructions CSS.

Un style déclaré par le biais d'un **attribut de style local** présente un inconvénient majeur. En effet, il ne s'applique qu'à la balise sur laquelle il est déclaré. Or, sur une page Web, il existe plusieurs éléments qui se ressemblent (décrits par la même balise en général). Il est donc judicieux que ces éléments aient le même style, et par conséquent on serait obligé de redéclarer le même style local autant de fois qu'il y a d'éléments similaires.

Chapitre 2: Le langage CSS

Comment déclarer un style CSS?

Attribut de style local

Pour faire simple, prenons le code suivant:

```
<font color="red" face="verdana" size="3">  
    Bonjour à tous!  
</font>
```

Bonjour à tous!

Pourtant, il existe une autre syntaxe qui permet de réussir exactement le même affichage. La voilà:

```
<font style="color:red; font-family:verdana; font-size:12pt">  
    Bonjour à tous!  
</font>
```

Bonjour à tous!

Chapitre 2: Le langage CSS

Comment déclarer un style CSS?

Attribut de style local

```
<font style="color:red; font-family:verdana; font-size:12pt">  
    Bonjour à tous !  
</font>
```

- Les attributs HTML **color**, **face** et **size** ont disparu et ils ont été remplacés par un seul et unique attribut; il s'agit de **style**.
- Comme valeur, **style** reçoit une séquence d'instructions qui sont respectivement:
 - **color:red** qui désigne la couleur rouge,
 - **font-family:verdana** qui signifie la police verdana
 - **size:12pt** qui spécifie la taille de police 12pt (équivalente en HTML à **size="3"**).

La séquence d'instructions qui représente la valeur de l'attribut **style** est en fait, du CSS.

Chapitre 2: Le langage CSS

Comment déclarer un style CSS?

Attribut de style global

- Au lieu de déclarer les styles sur la balise ****, on les a regroupé à l'entête du document (balise **<head>**), mais pas n'importe où dans l'entête.
- La balise **<style>** se déclare toujours dans la balise **<head>** et contient donc exclusivement de la syntaxe CSS.
- Un document HTML peut contenir du code HTML et du code CSS. Il suffit d'indiquer au navigateur où commence le CSS et où finit-il. C'est le rôle de la balise **<style>**.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <style>
      font{
        color:red;
        font-family:verdana;
        font-size:12pt;
      }
    </style>
  </head>
  <body>
    <font>
      Bonjour à tous!
    </font>
  </body>
</html>
```

Chapitre 2: Le langage CSS

Comment déclarer un style CSS?

Attribut de style global

Puisque les styles sont exportés loin de la balise, alors comment le navigateur saurait-il si on souhaitait l'appliquer sur telle ou telle balise?

C'est le rôle du **sélecteur de balise** représenté cette fois par le mot **font**.

Attention, il ne s'agit pas d'une balise car elle ne contient pas les chevrons ouvrant et fermant, il s'agit juste d'un sélecteur qui a le même nom que la balise ciblée.

Les accolades sont là pour englober l'ensemble des styles à appliquer sur les balises présentes dans le document et qui portent le même nom que le sélecteur déclaré (balises **** dans ce cas).

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <style>
      font{
        color:red;
        font-family:verdana;
        font-size:12pt;
      }
    </style>
  </head>
  <body>
    <font>
      Bonjour à tous!
    </font>
  </body>
</html>
```

Chapitre 2: Le langage CSS

Comment déclarer un style CSS?

La feuille de style en cascade

Si la première méthode sert à définir un style en local sur la balise concernée, et la deuxième permet d'englober les styles pour qu'ils s'appliquent sur tous les éléments semblables dans le document, il existe une troisième méthode qui est la plus pratique et la plus utilisée. D'ailleurs le nom du langage fait allusion à cette méthode, il s'agit de la **feuille de style en cascade**.

La feuille de style en cascade permet de définir des styles qui s'appliqueront, non pas sur le contenu d'un seul document, mais sur le contenu de tous les documents qui constituent votre projet Web (site ou application Web). Cette technique permet d'uniformiser le design sur toutes les pages en gratifiant les éléments similaires du même style. Ainsi, si le grand titre d'une page était bleu, il le serait aussi sur les autres pages.

La feuille de style est un fichier texte qui contient uniquement du code CSS. Par conséquent il a l'extension ".css".

Chapitre 2: Le langage CSS

Comment déclarer un style CSS?

La feuille de style en cascade

Supposons que nous disposons d'une feuille de style du nom de **style.css**, et qui contient le code suivant:

```
font{
    color:red;
    font-family:verdana;
    font-size:12pt;
}
h1{
    color:blue;
}
```

Nous avons déclaré deux sélecteurs de balises **font** et **h1**, chacun avec un style qui s'appliquera sur les balises correspondantes. Maintenant on veut appliquer ces styles à une page Web.

Chapitre 2: Le langage CSS

Comment déclarer un style CSS?

La feuille de style en cascade

On va donc faire appel à la feuille de style au sein de la page Web souhaitée dont le code ressemblerait à ceci:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <link rel="stylesheet" type="text/css" href="style.css" />
  </head>
  <body>
    <h1>Exemple de CSS</h1>
    <font>
      Voici à quoi ressemble l'effet de la feuille de style sur ce document.
    </font>
  </body>
</html>
```

Chapitre 2: Le langage CSS

Comment déclarer un style CSS?

La feuille de style en cascade

La balise `<link>` est une balise orpheline et c'est elle qui se charge d'associer la feuille de style au document HTML. Ses attributs sont:

- Attribut **rel**: permet de définir le type de ressource à intégrer au document. Dans ce cas nous avons spécifié **stylesheet**, c'est à dire **feuille de style**.
- Attribut **type**: c'est un attribut facultatif qui permet de spécifier le type de la ressource à intégrer. Dans ce cas il s'agit de **text/css** qui désigne: document texte écrit en CSS.
- Attribut **href**: cet attribut permet de spécifier le chemin (relatif ou absolu) de la feuille de style à appliquer. Dans l'exemple, nous avons supposé que le document HTML et la feuille de style sont dans le même emplacement.

```
<head>
    <meta charset="UTF-8" />
    <link rel="stylesheet" type="text/css" href="style.css" />
</head>
```

Chapitre 2: Le langage CSS

Comment déclarer un style CSS?

La feuille de style en cascade

L'exécution des codes précédents donne:

Exemple de CSS

Voici à quoi ressemble l'effet de la feuille de style sur ce document.

La troisième méthode qui centralise tous les styles en un seul endroit est la plus recommandée , ce qui garanti l'uniformité du design sur tout le projet et aussi, facilite les modifications éventuelles du code CSS.

Chapitre 2: Le langage CSS

Comment déclarer un style CSS?

1. Peut-on déclarer deux balises `<link>` pour appeler deux feuilles de styles CSS différentes sur le même document?
2. Cette balise
`<link href="style.css" />`
ne fonctionnera pas à cause du:
3. Au cas où un même style a été décrit en tant que style local et dans une feuille de style CSS. Lequel des deux sera appliqué à la balise concernée?
4. Peut-on appeler une feuille de style CSS et en même temps se servir de la balise `<style>` dans la page?

Chapitre 2: Le langage CSS

Couleurs de textes et propriétés agissant sur le fond

Couleur de texte

La propriété **color** permet de spécifier la couleur à appliquer à un texte.

Le texte peut être placé dans n'importe quelle balise (**<div>**, ****, **<header>**...).

Pour décrire la couleur il existe plusieurs méthodes:

- **Nom de la couleur en anglais**
- **Code hexadécimal**
- **Code décimal (ou RGB)**: En CSS, une autre méthode pour définir les couleurs existe.

Il s'agit d'indiquer le code décimal de la couleur. Le principe reste le même que pour le codage hexadécimal, mais les codes sont exprimés à la base décimale dont les nombres seront, cette fois, compris entre 0 et 255.

A titre d'exemple, le code du rouge est : **rgb(255,0,0)**.

Les valeurs dans les parenthèses représentent respectivement les couleurs primaires rouge, vert et bleu.

Chapitre 2: Le langage CSS

Couleurs de textes et propriétés agissant sur le fond

Couleur de texte

- **Code RGBA:** Depuis la version 3 de CSS, une nouvelle méthode pour décrire les couleurs est apparue.

Il s'agit de **RGBA**. Cette notation permet de prendre en compte le degré de transparence (le A pour Alpha).

Les trois premiers paramètres de **RBGA** désignent les couleurs primaires et le quatrième indique le degré de transparence de la couleur appliquée.

Il est compris entre 0 (pour totalement transparent) et 1 (pour totalement opaque), et peut avoir deux chiffres après la virgule.

Par exemple, **rgba(255,0,0,0.5)** désigne un rouge translucide.

Chapitre 2: Le langage CSS

Couleurs de textes et propriétés agissant sur le fond

Couleur de texte

Ce code applique la couleur orange sur les textes contenus dans les balises `<h1>`, gris pour les `<div>`, bleu pour les `` et noir translucide pour les `<footer>`.

```
h1{  
    color:orange;  
}  
div{  
    color:#888888;  
}  
span{  
    color:rgb(0,0,255);  
}  
footer{  
    color:rgba(0,0,0,0.5);  
}
```

Chapitre 2: Le langage CSS

Propriétés d'arrière plan

CSS agit sur les arrière plan en leur appliquant différents style dont voici les plus utilisés:

Propriété background-color

La propriété **background-color** permet d'appliquer une couleur d'arrière plan à l'objet HTML souhaité. Pratiquement, tous les objets HTML peuvent avoir un arrière plan. La valeur de cette propriété correspond à la couleur que l'on peut décrire par l'une des 4 méthodes expliquées.

Si on souhaite colorer le fond de la page Web en entier alors on utilise le sélecteur de balise **body**:

```
body{  
    background-color:#cccccc;  
}
```

Ce code applique un gris clair à l'arrière-plan du document qui appelle la feuille de style.

Chapitre 2: Le langage CSS

Propriétés d'arrière plan

Propriété background-image

La propriété **background-image** sert à insérer une image en arrière plan de l'objet souhaité. En utilisant la syntaxe suivante:

background-image:url("chemin_de_l_image").

Le chemin de l'image peut être absolu ou relatif à partir de l'emplacement où est déclaré le style.

Il faut noter que l'image est appliquée en arrière plan avec sa taille réelle. Si elle est plus grande que l'objet qui l'accueille alors elle sera tronquée et si elle est plus petite alors elle sera répétée jusqu'à ce que tout le fond de l'objet soit couvert.

```
body {  
    background-image:url ("images/footerlogo.png") ;  
}
```

Chapitre 2: Le langage CSS

Propriétés d'arrière plan

Propriété background-repeat

La propriété **background-repeat** permet de contrôler la répétition de l'image au cas où celle-ci est plus petite que l'objet qui l'accueille. On peut lui donner les valeurs suivantes:

- **repeat**: C'est la valeur par défaut. Elle permet de répéter l'image sur la largeur et la hauteur de l'objet.
- **repeat-x**: Sert à répéter l'image seulement sur la largeur de l'objet.
- **repeat-y**: Sert à répéter l'image seulement sur la hauteur de l'objet.
- **no-repeat**: Avec cette valeur, l'image est appliquée une seule fois sans répétition.
Par défaut elle sera placée sur le coin haut à gauche de l'obiet.

```
body {  
    background-image:url ("images/footerlogo.png");  
    background-repeat:repeat-x;  
}
```

Chapitre 2: Le langage CSS

Propriétés d'arrière plan

Propriété **background-position**

La propriété **background-position** permet de placer l'image de l'arrière plan à l'endroit souhaité de l'objet. Par défaut, la première image qui s'applique en arrière-plan est placée en haut à gauche, puis les autres images la suivent pour combler tout l'espace disponible.

La valeur que l'on peut donner à cette propriété peut être exprimé en pourcentage (%) ou par les mots clé: **top**, **center**, **bottom**, **left** et **right**.

Par exemple si on souhaitait placer l'image d'arrière plan au juste milieu de l'objet, la propriété CSS ressemblerait à ceci:

background-position:center center ou **background-position:50% 50%**.

Nous avons exprimé deux valeurs séparées par un espace, une pour la largeur et l'autre pour la hauteur.

Chapitre 2: Le langage CSS

Propriétés d'arrière plan

Propriété background-attachment

La propriété **background-attachment** permet de fixer l'arrière-plan de l'objet si celui là contient une barre de défilement. Elle peut avoir deux valeurs:

scroll qui est la valeur par défaut et qui laisse défiler l'arrière plan avec l'avant plan si le client entraîne la barre de défilement qui apparaît sur l'objet et **fixed** qui permet de figer l'arrière plan pendant que l'avant plan glisse en dessus.

Pour mieux comprendre à quoi cela ressemble on va exécuter le code suivant:

```
body{  
    background-image:url("test.png");  
    background-repeat:no-repeat;  
    background-position:50% 50%;  
    background-attachment:fixed;  
}
```

Chapitre 2: Le langage CSS

Propriétés d'arrière plan

La sous propriété **linear-gradient**

La sous propriété **linear-gradient** s'applique à la propriété **background** pour générer un arrière plan en dégradé de couleur. Elle possède trois paramètres:

- **Direction du dégradé:** qui peut avoir les valeurs comme **to bottom** (vers le bas), **to left** (vers la gauche), **to left bottom** (incliné vers la gauche et vers le bas) etc... Par défaut elle vaut la valeur **to bottom**.
- **Couleur de début:** Précise à partir de quelle couleur le dégradé sera engendré. La couleur est spécifiée grâce à l'une des 4 méthodes déjà vues.
- **Couleur de fin:** Précise la deuxième couleur du dégradé.

Entre la couleur de début et la couleur de fin le navigateur générera des nuances assurant un passage progressif entre les deux couleurs.

Chapitre 2: Le langage CSS

Propriétés d'arrière plan

La sous propriété linear-gradient

Exemple:

```
body{  
    background:linear-gradient(to bottom, #DDDDDD, #FFFFFF);  
}
```

Chapitre 2: Le langage CSS

Propriétés d'arrière plan

La sous propriété **linear-gradient**

- La syntaxe de la sous propriété **linear-gradient** est légèrement modifiée pour les anciennes versions de certains navigateur afin de la rendre compatible. En effet, on la précède par un préfixe connu sous le nom de **préfixe vendeur**.
Ce préfixe fait référence aux différents moteurs de rendu existants.

Voici la liste des préfixes vendeur connus:

- **-moz-**: signifie Mozilla et fait référence à son moteur de rendu Gecko.
- **-webkit-**: signifie le moteur de rendu Webkit (utilisé par Google Chrome et Safari).
- **-o-**: signifie Opera et son moteur de rendu Presto.
- **-ms-**: signifie Microsoft et fait référence à son moteur de rendu Trident.

Chapitre 2: Le langage CSS

Propriétés d'arrière plan

Exemple:

```
body{  
    background:linear-gradient(to bottom, #FF0, #F00, #F0F);  
}
```

Ce qui donne:



Chapitre 2: Le langage CSS

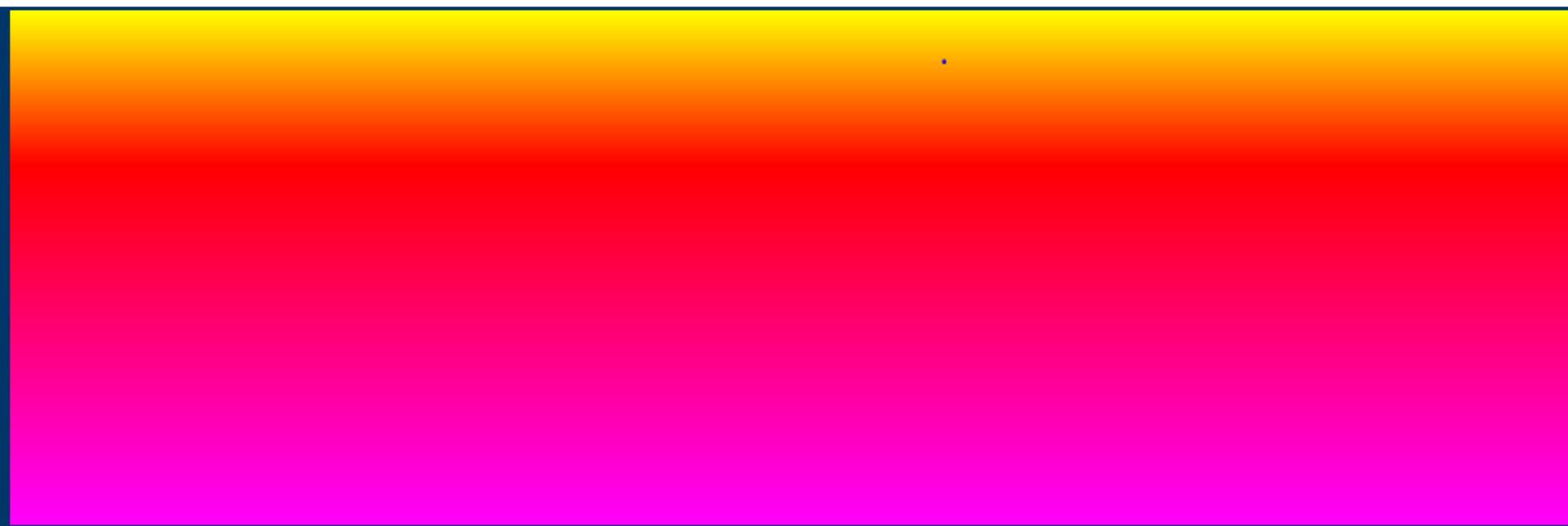
Propriétés d'arrière plan

Exemple:

```
body{  
    background:linear-gradient(to bottom, #FF0 0%, #F00 30%, #FOF 100%);  
}
```

Dans ce cas on a précisé que le jaune s'applique au début. A 30% de la zone on applique le rouge. Le dégradé entre le jaune et le rouge s'étale donc sur 30% de la hauteur de la zone colorée. Ensuite le magenta s'applique à la fin et le dégradé entre le rouge et le magenta s'applique donc sur 70% de la zone colorée.

Ce qui donne:



Chapitre 2: Le langage CSS

- Laquelle de ces propriétés CSS nous permet de changer la couleur de fond d'un élément ?
 - 1.color
 - 2.background-color
 - 3.background-image
- Par défaut, une image de fond se répète...
 - 1.Horizontalement
 - 2.Pas du tout
 - 3.Horizontalement et verticalement
- A quoi sert un préfixe vendeur ?
 - 1.A vendre plus pour un site e-commerce
 - 2.A forcer la compatibilité d'une propriété CSS pour certaines versions de certains navigateurs
 - 3.A créer des dégradés pour tous les navigateurs

Chapitre 2: Le langage CSS

Les polices

La propriété CSS font-family

```
body{  
    font-family: "Source code pro", Verdana, sans-serif;  
}
```

La propriété CSS font-family va nous permettre de définir la police de notre texte.

Vous devez savoir que tous les navigateurs, selon la version possédée par vos visiteurs, ne supportent pas tous les mêmes polices.

Pour cette raison, nous indiquerons toujours plusieurs noms de police à utiliser en valeur de la propriété font-family, en commençant par celle souhaitée, et en séparant chaque valeur par une virgule.

Ainsi, le navigateur va lire les différentes polices renseignées dans font-family dans l'ordre et utiliser la première qu'il supporte.

Si vous renseignez un nom de police qui contient des espaces, vous devrez le mettre entre guillemets ou apostrophes.

Chapitre 2: Le langage CSS

Les polices

Web safe fonts et familles génériques

Famille générique	Police
Serif	Times New Roman, Georgia
Sans-serif	Arial, Verdana
Monospace	Courier New, Lucida Console
Cursive	Comic sans MS

Chapitre 2: Le langage CSS

Les polices

Les propriétés de type font

Les propriétés CSS de type font- vont nous permettre de modifier l'apparence de notre police d'écriture, et donc de nos textes.

Nous allons par exemple pouvoir transformer la taille, le poids ou le style de notre police d'écriture.

Dans cette partie, nous allons voir les propriétés de type font- les plus utilisées :

- font-size pour modifier la taille de nos textes ;
- font-style pour modifier le style de nos textes ;
- font-weight pour modifier le poids de nos textes.

Chapitre 2: Le langage CSS

La propriété CSS font-size

La propriété CSS font-size va nous permettre de modifier la taille de notre police d'écriture.

Cette propriété va accepter deux grands types de valeurs :

des tailles de polices **absolues ou fixes** et des tailles **relatives ou variables**.

Les valeurs de type absolu vont souvent être exprimées en px (pixels) ou éventuellement en pt (points).

- Une taille absolue est fixée : elle ne bougera jamais.
- Les valeurs de type relatif vont être exprimées en % (pourcentage).

L'intérêt des valeurs relatives est qu'elles vont s'adapter relativement à certains paramètres : soit par rapport aux préférences enregistrées dans le navigateur par un visiteur, soit par rapport à la taille d'un élément parent.

- la valeur « inherit », ce qui veut dire que l'élément ciblé héritera de la valeur de son parent. Cette valeur fonctionne avec beaucoup de propriétés CSS et peut être utile pour annuler un comportement par défaut ou en cas de conflit de styles. Nous y reviendrons plus tard.

Chapitre 2: Le langage CSS

La propriété CSS font-style

La propriété CSS font-style va nous permettre de forcer le style de notre police.

Nous allons par exemple pouvoir mettre nos textes en italique.

La propriété font-style accepte quatre valeurs différentes :

- normal (par défaut) ;
- italic (italique) ;
- oblique (penché) ;
- inherit (l'élément ciblé hérite du style de son élément parent).

Chapitre 2: Le langage CSS

La propriété CSS font-weight

La propriété CSS font-weight va nous permettre de définir le poids d'une police, c'est-à-dire son épaisseur.

Cette propriété peut prendre différentes valeurs :

- normal (valeur par défaut) ;
- Lighter (la police sera plus fine) ;
- bold (la police sera plus épaisse) ;
- bolder (la police sera très épaisse) ;
- une centaine entre 100 (police très fine) et 900 (police très épaisse). 400 correspond à la valeur « normal » et 700 à « bold » ;
- inherit (l'élément hérite du style de son parent) ;
- initial (définit la propriété sur sa valeur d'origine).

Chapitre 2: Le langage CSS

Les propriétés CSS de type « text- »

Les propriétés CSS de type text- vont nous permettre de changer la mise en forme de nos textes et leur apparence.

A la différence des propriétés de type font-, les propriétés de type text- ne sont pas dépendantes de la police utilisée.

Dans cette partie, nous allons étudier les propriétés CSS de type text- suivantes :

- La propriété text-align (gère l'alignement) ; (left, right, center)
- La propriété text-transform (gère la mise en majuscules / minuscules) ;
- La propriété text-decoration (gère la décoration) ;
- La propriété text-indent (gère l'indentation) ;
- La propriété text-shadow (gère les ombres).

Chapitre 2: Le langage CSS

Les propriétés CSS de type « text-transform »

La propriété CSS text-transform va nous permettre de transformer un texte ou une partie d'un texte en majuscules ou en minuscules.

Cette propriété nous permet de choisir parmi cinq valeurs :

- Lowercase : Met tout le texte en minuscules ;
- Uppercase : Met tout le texte en majuscules ;
- Capitalize : Met la première lettre de chaque mot en majuscule ;
- Inherit : Hérite de la valeur de l'élément parent ;
- None : Pas de transformation du texte. Utile pour annuler une transformation par défaut donnée par héritage.

<h1>Propriétés CSS "text-</h1>

```
h1{  
    text-transform: uppercase;  
}
```

PROPRIÉTÉS CSS "TEXT-

Chapitre 2: Le langage CSS

Les propriétés CSS de type « text-decoration »

La propriété CSS text-decoration va nous permettre d'ajouter ou d'enlever des décos à nos textes.

Cette propriété accepte six valeurs différentes :

- Underline : Le texte sera souligné ;
- Overline : Une ligne apparaît au dessus du texte ;
- Line-through : Le texte sera barré ;
- Inherit : Hérite de la valeur de l'élément parent ;
- Initial : Utilise la valeur par défaut de la propriété ;
- None : Pas de décoration.

```
a{  
    text-decoration: none;  
}
```

```
>Un <a href="http://wikipedia.org">lien</a>
```

Un lien

Chapitre 2: Le langage CSS

Les propriétés CSS de type « text-indent »

La propriété CSS text-indent va nous permettre de gérer l'indentation d'un texte. Pour rappel, indenter un texte, c'est le décaler vers la droite.

La propriété text-indent accepte aussi bien des valeurs absolues (en px par exemple) que des valeurs relatives (en % par exemple). On peut également lui attribuer des valeurs négatives afin de décaler un texte vers la gauche.

```
h1{  
    text-indent: -50px;  
}
```

propriétés CSS "text-

Chapitre 2: Le langage CSS

Les propriétés CSS de type « text-shadow »

La propriété CSS text-shadow va nous permettre de créer des ombres autour de nos textes, afin que ceux-ci se détachent de l'arrière plan.

On va devoir indiquer quatre valeurs dans un ordre précis à la propriété text-shadow afin que celle-ci fonctionne correctement :

- La projection horizontale de l'ombre (en px) ;
- La projection verticale de l'ombre (en px) ;
- Le rayon de propagation de l'ombre (le « radius », en px) ;
- La couleur de l'ombre. Accepte les mêmes valeurs que la propriété color.

Des valeurs positives pour les projections horizontales et verticales de l'ombre procure un ombre qui sera projetée en bas à droite du texte. En indiquant des valeurs négatives, on peut modifier l'orientation.

```
/*Ombre gris-foncé en bas à droite*/
h1{
    text-shadow: 1px 1px 4px #444;
}
```

Chapitre 2: Le langage CSS

- Pourquoi doit-on préciser plusieurs valeurs pour la propriété font-family ?

1- Pour laisser la possibilité à l'utilisateur de choisir celle qu'il préfère

2- Car certaines polices peuvent ne pas être lues par un navigateur

3- Il n'est pas nécessaire de préciser plusieurs valeurs

- Quelle est la différence entre les valeurs de type RGB et RGBA

1- RGB utilise des notations hexadécimales, ce qui n'est pas le cas de RGBA

2- RGB gère la transparence à l'inverse de RGBA

3- RGBA gère la transparence à l'inverse de RGB

- Combien faut il donner de valeurs à text-shadow ?

2 valeurs

3 valeurs

4 valeurs

Chapitre 2: Le langage CSS

Marges intérieures et marges extérieures

En CSS, nous allons devoir distinguer deux types de marges : les marges intérieures (le padding) et les marges extérieures (le margin).

Les marges intérieures se trouvent entre le contenu de l'élément et sa bordure. Ainsi, définir une marge intérieure importante va éloigner la bordure de l'élément de son contenu. Si on définit une couleur de fond pour notre élément, celle-ci s'applique également dans l'espace correspondant aux marges intérieures.

Les marges extérieures, au contraire, vont définir l'espace autour d'un élément. Les marges extérieures se trouvent en dehors des bordures d'un élément et servent généralement à éloigner un élément d'un autre. Comme les marges extérieures se trouvent « en dehors » d'un élément, celles-ci ne sont pas affectées par la couleur de fond donnée à un élément.

Chapitre 2: Le langage CSS

Les marges intérieures

Les marges intérieures d'un élément vont pouvoir être définies en CSS grâce à la propriété padding. Cette propriété peut prendre des valeurs absolues (px) ou relatives (%).

Notez que l'on va pouvoir définir des marges intérieures haute, droite, basse et gauche de tailles diverses avec padding-top, padding-right, padding-bottom et padding-left.

```
div{  
    background-color: #269;  
    width: 400px;  
    border: 5px ridge #444;  
    border-radius: 15px;  
    padding-left: 100px;  
}
```

```
<div>  
    <h1>Les marges</h1>  
</div>
```

Les marges

Chapitre 2: Le langage CSS

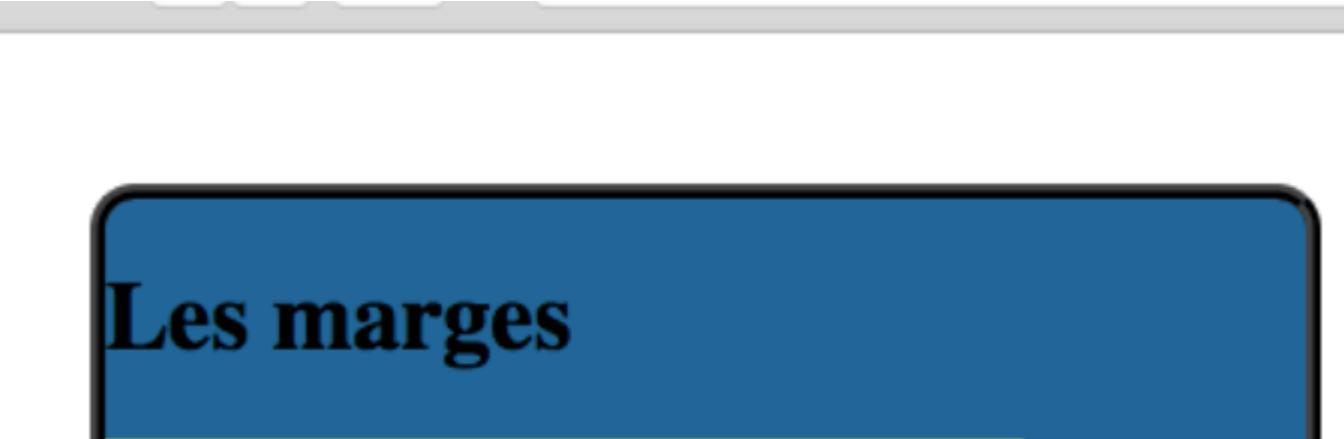
Les marges extérieures

Nous allons pouvoir définir des marges extérieures (« en dehors » de nos éléments) en CSS grâce à la propriété margin.

Cette propriété accepte en valeurs des valeurs absolues, relatives, ainsi que les valeurs inherit ou auto. Cette dernière valeur, auto, va souvent nous être utile pour centrer horizontalement un élément par rapport à un élément parent.

Une nouvelle fois, nous allons pouvoir définir des marges extérieures haute, droite, basse et gauche d'importances diverses grâce à margin-top, margin-right, margin-bottom et margin-left.

```
div{  
    background-color: #269;  
    width: 400px;  
    border: 5px ridge #444;  
    border-radius: 15px;  
    margin-top: 50px;  
    margin-left: 50px;  
}
```



Les marges

Chapitre 2: Le langage CSS

```
div{  
    background-color: #269;  
    width: 400px;  
    border: 5px ridge #444;  
    border-radius: 15px;  
    margin: auto;  
}
```

Les marges

Un premier paragraphe.

Un autre paragraphe

Chapitre 2: Le langage CSS

```
div{  
    background-color: #269;  
    width: 400px;  
    border: 5px ridge #444;  
    border-radius: 15px;  
    margin: auto;  
}
```

Les marges

Un premier paragraphe.

Un autre paragraphe

```
margin: 50px 0px 0px 20px;  
padding: 0px 0px 0px 80px;
```

La première valeur correspond à la marge haute, la seconde à la marge droite, la troisième à la marge basse et la dernière à la marge gauche.

```
padding: 50px 0px;
```

Dans ce cas, la première correspond aux marges haute et basse et la seconde aux marges droite et gauche.

Chapitre 2: Le langage CSS

Les dimensions

Tout élément HTML possède une largeur et une hauteur.

La hauteur par défaut d'un élément HTML est déterminée par son contenu. Ainsi, un paragraphe, selon qu'il occupe une ou deux lignes, ne possèdera pas la même hauteur par défaut.

La largeur d'un élément, en revanche, est avant tout déterminée par son type : en effet, les éléments de type block occuperont par défaut tout l'espace disponible. Les éléments de type inline, en revanche, n'occuperont que la largeur nécessaire à leur contenu.

Pour modifier les dimensions par défaut d'un élément, nous allons utiliser les propriétés `height`, `width`.

Chapitre 2: Le langage CSS

Les dimensions

Vous devez bien vous rappeler que vous ne définissez que la hauteur et la largeur du contenu de l'élément en soi. Les tailles des marges intérieures, extérieures et des bordures viendront s'ajouter à cette hauteur et à cette largeur afin de former la taille totale de l'élément.

Pour régler efficacement la hauteur et la largeur d'un élément, il faut avant tout bien avoir compris le modèle des boîtes.

```
<div>
  <h1>Le modèle des boîtes</h1>
</div>
```

Créer des éléments plus grands que leurs éléments parents est une mauvaise pratique en HTML et en CSS.

```
div{
  background-color: #088;
  width: 500px;
}

h1{
  width:300px;
  background-color: white;
  height: 100px;
  border: 1px solid black;
}
```

Chapitre 2: Le langage CSS

Les bordures

Nous allons pouvoir définir des bordures de couleurs, épaisseurs ou types différents autour de nos éléments HTML en CSS.

L'espace pris par la bordure va se trouver entre la marge intérieure et la marge extérieure d'un élément HTML.

Nous pouvons définir les bordures d'un élément de différentes manières en CSS : soit en utilisant les trois propriétés border-style, border-width et border-color, soit un utilisant directement la notation courte border.

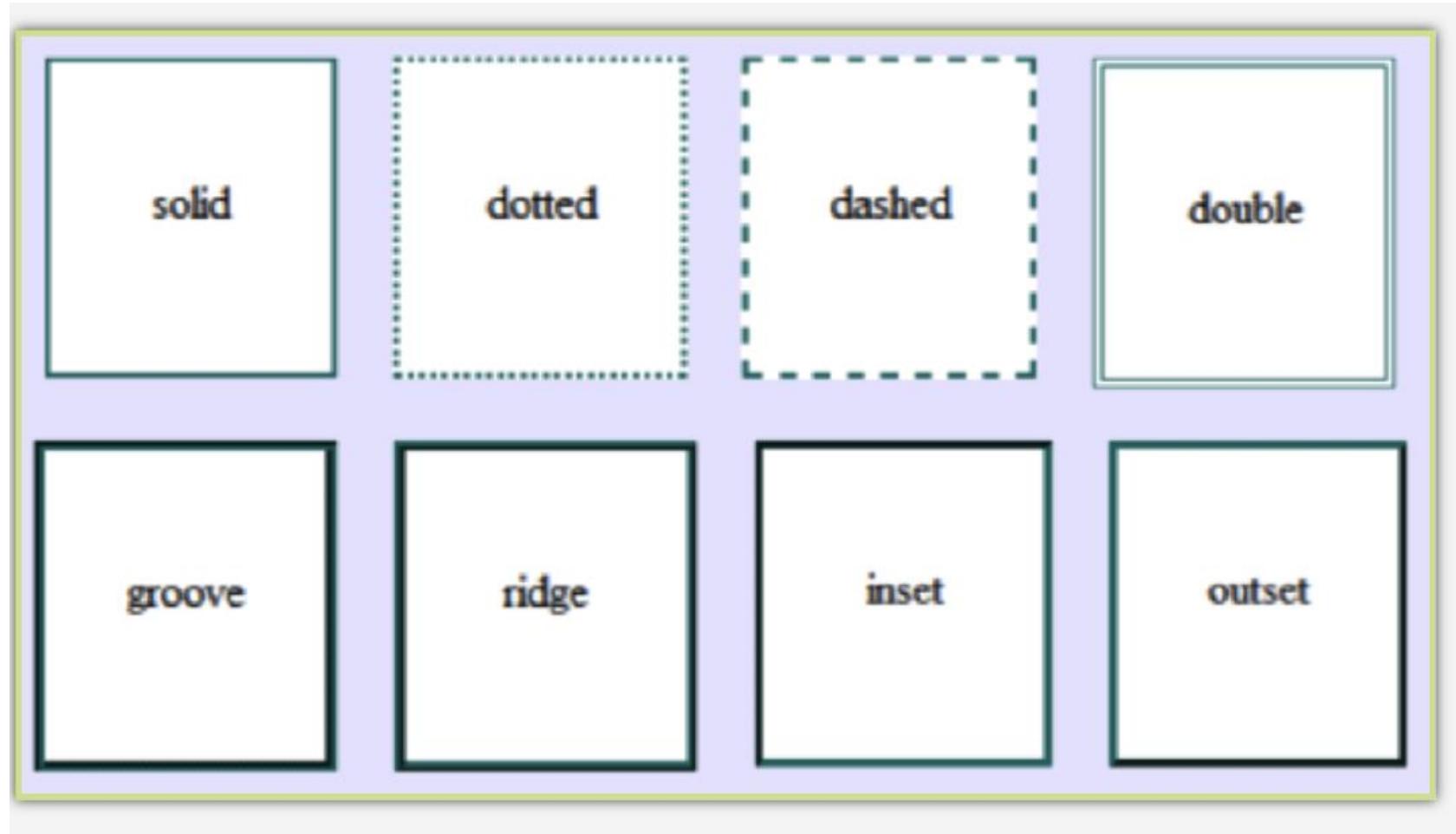
```
div{  
    background-color: #269;  
    width: 400px;  
    border: 5px solid black;  
}
```

```
border-width: 2px;  
border-style: solid;  
border-color: yellow;
```

Chapitre 2: Le langage CSS

Les bordures

Les valeurs de border-style sont les suivantes:



Chapitre 2: Le langage CSS

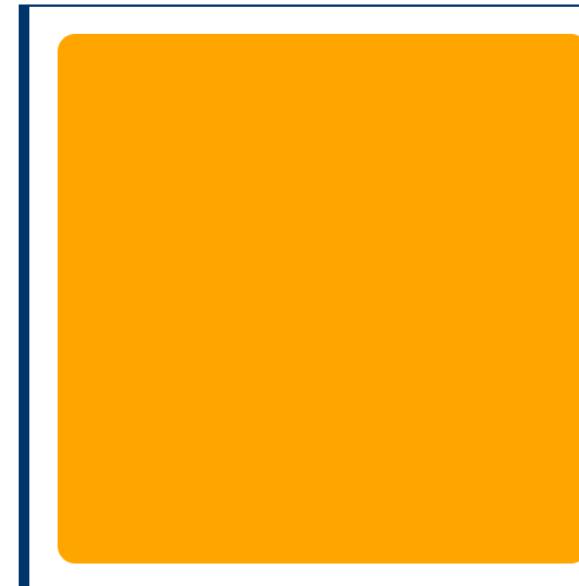
Les bordures

Les bordures arrondies

Nous allons pouvoir créer des bordures arrondies avec la propriété border-radius. Cette propriété va prendre une valeur en pixels qui va correspondre à la valeur du radius de la bordure, et nous allons donc devoir l'utiliser de concert avec la propriété border.

Encore une fois, on va pouvoir définir des bordures plus ou moins arrondies de chaque côté d'un élément HTML, cette fois-ci en utilisant les mots clefs top-right, bottom-right, bottom-left, top-left.

```
div{  
    width:300px;  
    height:300px;  
    background-color:orange;  
    border-radius:10px;  
}
```



Chapitre 2: Le langage CSS

Les bordures

Les bordures arrondies

Code HTML:

```
<div><h1>DIV stylée</h1></div>
```

Code CSS:

```
div{  
    height:100px;  
    background-color:#EEEEEE;  
    text-align:center;  
    padding:20px;  
    border:solid 5px #BBBBBB;  
    border-radius:10px;  
    font-family:verdana;  
}
```

DIV stylée

Chapitre 2: Le langage CSS

Les positions

En CSS, il est possible de placer un élément n'importe où dans le document peut importe l'endroit où il est déclaré en HTML. Par exemple, si dans le code HTML nous avons déclaré deux balises `<div>` qu'on retiendra pas les noms `div1` et `div2` de telle sorte à ce que la `div1` soit déclarée avant la `div2`, alors il est logique que quand on affichera le résultat sur le navigateur, la `div1` sera placée avant la `div2`.

Avec les propriétés CSS qui gèrent les positions, et sans toucher au code HTML, on peut mettre la balise `div1` après la balise `div2`, voir on peut les superposer de telle sorte que l'une survole l'autre etc...

La propriété `position` permet de rendre l'ordre d'affichage des objets indépendant de l'ordre de leur déclaration en HTML. Elle peut avoir plusieurs valeurs:

Chapitre 2: Le langage CSS

Les positions

- **static:** Il s'agit de la valeur par défaut qui n'applique aucun positionnement particulier à l'objet et celui-ci est intégré normalement à l'emplacement décrit par le flux.
- **absolute:** Cette valeur permet de rendre la position de l'objet **absolue**. On peut donc l'afficher n'importe où, mais ce n'est pas tout. Le faire de déclarer le style **position:absolute** sur un objet, procure à celui ci la possibilité de survoler les autres objets. Il peut donc cacher un texte ou une image totalement ou partiellement selon l'endroit exacte où on l'a placé.
- **relative:** Cette valeur permet de décaler l'affichage d'un objet par rapport à son emplacement naturel (celui décrit par HTML).
- **fixed:** Cette valeur permet de fixer un élément sur la page même si celle ci est défilée à l'aide des barres de défilement.

Chapitre 2: Le langage CSS

Les positions

Pour pouvoir placer les objets là où l'on souhaite, il faut en plus de la propriété **position** utiliser d'autres propriétés CSS qui permettent de spécifier les coordonnées. De cette manière on peut déclarer avec précision l'emplacement de chaque objet auquel on a appliqué la propriété **position**.

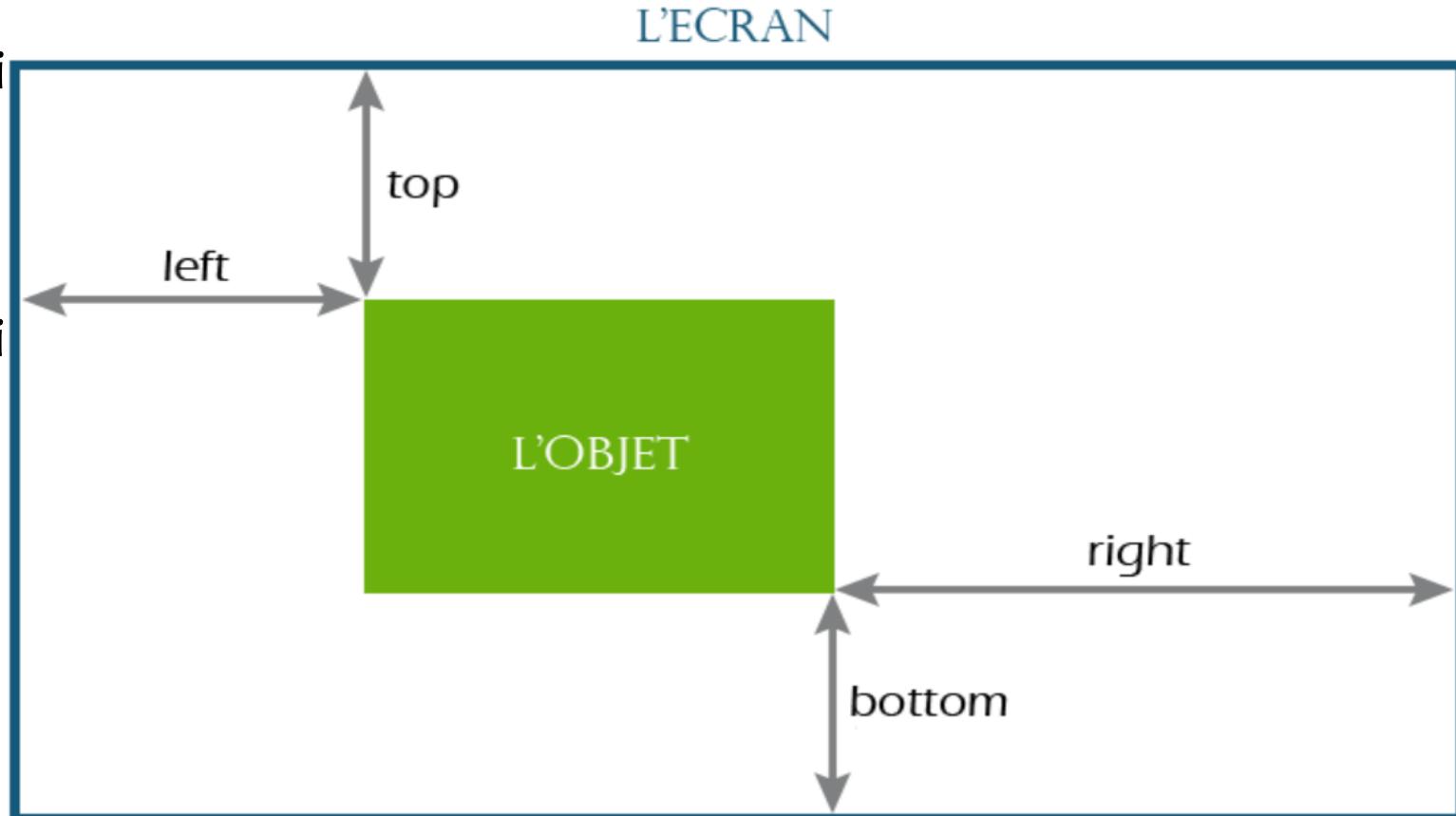
Le système de coordonnées

Quand on parle de coordonnées, on pense à un repère. Le repère n'est en fait rien d'autre que l'écran de l'utilisateur (ou la fenêtre du navigateur pour être plus précis). L'origine du repère n'est pas (tel qu'on l'a appris en Mathématiques) le coin bas à gauche, mais le coin haut à gauche du navigateur. Le sens positif horizontalement est vers la droite et verticalement vers le bas. Pour spécifier le **x** et le **y** on fait appel aux propriétés CSS **left**, **top**, **right** et **bottom**.

Chapitre 2: Le langage CSS

Le système de coordonnées

- **left:** exprime le nombre de pixels qui séparent l'extrémité gauche de l'objet du côté gauche du navigateur.
- **top:** exprime le nombre de pixels qui séparent l'extrémité supérieure de l'objet du haut du navigateur.
- **right:** exprime le nombre de pixels qui séparent l'extrémité droite de l'objet du côté droit du navigateur.
- **bottom:** exprime le nombre de pixels qui séparent l'extrémité inférieure de l'objet du bas du navigateur.



Chapitre 2: Le langage CSS

Le système de coordonnées

```
div{  
    width:300px;  
    height:200px;  
    background-color:orange;  
    position:absolute;  
    left:40px;  
    top:30px;  
}
```

Avec ce code, si du texte (ou autre contenu) était déjà à l'emplacement indiqué par les coordonnées, alors il serait caché par la balise <div>.

Notez que les positions et le système de coordonnées s'appliquent uniquement aux conteneurs (div, header, footer...). Si vous voulez les appliquer à une image par exemple, il faut donc la mettre dans un conteneur et appliquer à celui-ci le style souhaité.

Chapitre 2: Le langage CSS

Transformation des éléments

Vous vous souvenez probablement des éléments **inline** et **block**.

- Les éléments de type **inline** occupent juste assez d'espace pour qu'ils s'affichent correctement. Par exemple la balise `` est de type inline. Elle n'engendre pas de retour à la ligne automatique car elle se contente de l'espace nécessaire pour que ce qu'elle contient (texte ou autre objet) s'affiche convenablement. C'est le cas pour les balises ``, `<i>`, ``...
- Les balises de type **block** constituent un "bloc". Comme la balise `<div>` elle occupe toute la largeur de la page même si son contenu n'est pas aussi large. C'est ce qui explique le retour à la ligne automatique qui survient avant et après ce type de balises. Les balises `<header>`, `<footer>`, `<p>` et bien d'autres font aussi partie de cette famille.

Chapitre 2: Le langage CSS

Transformation des éléments

Balise DIV

Balise SPAN

Balise H1

Balise FONT

Vous voyez maintenant que les balises inline ont un arrière plan qui s'étale juste sur la largeur du contenu alors que pour les block l'arrière plan s'allonge jusqu'à la fin de la ligne.

Choisissons maintenant une balise de chaque famille et essayons de modifier leur largeur et leur hauteur.

Code HTML:

```
<div>Balise DIV</div>
<span>Balise SPAN</span>
<h1>Balise H1</h1>
<font>Balise FONT</font>
```

Code CSS:

```
div{
    background-color:#6666FF;
}
span{
    background-color:#FFFF00;
}
h1{
    background-color:#FF8800;
}
font{
    background-color:#AAAAAA;
}
```

Chapitre 2: Le langage CSS

Transformation des éléments

Balise DIV



Balise SPAN



Code HTML:

```
<div>Balise DIV</div>
<span>Balise SPAN</span>
```

Code CSS:

```
div{
    background-color:#6666FF;
    width:50%;
    height:100px;
}
span{
    background-color:#FFFF00;
    width:50%;
    height:100px;
}
```

Chapitre 2: Le langage CSS

Transformation des éléments

En effet, c'est tout à fait normal. Seules les balises de type **block** sont redimensionnables. Les balises **inline** ignorent les styles de redimensionnement à l'exception des images et les champs de formulaire.

Propriété display

Il a prévu des styles qui transforment les balises, c'est à dire qu'on peut rendre la balise `` de type **block** et la balise `<div>` de type **inline** et cela s'applique sur toutes les balises HTML. La propriété magique qui rend cet exploit possible s'appelle **display**.
de chacune des types **inline** et **block**.

Chapitre 2: Le langage CSS

Transformation des éléments

Propriété display

- La propriété **display** peut accueillir les valeurs suivantes:
inline: permet de transformer une balise HTML en type **inline**.
- **block**: permet de transformer une balise HTML en type **block**.
- **none**: permet de faire disparaître l'élément HTML. L'élément aura l'air comme s'il n'a pas été déclaré dans le code HTML.
- **inline-block**: cette valeur a été introduite en CSS3. Elle permet de réunir les principales caractéristiques de chacune des types **inline** et **block**. Si on applique la propriété **display:inline-block** à un élément, il n'engendre pas de retour automatique à la ligne (comme les **inline**), et peut être redimensionné (comme les **block**).

Chapitre 2: Le langage CSS

Transformation des éléments

Code HTML:

```
<div>Balise DIV</div>
<span>Balise SPAN</span>
```

Balise DIV
Balise SPAN

566FF;

```
}
```

```
span{
    background-color:#FFFF00;
    width:50%;
    height:100px;
    display:block;
}
```

Chapitre 2: Le langage CSS

Gestion de l'opacité

Propriétés opacity, filter:alpha(opacity)

Pour rendre un élément transparent, opaque ou translucide, on fait appel à la propriété **opacity**. Sa valeur est comprise entre 0 et 1 avec un pas de 0.01. 0 signifie complètement transparente, 1 veut dire complètement opaque et entre 0 et 1 ce sont les 99 niveaux de transparence possibles.

La propriété **opacity** est reconnue sur pratiquement tous les navigateurs connus sauf pour Internet Explorer dont la version est inférieure à 9 qui, lui, reconnaît une autre syntaxe, c'est **filter:alpha(opacity=x)** où x est la valeur de l'opacité comprise entre 0 et 100 (0 pour transparent et 100 pour opaque).

Chapitre 2: Le langage CSS

Gestion de l'opacité

Il existe une autre syntaxe mais qui est devenue obsolète, il s'agit de **-moz-opacity**. Elle était destinée aux moteurs de rendu **Gecko** comme celui embarqué sur Firefox. Elle acceptait les mêmes valeurs que la propriété **opacity**.

Mais, la question qui se pose est: "**Comment saurons nous quel navigateur utilise le client pour décider quelle propriété envoyer?**". La réponse est aussi simple que la solution; il suffit de déclarer toutes les propriétés ensemble (ou au moins les deux qui sont encore en service) et laisser le navigateur exécuter celle qu'il comprend.

```
img {  
    opacity:0.3;  
    filter:alpha(opacity=30);  
}
```



Chapitre 2: Le langage CSS

Gestion du débordement

Supposons que vous avez déclaré une balise `<div>` à laquelle vous avez fixé la largeur et la hauteur. Pourtant, vous y avez mis un texte qui dépasse. Comment la `<div>` se comporterait-elle? En fait, elle a tendance à afficher quand même tout le texte qui (selon le navigateur utilisé) aura l'air qui déborde de la `<div>`.

Code HTML:

```
<div>
CSS désigne Cascading Style Sheets (pour Feuilles de style en cascade). Il s'agit d'un langage de style dont la syntaxe est extrêmement simple mais son rendement est remarquable. En effet, le CSS s'intéresse à la mise en forme du contenu intégré avec du HTML.<br />
Donc, si vous voulez changer la couleur de votre arrière plan, la police de vos textes ou l'alignement et les marges de vos objets et bien réussir d'autres prouesses, CSS est là pour vous servir.
</div>
```

Chapitre 2: Le langage CSS

Gestion du débordement

Code CSS:

```
div{  
    height:60px;  
    border:solid 1px #AAAAAA;  
    padding:10px;  
}
```

CSS désigne Cascading Style Sheets (pour Feuilles de style en cascade). Il s'agit d'un langage de style dont la syntaxe est extrêmement simple mais son rendement est remarquable. En effet, le CSS s'intéresse à la mise en forme du contenu intégré avec du HTML.

Donc, si vous voulez changer la couleur de votre arrière plan, la police de vos textes ou l'alignement et les marges de vos objets et bien réussir d'autres prouesses, CSS est là pour vous servir.

Chapitre 2: Le langage CSS

Gestion du débordement

Propriété **overflow**

La propriété **overflow** permet de gérer le débordement sur un élément (de type conteneur). Elle a comme valeur:

- **visible**: C'est la valeur par défaut. Le contenu qui dépasse de l'élément est visible normalement.
- **hidden**: Le contenu qui dépasse sera masqué.
- **scroll**: Une barre de défilement apparaît qu'il y ait débordement ou non, mais elle ne sera opérationnelle que si le contenu dépasse de l'élément. Notez que l'imprimante peut imprimer tout le contenu, même celui qui dépasse.
- **auto**: Une barre de défilement apparaît s'il y a un débordement.

Chapitre 2: Le langage CSS

Gestion du débordement

Propriété overflow

Code CSS:

```
div{  
    height:60px;  
    border:solid 1px #AAAAAA;  
    padding:10px;  
    overflow:scroll;  
}
```

CSS désigne Cascading Style Sheets (pour Feuilles de style en cascade). Il s'agit d'un langage de style dont la syntaxe est extrêmement simple mais son rendement est remarquable. En effet, le CSS s'intéresse à la mise en forme du contenu intégré avec du HTML.

Donc, si vous voulez changer la couleur de votre arrière plan, la police de vos textes ou l'alignement et les marges de vos objets et bien réussir d'autres prouesses, CSS est là pour

Chapitre 2: Le langage CSS

Sélecteurs spéciaux

Groupement et identification d'éléments

Jusqu'ici, pour désigner un élément à styler en CSS on s'est servi du sélecteur de balise. Le navigateur applique donc le style à toutes les balises qui portent le même nom que le sélecteur.

```
div{  
    color:#888888;  
    text-decoration:underline;  
}
```

Dans ce cas, le navigateur parcourt toute la page en cherchant les balises `<div>` auxquelles il appliquera le style décrit par le sélecteur. Alors, le contenu de toutes les `<div>` sera gris et souligné. Mais, est ce qu'on souhaite vraiment que toutes les `<div>` de cette page soient ainsi? Sinon, comment distinguerons nous les balises auxquelles on appliquera ce style de celles qui n'en sont pas concernées?

Chapitre 2: Le langage CSS

Sélecteurs spéciaux

Groupement et identification d'éléments

Heureusement, il n'y a pas que le sélecteur de balise qui peut désigner l'élément. Il y a aussi le sélecteur de classe et l'identifiant d'élément.

L'attribut class: Groupement d'éléments

En HTML, il existe un attribut qui est très sollicité en CSS et il s'agit de l'attribut **class**.

Cet attribut peut avoir comme valeur un nom de votre choix et peut être appliqué à toutes les balises HTML déclarées entre **<body>** et **</body>**.

On peut appliquer le même nom de classe à plusieurs éléments similaires ou non. Cet ensemble d'éléments constitue un **groupement d'éléments**.

Chapitre 2: Le langage CSS

Sélecteurs spéciaux

Groupement et identification d'éléments

L'attribut class: Groupement d'éléments

```
.contenu{  
    /* style de la première DIV */  
}  
.meteo{  
    /* style de la deuxième DIV */  
}  
.rss{  
    /* style de la troisième DIV */  
}
```

Exemple du code HTML utilisant les classes:

```
<h1 class="titre"></h1>  
<nav class="menu"></nav>  
<div class="contenu"></div>  
<article>  
    <div class="meteo"></div>  
    <div class="rss"></div>  
</article>
```

Dans ce code, nous disposons de trois balises **<div>**. Si on utilisait le sélecteur de balise pour leur appliquer un style il se ressembleraient toutes. Donc nous allons faire appel au sélecteur de classes qu'on a déjà défini sur chacune des balises.

Chapitre 2: Le langage CSS

Sélecteurs spéciaux

Groupement et identification d'éléments

L'attribut class: Groupement d'éléments

Grâce au point qui vient juste avant le nom de la classe, le navigateur comprend qu'il doit appliquer le style à la balise qui renferme la classe dont le nom est défini en tant que sélecteur.

C'est ainsi qu'on peut appliquer des styles différents aux balises même si elles portent le même nom.

Mieux encore, on peut appliquer le même style aux balises différentes car ils ont la même classe.

On peut donner deux (ou plusieurs) noms de classe en même temps à la même balise comme ceci `class="classe1 classe2"`.

Il faut séparer les noms par un espace. De cette manière la balise aura les styles de la `classe1` et de la `classe2` simultanément.

Chapitre 2: Le langage CSS

Sélecteurs spéciaux

Groupement et identification d'éléments

Attribut id: identifiant d'élément

Si une classe peut être déclarée plusieurs fois dans le même document pour constituer un groupement d'éléments (qui accueilleront le même style), il existe un autre attribut CSS qui, quant à lui, ne doit figurer qu'une seule fois dans le même document. Il s'agit de l'attribut **id** connu par **identifiant d'élément**.

```
<img id="logo" />
<div id="slogan"></div>
```

```
#logo {
    /* Style de l'image affichant le logo */
}
#slogan{
    /* Style de la DIV qui contient le slogan */
}
```

Chapitre 2: Le langage CSS

Sélecteurs spéciaux

Groupement et identification d'éléments

Jusqu'ici nous avons vu le sélecteur de balise qui utilise le nom de la balise pour appliquer un style CSS, le sélecteur de classe qui se sert du nom de la classe préfixé par le point et l'identifiant d'élément qui commence par un dièse. Mais ce n'est pas tout. En CSS on peut utiliser d'autres méthodes pour cibler les éléments.

Cibler tous les éléments

Pour cibler tous les éléments d'un document HTML, on utilise le symbole astérisque (*). De cette manière, la balise **<body>** et toutes les balises qu'elle contient sont concernées par le style.

Chapitre 2: Le langage CSS

Sélecteurs spéciaux

Cibler tous les éléments

Pour cibler tous les éléments d'un document HTML, on utilise le symbole astérisque (*). De cette manière, la balise **<body>** et toutes les balises qu'elle contient sont concernées par le style.

Exemple:

```
* {  
    color:#888888;  
    font-size:10pt;  
    font-family:verdana, arial, sans-serif;  
}
```

Chapitre 2: Le langage CSS

Sélecteurs spéciaux

Cibler tous les éléments ayant un attribut particulier

Pour cibler toutes les balises sur lesquelles on a déclaré un attribut particulier (quelque soit sa valeur) on fait appel au sélecteur de balise suivi de crochets qui renferment le nom de l'attribut souhaité.

Dans ce cas, toutes les balises `` qui ont un attribut **title** seront translucides.

```
img[title] {  
    opacity:0.7;  
    filter:alpha(opacity=70);  
}
```

Chapitre 2: Le langage CSS

Sélecteurs spéciaux

Cibler tous les éléments ayant un attribut avec une valeur particulière

Pour cibler toutes les balises sur lesquelles on a déclaré un attribut qui a une valeur bien définie on fait appel au sélecteur de balise suivi de crochets qui renferment le nom de l'attribut souhaité auquel on a attribué la valeur voulue (comme si on le déclarait en HTML).

Exemple:

```
input[type="text"] {  
    border:solid 1px blue;  
}
```

Dans ce cas, toutes les zones de texte auront une bordure normale colorée en bleu.

Chapitre 2: Le langage CSS

Sélecteurs spéciaux

Cibler un élément contenu dans un autre

Pour cibler un élément contenu dans un autre, on déclare le sélecteur du premier élément suivi d'un espace suivi du sélecteur du deuxième élément. Le sélecteur peut être n'importe lequel parmi ceux vu précédemment (balise, classe, id...).

```
div img{  
    margin:10px;  
}
```

Dans ce cas, toutes les images qui sont déclarées dans une balise `<div>` auront une marge de 10px de tous les cotés.

Chapitre 2: Le langage CSS

Sélecteurs spéciaux

Cibler un élément enfant direct d'un parent

Si on veut désigner un élément enfant appartenant directement à un parent spécifié alors on utilise le symbole >.

```
div>img{  
    margin:10px;  
}
```

Dans ce cas, seules les images qui sont intégrées directement dans une balise `<div>` seront stylées.

Chapitre 2: Le langage CSS

Sélecteurs spéciaux

Cibler plusieurs éléments à la fois

Pour désigner plusieurs éléments à la fois, il suffit de les déclarer tous séparés par une virgule.

```
header, footer, nav, section{  
    margin:0;  
    border:none;  
}
```

Dans ce cas, les balises `<header>`, `<footer>`, `<nav>` et `<section>` n'auront aucune marge ni aucune bordure.

Chapitre 2: Le langage CSS

Sélecteurs spéciaux

Cibler un élément précédé immédiatement par un autre

Afin de spécifier un élément HTML qui suit directement un élément particulier on utiliser le caractère +.

Exemple:

```
nav+h1{  
    margin:auto;  
}
```

Dans cet exemple le style sera appliqué à la balise **<h1>** qui vient directement après la balise **<nav>**. C'est à dire qu'on a déjà fermé la balise **<nav>** puis on a ouvert la balise **<h1>** juste après.

Chapitre 2: Le langage CSS

Sélecteurs spéciaux

Ordre d'application des styles

Souvent, on se confronte à un élément dont le style est décrit plusieurs fois. La règle est simple, les propriétés qui ne sont pas répétés d'un style à un autre sont combinées et appliquées toutes à l'élément, par contre, les règles qui se répètent s'annulent les unes par les autres, celles qui sont définies en dernier seront appliquées.

Chapitre 2: Le langage CSS

Pseudo-classes et pseudo-éléments

Pseudo-classes

Une pseudo-classe est un mot clé préfixé par deux points (:) qui s'ajoute à un sélecteur CSS pour appliquer un style à un élément dans un cas particulier. L'utilisation des pseudo-classes est très populaire sur les liens hypertextes qui changent de décor selon si ils sont nouvellement affichés, déjà visités ou survolés... Or, leur utilisation s'étend aux autres objets HTML tel que les images, les conteneurs, les listes...

Voici la liste des pseudo-classes les plus utilisées en CSS:

Pseudo-classe :link

La pseudo-classe **:link** désigne un lien hypertexte dont la page cible (celle spécifiée dans son attribut **href**) n'a pas encore été visitée. Elle peut aider les internautes à distinguer les liens qu'ils n'ont pas encore exploré.

Chapitre 2: Le langage CSS

Pseudo-classes et pseudo-éléments

Pseudo-classes

Pseudo-classe :visited

A l'inverse de **link**, la pseudo-classe **:visited** désigne un lien hypertexte dont la page cible a déjà été visitée. C'est un genre d'historique en quelque sorte.

Pseudo-classe :active

La pseudo-classe **:active** désigne un lien hypertexte sélectionné. Il s'agit de l'instant où le visiteur clique sur le lien, si celui-ci reste affiché sur la page (cas où la page est figée après le clic, ou le lien est ouvert dans une nouvelle fenêtre) on peut voir l'effet de cette pseudo-classe.

Pseudo-classe :hover

La pseudo-classe **:hover** désigne un objet survolé par le curseur de la souris, il peut être un hyperlien ou n'importe quel autre objet.

Chapitre 2: Le langage CSS

Pseudo-classes et pseudo-éléments

Pseudo-classes

Hyperlien

Hyperlien

Dans cet exemple, nous avons spécifié un style qui s'applique à tous les liens quelque soit leur état (sélecteur **a**). Après nous avons explicité le style de l'état survolé (**a:hover**). Il faut savoir que tous les styles déjà déclarés dans le premier sélecteur sont hérités par le sélecteur de l'état survolé, et ceux qui ont été redéfinis seront écrasés et remplacés par leur nouvelle valeur (cas des propriétés **color** et **text-decoration**).

Code HTML:

```
<a href="#">Hyperlien</a>
```

Code CSS:

```
a{  
    font-weight:bold;  
    color:#EE6600;  
    text-decoration:none;  
}  
a:hover{  
    color:#000088;  
    text-decoration:underline;  
}
```

Chapitre 2: Le langage CSS

Pseudo-classes et pseudo-éléments

Pseudo-classes

Si vous survolez l'image, vous constatez que le curseur de la sourie change de forme pour ressembler à celle qu'on a l'habitude de voir sur des liens hypertextes. C'est grâce à la propriété **cursor** à laquelle on a attribué la valeur **pointer**. Il existe d'autres valeurs comme **move**, **crosshair**, **default**...

Code HTML:

```

```

Code CSS:

```
#logo{  
    opacity:1;  
    filter:alpha(opacity=100);  
    cursor:pointer;  
}  
#logo:hover{  
    opacity:0.5;  
    filter:alpha(opacity=50);  
}
```

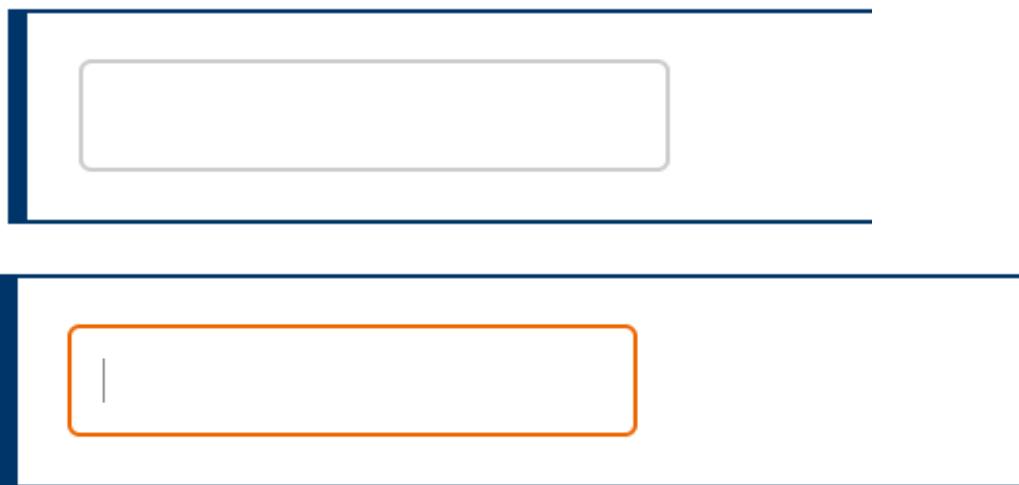
Chapitre 2: Le langage CSS

Pseudo-classes et pseudo-éléments

Pseudo-classes

Pseudo-classe :focus

La pseudo-classe **:focus** désigne un élément activé ou sélectionné suite à un clic ou une tabulation. C'est souvent utilisé sur les champs de formulaire.



Code HTML:

```
<input type="text" name="login" />
```

Code CSS:

```
input[type="text"] {  
    border:solid 1px #CCCCCC;  
    color:#888888;  
    padding:10px;  
    border-radius:4px;  
}  
input[type="text"]:focus {  
    border:solid 1px #EE6600;  
    color:#888888;  
    outline:none;  
}
```

Chapitre 2: Le langage CSS

Pseudo-classes et pseudo-éléments

Pseudo-classes

Pseudo-classe :focus

Là encore vous avez remarqué la propriété **outline** à laquelle on a donné la valeur **none**.

Elle désigne un cadre qui ressemble à la bordure mais qui entoure l'objet (sa bordure comprise). Certains navigateur (comme Google Chrome) appliquent automatiquement ce cadre aux champs de formulaires sélectionnés, ce qui peut altérer un peu le design souhaité. La valeur **none** élimine tout cadre éventuel.

*La propriété **outline** a les mêmes valeurs que la propriété **border**.*

Chapitre 2: Le langage CSS

Pseudo-classes et pseudo-éléments

Pseudo-classes

Pseudo-classe :first-child

La pseudo-classe **first-child** désigne le premier élément enfant. Supposons que nous avons déclaré le sélecteur suivant **span:first-child**. Le navigateur appliquera le style associé à toutes les balises **** qui figurent en tant que premier élément enfant de n'importe quelle balise. Si le code HTML était comme ceci:

```
<div>
  <span>Bonjour</span>
  à
  <span>tous</span>
</div>
```

Alors seule la première balise **** s'attribuerait le style décrit dans le sélecteur.

Chapitre 2: Le langage CSS

Pseudo-classes et pseudo-éléments

Pseudo-classes

Pseudo-classe :nth-child(n)

Si la pseudo-classe **first-child** permet d'accéder au premier élément enfant d'un parent quelconque, la pseudo-classe **nth-child(n)** quant à elle, permet d'accéder à un élément enfant de n'importe quel rang et pas qu'au premier. Le paramètre **n** mis entre les parenthèses désigne le rang de l'élément à styler. Il s'agit d'un indice numérique qui commence de 1 (1 étant le premier élément, 2 le deuxième et ainsi de suite...).

Pour l'exemple HTML précédent, le sélecteur **div>span:nth-child(2)** fait référence à la deuxième balise ****.

Chapitre 2: Le langage CSS

Pseudo-classes et pseudo-éléments

Pseudo-éléments

Tout comme les pseudo-classes, les **pseudo-éléments** sont prefixés par deux points (:) et ajoutés au sélecteur. Si les pseudo-classe décrivent un état d'un élément (comme un lien hypertexte survolé), les **pseudo-éléments** eux accèdent à certaines parties de l'élément pour les styler.

Les pseudo-éléments les plus fréquents:

Pseudo-élément ::first-letter

Le pseudo-élément **::first-letter** désigne la première lettre de l'élément auquel il est associé. Il permet de donner un style particulier à la première lettre d'un élément.

Chapitre 2: Le langage CSS

Pseudo-classes et pseudo-éléments

Pseudo-éléments

Pseudo-élément ::first-letter

Le pseudo-élément **::first-letter** désigne la première lettre de l'élément auquel il est associé. Il permet de donner un style particulier à la première lettre d'un élément.

Ce qui donne:

La première lettre de ce texte est différente!

Exemple:

Code HTML:

```
<div>
    La première lettre de ce texte est différente!
</div>
```

Code CSS:

```
div::first-letter{
    font-size:24pt;
    color:orange;
}
```

Chapitre 2: Le langage CSS

Pseudo-classes et pseudo-éléments

Pseudo-éléments

Tout comme les pseudo-classes, les pseudo-éléments sont préfixés par deux points (:) et ajoutés au sélecteur. Si les pseudo-classes décrivent un état d'un élément (comme un lien hypertexte survolé), les pseudo-éléments eux accèdent à certaines parties de l'élément pour les styler.

Pseudo-élément ::first-line

Le pseudo-élément **::first-line** désigne la première ligne de l'élément. On peut lui donner un style particulier tout comme pour **first-letter**. Le contenu de la première ligne peut changer selon la largeur de la fenêtre du navigateur.

Code HTML:

```
<p class="paragraphe">  
Tout comme les pseudo-classes, les pseudo-éléments sont préfixés par deux points (:) et ajoutés au sélecteur. Si les pseudo-classes décrivent un état d'un élément (comme un lien hypertexte survolé), les pseudo-éléments eux accèdent à certaines parties de l'élément pour les styler.  
</p>
```

Code CSS:

```
.paragraphe::first-line{  
    color:blue;  
}
```

Qu'est ce que "color" en CSS ?

- 1.Un sélecteur
- 2.Une propriété
- 3.Une déclaration

Comment sélectionne t-on le div class="div1" en CSS ?

- 1.Avec le sélecteur "div1"
- 2.Avec le sélecteur "#div1"
- 3.Avec le sélecteur ".div1"

Laquelle de ces propositions est exacte ?

- 1.On doit donner une valeur unique à un attribut id mais pas à des attributs class
- 2.On doit donner une valeur unique à un attribut class mais pas à des attributs id
- 3.On doit donner une valeur unique à chaque attribut class et id dans une page

Un élément de type block...

- 1.Occupe toute la largeur disponible dans la page
- 2.N'occupe que la largeur nécessaire à son contenu
- 3.Occupe toute la largeur disponible dans son élément parent

Quelle est la différence majeure entre les éléments div et span ?

- 1.L'élément span est de type block, l'élément div est de type inline
- 2.L'élément div est de type block, l'élément span est de type inline
- 3.L'élément div sert de conteneur, au contraire de l'élément span

Que se passe t-il si j'applique "display:none" à un élément ?

- 1.L'élément est effacé du code de ma page HTML
- 2.Le type de display de l'élément n'est pas modifié
- 3.L'élément ne sera pas affiché dans le navigateur

Le sélecteur CSS étoile (*) sert à...

- 1.Appliquer des bordures aux éléments
- 2.Sélectionner tous les éléments d'une page HTML
- 3.Sélectionner un élément possédant un attribut en particulier

Le sélecteur div + p sert à sélectionner...

- 1.Le premier élément p enfant d'un div
- 2.Tous les éléments p enfants dans un div
- 3.Tous les éléments p suivants un div (et de même niveau)

A quoi reconnaît-on les pseudo classes en CSS ?

- 1.Elles commencent toutes par ":"
- 2.Elles commencent toutes par "::"
- 3.Elles possèdent toutes le mot "class" dans leur nom

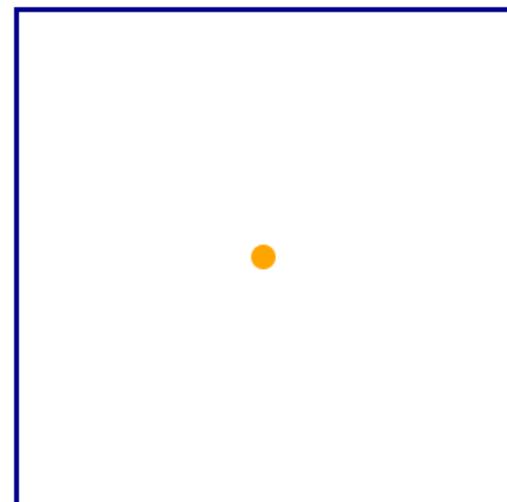
Chapitre 2: Le langage CSS

Les transformations CSS

Transformation de l'origine

A partir de la version 3 de CSS il est devenu possible de transformer les objets intégrés en 2D (et même en 3D) sans vraiment modifier leurs propriétés (largeur, hauteur...). Cette technique a permis aux designers de réussir des effets avec un minimum de code.

Les transformations d'objets sont appliquées par rapport à son point d'origine. Ce point est par défaut situé au centre l'objet.



Chapitre 2: Le langage CSS

Les transformations CSS

Fonctions de transformation qui s'appliquent à la propriété transform

Pour appliquer une transformation en CSS on utilise la syntaxe suivante:

transform: fonction1() fonction2() fonction3()...

Les fonctions sont séparées par un espace. Cependant on peut toujours utiliser une seule fonction si on veut.

Les fonctions contiennent des paramètres qui permettent de personnaliser la transformation souhaitée. Ces paramètres sont renseignés entre les parenthèses de la fonction.

Chapitre 2: Le langage CSS

Les transformations CSS

Fonctions de transformation qui s'appliquent à la propriété transform

Fonction translate()

La fonction **translate()** permet la translation de l'objet HTML d'un emplacement initial (où il est normalement défini) à un autre emplacement.

Les coordonnées du nouvel emplacement sont renseignées entre les parenthèses de **translate()** et ils sont exprimés en pixel.

L'origine des coordonnées, comme on l'a vu précédemment, est le point haut à gauche.

Le premier paramètre de la fonction **translate()** représente la translation horizontale, et le deuxième la translation verticale.

Notez que les sens positifs sont **vers la gauche** et **vers le bas**. Si vous renseignez des valeurs négatives à la fonctions alors la translation sera appliquée dans le sens opposé.

Chapitre 2: Le langage CSS

Les transformations CSS

Fonctions de transformation qui s'appliquent à la propriété transform

Il existe aussi les fonctions **translateX()** qui réalise une translation horizontale et **translateY()** qui réalise une translation verticale. Dans ce cas, nous spécifions une seule valeur dans les parenthèses.

```
div{  
    width:100px;  
    height:100px;  
    background-color:orange;  
    transform:translate(100px, 30px);  
}
```



Chapitre 2: Le langage CSS

Les transformations CSS

Fonctions de transformation qui s'appliquent à la propriété transform

Fonction scale()

La fonction **scale()** permet de redimensionner l'objet HTML (avec son contenu). Les parenthèses de la fonction peuvent accueillir une seule valeur ou deux valeurs séparées par une virgule. Ces valeurs désignent le facteur d'agrandissement. Ils n'ont pas d'unité et peuvent contenir de virgule. S'ils sont supérieur à 1 alors l'objet est agrandit, sinon alors il est rétréci.

Si une seule valeur est renseignée dans les parenthèses, alors le même facteur d'agrandissement agit sur la largeur et la hauteur de l'objet.

Chapitre 2: Le langage CSS

Les transformations CSS

Fonctions de transformation qui s'appliquent à la propriété transform

Fonction scale()

```
div{  
    width:100px;  
    height:100px;  
    background-color:orange;  
    transform-origin:0 0;  
    transform:scale(1.5);  
}
```



L'origine de l'objet pour que la transformation s'effectue par rapport au point haut à gauche.

Chapitre 2: Le langage CSS

Les transformations CSS

Fonctions de transformation qui s'appliquent à la propriété transform

```
div{  
    width:100px;  
    height:100px;  
    background-color:orange;  
    transform-origin:0 0;  
    transform:scale(2,0.8);  
}
```



Il existe également les fonctions **scaleX()** et **scaleY()** qui appliquent respectivement un facteur d'agrandissement uniquement sur la largeur ou la hauteur.

Chapitre 2: Le langage CSS

Les transformations CSS

Fonctions de transformation qui s'appliquent à la propriété transform

Fonctions skewX() et skewY()

Les fonctions **skewX()** et **skewY()** permettent d'incliner un objet respectivement horizontalement et verticalement par rapport à son origine. La valeur renseignée dans les parenthèses désigne l'angle d'inclinaison et elle est exprimée en degré (**deg**).

```
div{  
    width:100px;  
    height:100px;  
    background-color:orange;  
    transform-origin:0 0;  
    transform:skewX(30deg);  
}
```



Chapitre 2: Le langage CSS

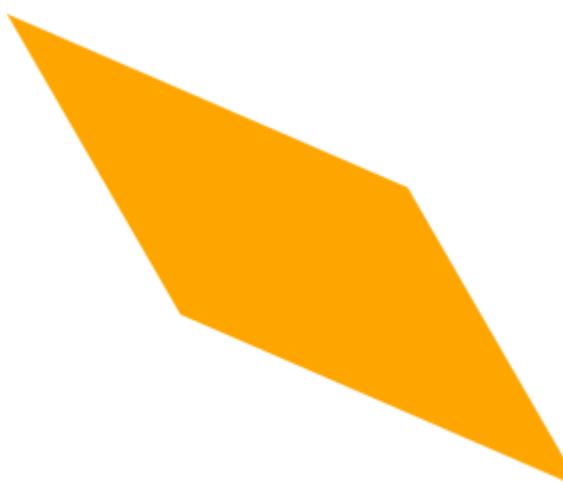
Les transformations CSS

Fonctions de transformation qui s'appliquent à la propriété transform

```
div{  
    width:100px;  
    height:100px;  
    background-color:orange;  
    transform:skewY(30deg);  
}
```



```
div{  
    width:100px;  
    height:100px;  
    background-color:orange;  
    transform:skewX(30deg) skewY(30deg);  
}
```



Chapitre 2: Le langage CSS

Les transformations CSS

Fonctions de transformation qui s'appliquent à la propriété transform

Fonction **rotate()**

Comme son nom l'indique, la fonction **rotate()** applique une rotation à l'objet autour de l'axe traversant son origine. Les parenthèses contiennent l'angle de la rotation exprimée en degré.

Selon l'axe de rotation on distingue trois variantes de la fonction **rotate()**:

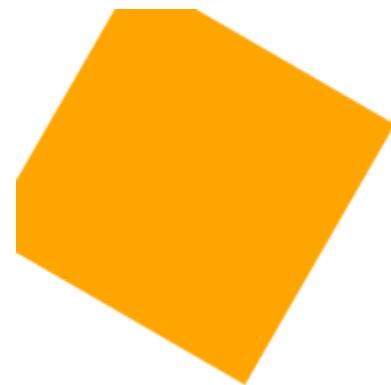
- **rotate()** ou **rotateZ()**: la rotation est appliquée par rapport à l'axe virtuel perpendiculaire à l'écran (le fameux axe des Z de **z-index**) qui traverse le point d'origine de l'objet. La rotation s'applique donc sur le même plan que l'écran.
- **rotateX()**: la rotation est appliquée par rapport à l'axe horizontale traversant l'origine de l'objet (l'axe des X). La rotation aura donc un effet 3D
- **rotateY()**: la rotation est appliquée par rapport à l'axe vertical traversant l'origine de l'objet (l'axe des Y). La rotation aura aussi un effet 3D

Chapitre 2: Le langage CSS

Les transformations CSS

Fonctions de transformation qui s'appliquent à la propriété transform

```
div{  
    width:100px;  
    height:100px;  
    background-color:orange;  
    transform:rotate(30deg);  
}
```



Chapitre 2: Le langage CSS

Les transformations CSS

Fonctions de transformation qui s'appliquent à la propriété transform

Fonction perspective()

La fonction **perspective()** permet de prendre en compte les changements des dimensions en fonction de la profondeur (ou perspectives). Par exemple, si la rotation est appliquée par rapport à l'axe des X, le côté le plus loin (celui qui, soi-disant, entre dans l'écran) sera plus petit que celui qui en sort.

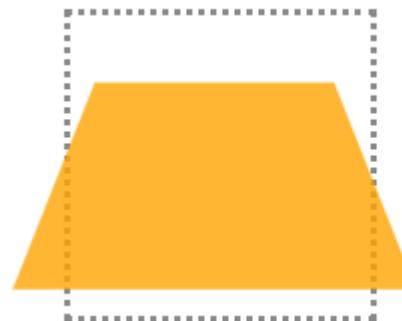
La fonction perspective() reçoit une valeur exprimée en pixel qui désigne la distance (virtuelle) qui sépare l'observateur de l'axe de rotation (qui traverse l'origine de l'objet). Plus cette distance est petite, plus l'effet de perspective est accentué et inversement.

Chapitre 2: Le langage CSS

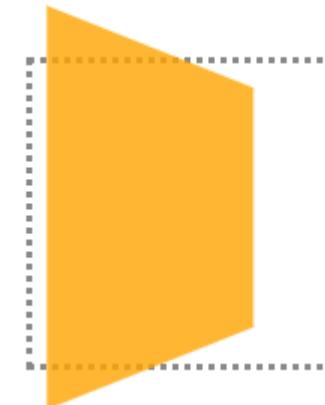
Les transformations CSS

Fonctions de transformation qui s'appliquent à la propriété transform

```
div{  
    width:100px;  
    height:100px;  
    background-color:orange;  
    transform:perspective(150px) rotateX(50deg);  
}
```



```
div{  
    width:100px;  
    height:100px;  
    background-color:orange;  
    transform:perspective(150px) rotateY(50deg);  
}
```



Chapitre 2: Le langage CSS

Les transitions CSS

Les transitions CSS permettent le passage d'un objet d'un état à un autre d'une manière progressive créant ainsi une animation. Pour être plus précis, ce type d'animation s'appelle **Interpolation de mouvement**.

En CSS, on a souvent recours aux transitions avec les pseudo-classes. Par exemple, après avoir survolé un objet, celui ci se met à bouger pour correspondre aux styles définis par **:hover**.

Pour mieux comprendre de quoi il s'agit nous allons créer un objet et nous allons définir son état survolé (grâce à **:hover**) deux fois, une sans transition et l'autre avec transitions.

Chapitre 2: Le langage CSS

Les transitions CSS

Propriété **transition-property**

Supposons que nous avons créé un objet dont on a défini l'arrière plan (propriété background) et la bordure (propriété border), puis nous avons créé des styles appliquées à une pseudo-classe de cet objet (:hover par exemple) qui changeront l'arrière plan et la bordure déjà définis. Si on applique une transition entre l'état initial et l'état survolé, on imagine que l'arrière plan et la bordure se mettront à s'animer pour correspondre aux styles définis à :hover. Cependant, CSS permet de désigner quels sont les propriétés qui devront subir la transition et quels sont celles qui ne s'animeront pas. Il s'agit de la propriété **transition-property**.

La propriété **transition-property** accepte comme valeur le nom (ou la famille) de styles sur lesquelles la transition sera appliquée.

Chapitre 2: Le langage CSS

Les transitions CSS

Propriété transition-duration

Comme son nom l'indique la propriété **transition-duration** permet de spécifier la durée de la transition (la durée que mettra l'animation pour aller de l'état 1 à l'état 2). Elle est exprimée en secondes.

Propriété transition-timing-function

- La fonction **transition-timing-function** permet de définir la cadence avec laquelle la transition sera faite. Elle permet de changer la vitesse de la transition au long de sa durée (on parle de la **courbe d'accélération**). Les valeurs que cette propriété peut accueillir sont:

Chapitre 2: Le langage CSS

Les transitions CSS

- **ease**: le mouvement est rapide au début puis ralentit petit à petit jusqu'à l'arrêt complet.
- **linear**: le mouvement garde la même vitesse du début à la fin de la transition.
- **ease-in**: le mouvement commence lentement puis gagne en vitesse avec le temps.
- **ease-out**: le mouvement est rapide au début puis ralentit à la fin (le ralenti avec **ease-out** est moins senti qu'avec **ease**).
- **ease-in-out**: le mouvement commence lentement et fini lentement, mais il est plus rapide au milieu de la transition.

Propriété **transition-delay**

La propriété **transition-delay** permet de retarder le début de la transition lorsque l'événement qui la déclenche est détecté.

Chapitre 2: Le langage CSS

Les transitions CSS

```
.cours_transitions {  
    width:200px;  
    height:200px;  
    background-color:#FF8800;  
    margin:30px;  
    cursor:pointer;  
    -webkit-transition:all 0.5s ease;  
    -o-transition:all 0.5s ease;  
    -moz-transition:all 0.5s ease;  
    -ms-transition:all 0.5s ease;  
    transition:all 0.5s ease;  
}  
.cours_transitions:hover{  
    background-color:#003569; |  
    border-radius:100px;  
    -webkit-transition:all 0.5s ease;  
    -o-transition:all 0.5s ease;  
    -moz-transition:all 0.5s ease;  
    -ms-transition:all 0.5s ease;  
    transition:all 0.5s ease;  
}
```

Chapitre 2: Le langage CSS

Les transitions CSS

menu accordéon

```
<html>
  <head>
    <meta charset="UTF-8" />
    <link rel="stylesheet" type="text/css" href="style3.css" />
  </head>
  <body>
    <div>
      <div class="exercice_accordeon">Accueil</div>
      <div class="exercice_accordeon">Qui sommes-nous?</div>
      <div class="exercice_accordeon">Nos produits</div>
      <div class="exercice_accordeon">SAV</div>
      <div class="exercice_accordeon">Contact</div>
    </div>
  </body>
</html>
```

Chapitre 2: Le langage CSS

Les transitions CSS

menu accordéon

```
.exercice_accordeon{
    width:200px;
    background-color:#cccccc;
    padding:10px;
    margin-bottom:1px;
    font:10pt verdana;
    color:#FFFFFF;
    border:solid 1px #cccccc;
    cursor:pointer;
    -webkit-transition:all 0.5s ease;
    -moz-transition:all 0.5s ease;
    -o-transition:all 0.5s ease;
    -ms-transition:all 0.5s ease;
    transition:all 0.5s ease;
}

.exercice_accordeon:hover{
    background-color:#FFFFFF;
    color:#003569;
    border:solid 1px #003569;
    border-bottom:solid 60px #003569;
    -webkit-transition:all 0.5s ease;
    -moz-transition:all 0.5s ease;
    -o-transition:all 0.5s ease;
    -ms-transition:all 0.5s ease;
    transition:all 0.5s ease;
}
```

Chapitre 3: Le langage Javascript

Langage Javascript

1. Javascript C'est quoi?
2. Comment intégrer du code Javascript?
3. Les bases du Javascript
4. Structures de contrôle
5. Les fonctions
6. Les évènements
7. Objets, méthodes et attributs
8. Objet Array: Les tableaux
9. Objet String: Chaines de caractères
10. Objet Math
11. Objet Date
12. Objet RegExp: les expressions régulières
13. Objet Window
14. Objet Screen
15. Objet Document
16. Objet Event
17. Gestion des exceptions

Chapitre 3: Le langage Javascript

Qu'est ce que Javascript?

Javascript est un langage de script côté client, c'est à dire qu'il est exécuté sur le navigateur. Il a été créé en 1995 par [Netscape Communication Corporation](#). A l'époque il s'appelait **LiveScript**. Fin 1995, et suite à une association entre Netscape et SUN (créateur du fameux langage Java), LiveScript s'est fait rebaptiser Javascript.

Puisque Javascript est un langage côté client alors il a un principal avantage et un principal inconvénient:

- **Avantage:** Le code source Javascript est chargé avec la page (en même temps que HTLM et CSS). A la fin du chargement, tous les codes sont présents chez le client. Par conséquent, l'exécution de Javascript est extrêmement rapide puisque tout est disponible sur place. Inversement aux langages côté serveur comme PHP, qui nécessitent à chaque fois l'interrogation du serveur, ce qui rend l'exécution plus lente.

- **Inconvénient:** La même raison qui a procuré à Javascript son principal avantage, cause cette fois-ci son ultime inconvénient. En effet, le fait que les codes soient disponibles sur le navigateur rend ceux ci visibles par tous. Le code Javascript est par conséquent non confidentiel.

Chapitre 3: Le langage Javascript

Comment intégrer du code Javascript?

Pour écrire du code Javascript il faut le séparer des codes HTML et CSS en le plaçant entre les balises `<script>` et `</script>`. Pour faire simple, imaginons que je veux afficher le message "Bonjour" dans une boite de dialogue à l'aide de Javascript. Le code de notre page ressemblerait à ceci:

```
<html>
  <head>
    <meta charset="UTF-8" />
  </head>
  <body>
    <script>
      alert("Bonjour");
    </script>
  </body>
</html>
```

Chapitre 3: Le langage Javascript

Comment intégrer du code Javascript?

Les deux déclarations sont explicites: "Le langage utilisé dans les délimiteurs est Javascript". La deuxième déclaration est par contre plus appréciée vu qu'elle fournit deux informations, à savoir le type et le MIME. Le type est **text** car Javascript c'est du texte, et le MIME précise qu'il ne s'agit pas de n'importe quel texte mais du **Javascript**.

```
<script language="javascript"></script>
```

```
<script type="text/javascript"></script>
```

Chapitre 3: Le langage Javascript

Comment intégrer du code Javascript?

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

Cependant, comme vous allez le découvrir dans ce cours, les codes Javascript ont tendance à être plus ou moins longs (en fonction du traitement souhaité). Notre document HTML sera donc plus ou moins volumineux si on y met directement nos scripts. La solution consiste alors à déclarer le Javascript dans un fichier à part (comme on l'a fait pour le CSS).

Imaginons que nous voulons déclarer nos scripts dans un fichier nommé **script.js** (vous avez remarqué l'extension? **js** signifie Javascript). Pour simplifier, on va supposer que le fichier **script.js** est placé dans le même dossier que notre fichier HTML.

Chapitre 3: Le langage Javascript

Comment intégrer du code Javascript?

```
<html>
  <head>
    <meta charset="UTF-8" />
  </head>
  <body>
    <script type="text/javascript" src="script.js"></script>
  </body>
</html>
```

```
        alert ("Bonjour");
```

Chapitre 3: Le langage Javascript

Comment intégrer du code Javascript?

Où placer nos scripts?

Il n'y a pas de restriction sur l'endroit où on peut placer nos scripts.

Nous pouvons les placer n'importe où (avant `<html>`, dans `<head>`, dans `<body>` ou après `</html>`).

Cependant, les développeurs ont pour coutume de placer les scripts Javascript dans la balise `<head>` (Nous parlerons de la raison de ce choix quand ça sera temps).

Le code HTML précédent devrait donc ressembler à cela:

```
<html>
  <head>
    <meta charset="UTF-8" />
    <script type="text/javascript" src="script.js"></script>
  </head>
  <body>

  </body>
</html>
```

Les commentaires

- Un commentaire est un texte qui est ignoré lors de l'exécution du programme. Il est visible dans le code source mais n'est pas pris en compte par l'interpréteur Javascript.

Comme en langage C, il existe deux façons d'insérer un commentaire:

- **Commentaire de fin de ligne:** Il est inséré grâce au double slash (//). Tout le texte qui vient après est considéré comme commentaire jusqu'à la fin de la ligne.
- **Commentaire sur plusieurs lignes:** Il est inséré grâce à la séquence /* */. Tout le texte entre /* et */ est considéré comme commentaire même s'il est déclaré sur plusieurs lignes.

Exemple:

```
// Ce commentaire tien jusqu'à la fin de la ligne.  
/* Celui là, par contre,  
tient jusqu'à  
ce symbole: */
```

Chapitre 3: Le langage Javascript

Les bases du Javascript?

Les variables

Dans un langage de programmation, une variable sert à stocker une valeur (ou plusieurs) qui peut changer au cours de l'exécution du programme. En Javascript aussi, la variable sert à la même finalité.

La déclaration des variables en Javascript ne demande pas de préciser le type. En effet, le typage est géré dynamiquement, contrairement à la plupart des langages de programmation qui exigent de préciser le type de données qu'accueillera la variable au cours du programme (comme c'est le cas en langage C par exemple).

Exemple:

Les deux valeurs affectées aux variables sont de type **chaîne de caractères** (ou **string**).

```
a="Bonjour";  
var b="Salut";
```

Les variables

Nom des variables

Pour identifier une variable on lui donne un nom unique (qui ne doit pas être attribué à une autre variable dans le même programme). Le nom de la variable peut être composé de lettres (minuscules, majuscules ou les deux), de chiffres (de 0 à 9) et les caractères _ et &. En plus, il doit impérativement commencer par une lettre ou le caractère _.

Exemple:

```
maVariable=10; // Juste  
_maVariable=10; // Juste  
lmaVariable=10; // Faux  
ma Variable=10; // Faux
```

Les variables

Portée des variables

- On appelle portée d'une variable, l'étendu de celle ci, c'est à dire jusqu'à quel niveau elle sera reconnue dans un programme. Selon la portée, il existe deux types de variables:
 - **Variable globale:** Cette variable est accessible de partout dans le programme, à l'intérieur des fonctions comme à l'extérieur.
 - **Variable locale:** On parle plus précisément d'une variable locale à une fonction. C'est une variable qui est accessible de l'intérieur de la fonction dans laquelle elle est déclarée, mais pas de l'extérieur.
- si la variable est déclarée **sans le mot clé var**, alors elle est **automatiquement globale** qu'elle soit déclarée à l'intérieur d'une fonction ou à l'extérieur.
- Par contre si elle est déclarée avec le mot clé **var** alors deux cas sont envisageables:
 - Si la variable est déclarée dans une fonction, alors elle est locale à cette fonction.
 - Si la variable est déclarée en dehors de toute fonction, alors elle est globale à tout le programme.

Les variables

Changement de type d'une variable (CAST)

Le fait d'affecter une valeur à une variable, renseigne automatiquement au moteur Javascript (l'interpréteur) son type. Mais pour avoir une idée sur les types disponibles en Javascript voilà un résumé:

- **Les nombres:** Ce sont les nombres avec ou sans virgule, positifs ou négatifs.
- **Les chaînes de caractères:** (ou **string**), ce sont des suites de caractères quelconques.
- **Les booléens:** Ce sont les nombres booléens qui peuvent avoir deux valeurs, soit **1** ou **0** exprimés de préférence par **vrai (true)** ou faux (**false**).
- **Les variables vides:** Ce sont des variables qui ne contiennent aucune valeur, ils ont la valeur implicite **null**.

Les variables

Changement de type d'une variable (CAST)

Donc quelque soit la valeur de notre variable, elle fait forcément partie de l'un des types précédents. Or, sur la plupart des langages de programmation (comme le C), on doit conserver le type de la variable du début à la fin du programme. Par exemple, si nous déclarons une variable de type **int**, alors toutes les valeurs qu'on doit lui affecter au cours du programme doivent être des nombres entiers. Si, pour une raison quelconque, on veut lui affecter un booléen par exemple, alors on doit "caster" la variable (le CAST signifie conversion de type).

En revanche, en Javascript, on peut changer le type de la variable à la volée. Donc au cours du même programme, on peut affecter à notre variable n'importe quelle valeur de n'importe quel type. Le CAST est donc fait automatiquement.

Les variables

Changement de type d'une variable (CAST)

Exemple:

```
a=12; // Nombre  
a="Bonjour"; // Chaîne de caractères  
a=true; // Booléen
```

Si on exécute ce bout de code, il n'y aura aucun problème. La variable **a** a été initialisée en tant que nombre, puis a changé de type deux fois (Chaîne de caractères puis booléen).

Les opérateurs

Les opérateurs sont des symboles qui permettent de faire des opérations (arithmétiques, logiques...) sur les variables. Généralement, on retrouve les même opérateurs dans la plupart des langages de programmation avec des symboles, dans la plupart des cas, similaires.

En Javascript (et en d'autres langages d'ailleurs) on distingue 5 familles d'opérateurs:

- **Opérateurs arithmétiques:** l'addition (+), la soustraction (-), la multiplication (*), la division (/) et le modulo (qui signifie le reste de la division)(%).
- **Opérateurs d'incrémentation:** incrémentation (++) et décrémentation (--)
- **Opérateurs assignement:** : l'affectation directe (=), l'affectation avec addition (+=), l'affectation avec soustraction (-=), l'affectation avec multiplication (*=), l'affectation avec division (/=) et l'affectation avec modulo (%=).
- **Opérateurs de comparaison:** égal (==), strictement inférieur (<), inférieur ou égal (<=), strictement supérieur (>), supérieur ou égal (>=), différent (!=)
- **Opérateurs logiques:** ET logique (&&), OU logique (||) et NON logique (!)

Chapitre 3: Le langage Javascript

Les bases du Javascript?

Exemple:

```
a=10;  
b=20;  
c=a+b; // c vaut 30  
d=a-b; // d vaut -10  
e=a*b; // e vaut 200  
f=a/b; // f vaut 0.5  
g=a%b; // g vaut 10
```

Exemple:

```
a=10;  
if(a%2==0)  
    res="Nombre pair";
```

Exemple:

```
a=10;  
a++; // a vaut donc 11  
a--; // a vaut à nouveau 10
```

```
a=10;  
b=a; // b vaut 10  
a+=5; // a vaut 15  
a-=3; // a devient 12
```

Chapitre 3: Le langage Javascript

Structures de contrôle

Structures de contrôles en Javascript

```
a=10;  
if(a%2==0)  
    b="Nombre pair";  
else  
    b="Nombre impair";
```

- Possibilité d'imbriquer.
- If elseif else

```
str="";  
for(i=1;i<5;i++) {  
    str+=i;  
}
```

```
str="";  
i=1;  
while(i<5) {  
    str+=i;  
    i++;  
}
```

```
a=3;  
switch(a) {  
    case 1: b="Un"; break;  
    case 2: b="Deux"; break;  
    case 3: b="Trois"; break;  
    default: b="Aucun des trois";  
}
```

Structures de contrôles en Javascript

Arrêt prématûr: instruction break

Exemple:

```
str="";
i=1;
while(i<5) {
    if(i==3) {
        break;
    }
    str+=i;
    i++;
}
```

Saut d'une itération: instruction continue

Exemple:

```
str="";
i=0;
while(i<5) {
    i++;
    if(i==3) {
        continue;
    }
    str+=i;
}
```

Les fonctions

Une fonction est un sous-programme qui contient un ensemble d'instructions. Ces instructions ne seront exécutées suite à l'appel de la fonction.

Les fonctions ont de nombreux avantages qu'on va résumer dans les points suivants:

- **Réutilisation du code**
- **Réduction de la taille du code**
- **Organisation du code**
- **Récursivité**

Les fonctions constituent un aspect fondamentale quand on programme en Javascript.

Où placer nos fonctions?

Javascript est un langage interprété. Par conséquent, on ne peut pas appeler une fonction qu'après l'avoir créée (ou déclarée). Cependant la quasi totalité des appels se font à partir de la balise **<body>**, il faut donc monter un peu plus haut dans le code HTML pour déclarer nos fonctions Javascript. Y'a pas mieux que le **<head>**!

Déclaration d'une fonction

```
function Nom_de_la_fonction(liste_des_arguments) {  
    Corps_de_la_fonction;  
    return Valeur_de_retour;  
}
```

Fonction récursive

Une fonction récursive est une fonction dans laquelle figure un appel à elle même.

Dans une fonction récursif, il faut toujours prévoir une condition de sortie qui cessera son exécution. Sinon, la fonction se rappellera indéfiniment.

Le retour d'une valeur à l'aide des instructions tels que **return**, **return true**, **return false**, **return valeur**... force l'arrêt de la fonction et par conséquent l'arrêt de la récursivité.

Exemple:

```
a=0;  
function f () {  
    a+=1;  
    if (a>=5)  
        return;  
    f(); // Appel récursif  
}  
f(); // Premier appel  
alert(a);
```

Javascript: un langage événementiel

Javascript est un langage événementiel. Donc les scripts déclarés dans la page ne sont pas destinés à s'exécuter séquentiellement, ligne après ligne, du début à la fin. Mais ils attendent jusqu'à ce qu'un événement soit détecté pour qu'une partie du code s'exécute.

Un **événement** est une action de l'utilisateur qui peut se produire sur le navigateur. Le plus célèbre est sans doute le **clic** qui peut survenir sur un objet quelconque (hyperlien, image, bouton...). Le navigateur écoute les événements qui se produisent sur la page Web puis les transmet au script par le biais de ce que l'on appelle **le gestionnaire d' événements**.

Javascript: un langage événementiel

Gestionnaire d'événements

Le **gestionnaire d'événements** permet d'associer un traitement à un événement. Sa syntaxe respecte la forme suivante:

onÉvénement = "code Javascript édité directement ici, ou fonction appelée à cet endroit"

Il faut noter que **onÉvénement** est un attribut HTML, et ce qui est entre les guillemets est sa valeur. Donc la ligne précédente est intégrée directement dans une balise HTML. Cependant, même si ce qui se trouve entre les guillemets est purement du code Javascript, nous ne le mettons pas dans les tags de script (**<script>** et **</script>**).

Il est très rare où on déclare directement la suite d'instructions Javascript à exécuter entre les guillemets. En général, on y appelle une fonction qu'on a déjà déclaré avant dans la balise **<head>**. Ainsi nous séparons les code HTML et Javascript ce qui rend le code plus lisible et plus facile à maintenir.

Javascript: un langage événementiel

```
<html>
  <head>
    <meta charset="UTF-8" />
    <script language="javascript">
      function f() {
        alert("Vous avez bien cliqué sur le bouton!");
      }
    </script>
  </head>
  <body>
    <input type="button" value="Cliquez ici" onClick="f()" />
  </body>
</html>
```

Le fait de cliquer sur le bouton donne naissance à l'événement **click** qui appellera (grâce au gestionnaire d'événements) la fonction **f()** définie dans le **<head>**. On aura donc le message "Vous avez bien cliqué sur le bouton!" qui s'affichera dans une boîte de dialogue **alert()**.

Javascript: un langage événementiel

Liste des événements Javascript

click (onClick)

Cet événement est capturé sur un objet quand on clique dessus. Idéal pour les boutons, images, hyperliens, vidéos...

dblClick (onDoubleClick)

Quand on clique sur un objet deux fois de suite (double clic).

mouseOver (onMouseOver)

Quand on survole un objet avec le curseur de la sourie.

submit (onSubmit)

Quand l'internaute clique sur n'importe quel bouton de type **submit** présent dans la page (ou dans le formulaire).

Javascript: un langage événementiel

Liste des événements Javascript

mouseout (onMouseOut)

Quand on quitte l'objet avec la souris après l'avoir survolé.

focus (onFocus)

Quand on active un élément (quand on place le curseur dans un champ de formulaire, par click ou par tabulation, pour commencer la saisie par exemple).

blur (onBlur)

Quand un élément perd le focus (quitter un champ de formulaire après être activé par exemple).

keydown (onKeyDown)

Quand une touche du clavier est enfoncée.

keyup (onKeyUp)

Quand une touche du clavier est relâchée.

mouseDown (`onMouseDown`)

Quand l'internaute appuie sur n'importe quel bouton de la souris.

mouseUp (`onMouseUP`)

Quand le bouton de la souris est relâché.

mouseMove (`onMouseMove`)

Quand l'internaute fait bouger le curseur de la souris dans la zone accueillant l'événement.

dragStart (`onDragStart`)

Se produit quand l'internaute commence le déplacement d'un élément par "glisser-déposer" (Drag & Drop).

dragEnter (`onDragEnter`)

Se produit quand l'internaute entre dans la zone où sera déposé l'élément glissé.

dragOver (`onDragOver`)

Se produit quand l'internaute survole la zone où sera déposé l'élément glissé (même si le concept est proche de celui de **onDragEnter**, ils sont des événements légèrement différents).

drop (`onDrop`)

Se produit quand l'internaute dépose l'élément glissé dans la zone prévue à cet effet.

Chapitre 3: Le langage Javascript

Les événements en Javascript

```
<html>
  <head>
    <meta charset="UTF-8" />
    <script language="javascript">
      function f() {
        alert("Vous avez cliqué sur le bouton 1.");
      }
      function g() {
        alert("Vous avez survolé le bouton 2.");
      }
      function h() {
        alert("Vous avez cliqué 2 fois sur le bouton 3.");
      }
    </script>
  </head>
  <body onLoad="alert('Page chargée.')">
    <input type="button" value="Bouton 1" onClick="f()" /><br />
    <input type="button" value="Bouton 2" onMouseOver="g()" /><br />
    <input type="button" value="Bouton 3" onDblClick="h()" /><br />
  </body>
</html>
```

Javascript: langage de programmation orienté prototype

Javascript est un langage de programmation orienté prototype.

Il renferme une panoplie d'objets prédéfinies (appelés prototypes) qui peuvent donner naissance à d'autres objets que l'on souhaite créer.

Puisque Javascript est considéré comme une extension du langage HTML, alors la plupart de ses objets sont en réalité des objets HTML présents dans le document, comme les images, les champs de formulaire... Par contre, il existe des objets propres au Javascript qui permettent d'exécuter des traitement comme le calcul, la manipulation de la date, le traitement des chaines de caractères...

Hiérarchie des objets

Les objets Javascript qui dépendent du contenu d'un document HTML sont structurés hiérarchiquement. C'est à dire que des objets s'emboîtent dans d'autres jusqu'à atteindre l'objet le plus élevé à l'échelle de la hiérarchie.

L'objet le plus élevé est la fenêtre du navigateur, il est identifié par le nom **window**. Cet objet peut être manipulé directement à travers des fonctions qui lui sont prédéfinies. On appelle ces fonctions des **méthodes**. A titre d'exemple **alert()** qui affiche les messages des exemples précédents est une méthode de l'objet **window**, on aurait pu donc l'écrire comme ceci: **window.alert()**.

Le point (.) permet:

- de passer d'un objet à une de ces méthodes
- d'un objet parent à un autre objet enfant
- d'un objet à une variable qui lui est propre. On appelle cette variable un **attribut**.

- Un **objet** peut représenter un élément HTML (image, texte, formulaire...) ou un élément propre au Javascript qui sert à exécuter des traitements spéciaux.
- Une **méthode** est une **fonction** prédéfinie appartenant à l'objet et qui permet de faire les traitement sur celui-ci (on peut dire qu'elle décrit le comportement de l'objet).
- Un **attribut** est une **variable** qui appartient à l'objet . il représente une propriétés ou caractéristique de celui-ci. On peut le manipuler grâce aux **méthodes** ou directement.

L'objet **window** contient un sous-objet (ou un objet subalterne), c'est le document HTML lui même. Cet objet est identifié par le nom **document**. Cet objet peut contenir les éléments HTML conventionnels comme du texte, les images, les formulaires... Tous ces éléments constituent des sous-objets de l'objet **document**. Par exemple l'objet **forms** désigne les formulaires contenus dans le document, et l'objet **images** englobe toutes les images intégrées...

L'objet Array

Les tableaux sont un type spécial de variables qui permettent de contenir plusieurs valeurs à la fois. Si une variable classique peut avoir une seule valeur, les tableaux, quant à eux, peuvent en contenir autant que l'on souhaite.

En Javascript, un tableau est un objet. Il peut contenir des entrées du même type et de types différents. Il peut même contenir d'autres tableaux. On parle alors d'un tableau à 2 dimensions ou à 3 dimensions etc...

Première méthode:

```
tab=new Array();
tab[0]="Javascript";
tab[1]="Coté client";
tab[2]=40;
```

Deuxième méthode:

```
tab=new Array();
tab=["Javascript", "Coté client", 40];
```

Troisième méthode:

```
tab=new Array("Javascript", "Coté client", 40);
```

Tableau à plusieurs dimensions

```
tab1=new Array("Entreprise A", 40, 1000000);
tab2=new Array("Entreprise B", 90, 5000000);
tab3=new Array("Entreprise C", 110, 9000000);
```

```
tab=new Array(tab1, tab2, tab3);
```

L'objet Array

Parcourir un tableau

Pour parcourir tous les éléments d'un tableau, la méthode la plus connue et l'utilisation de la boucle **for** ou la boucle **while**.

```
str="";
for(i=0;i<3;i++) {
    for(j=0;j<3;j++) {
        str+=tab[i][j]+" ";
    }
}
alert(str);
```

Attributs et méthodes de l'objet Array

- L'objet **Array** dispose d'un ensemble d'attributs et méthodes qui permettent de manipuler les tableaux. En voici les plus utilisés:
- **length**: Cet attribut retourne la taille du tableau, c'est à dire le nombre d'élément que celui ci contient.
- **sort()**: Cette méthode retourne le tableau dont les éléments sont triés dans l'ordre alphanumérique croissant.
- **reverse()**: Cette méthode retourne le tableau dont l'ordre des éléments a été inversé.
- **join()**: Cette méthode retourne une chaîne de caractères constituée des éléments du tableau séparés par l'occurrence passée en paramètres (entre les parenthèses).

Attributs et méthodes de l'objet Array

```
tab=new Array("A", "B", 10);
tab.length;
/* Retourne 3 */

tab.sort();
/* Retourne un tableau dont les éléments sont déclarés dans cet ordre: 10, A, B */

tab.reverse();
/* Retourne un tableau dont les éléments sont déclarés dans cet ordre: 10, B, A */

tab.join("-");
/* Retourne la chaîne de caractères "A-B-10" */

tab.sort().reverse().join("-");
/* Retourne la chaîne de caractères: "B-A-10" */
```

L'objet String

Normalement, une chaîne de caractère représente un type pour les variables. C'est le cas en Javascript mais on parle plutôt d'un objet au lieu d'une variable.

Pour déclarer un objet **String** nous n'avons pas vraiment besoin de faire appel au mot clé **new** comme on a fait pour l'objet **Array**. Il suffit d'affecter la chaîne voulue à l'objet comme si on affecte une valeur à une variable.

```
str="Bonjour";
```

str dans ce cas n'est pas une variable proprement dite, mais il s'agit en fait d'un objet. Afin de manipuler les chaînes de caractères (objets String) nous avons à notre disposition un ensemble d'attributs et de méthodes.

Chapitre 3: Le langage Javascript

Objet String: chaînes de caractère

```
str="Bonjour";
str.length;
/* Retourne 7 */

str.charAt(0);
/* Retourne B */

str.indexOf("o");
/* Retourne 1 */

str.indexOf("o", 2);
/* Retourne 4 */

str.indexOf("x");
/* Retourne -1 */

str.substring(0, 3);
/* Retourne Bon */

str.substring(3, 6);
/* Retourne jou */
```

```
tab=str.split("o");

tab.length;
/*Retourne 3 */

tab.join("-");
/*Retourne B-nj-ur */
```

En fait, **str.split("o")** découpe la chaîne de caractères au niveau des lettres "o". Puisqu'il y'en a deux dans "Bonjour", alors on obtient trois fragments "B", "nj" et "ur". Chaque portions constitue une entrée du tableau **tab** (**tab[0]**="B", **tab[1]**="nj" et **tab[2]**="ur").

Vous avez donc compris que la méthode **join()** qui s'applique aux tableaux et la méthode **split()** qui s'appliquent aux chaînes de caractères sont complémentaire. Par conséquent si on avait déclaré **tab.join("o")** à la place de la dernière instruction du code, on aurait obtenu "Bonjour".

Attributs et méthodes agissant sur les chaînes de caractères

- **length:** Cet attribut retourne le nombre de caractères contenus dans la chaîne. Notez que si on applique le même attribut à un tableau, il retourne la taille de celui-ci.
- **charAt():** La méthode charAt(x) (avec A majuscule) permet de retourner le caractères qui se trouve à la position x passé en paramètre. Le paramètre est un entier qui commence de 0. La valeur 0 indexe le premier caractère de la chaîne.
- **indexOf():** La méthode indexOf(car) (avec O en majuscule) permet de retourner la position du caractères car passé en paramètre. Si le caractère existe dans la chaîne, alors sa position (comprise entre 0 et la longueur de la chaîne - 1) est retournée, sinon (le caractère ne figure pas dans la chaîne) alors la valeur -1 est retournée.
 - La méthode **indexOf()** peut accueillir un deuxième paramètre qui est un entier qui indique à partir de quel position de la chaîne on commencera la recherche du caractère passé en premier paramètre.

Attributs et méthodes agissant sur les chaînes de caractères

- **substring():** La méthode substring(début,fin) permet d'extraire une partie de la chaîne de caractères commençant de la position **début** et finissant à la position **fin-1**.
- **split():** La méthode split(occurrence) permet de retourner un tableau à partir des fractions de la chaîne de caractères obtenues en divisant celle-ci au niveau de l'**occurrence**.
- **toLowerCase():** La méthode toLowerCase() permet de retourner la chaîne de caractère entièrement en minuscules.
- **toUpperCase():** La méthode toUpperCase() permet de retourner la chaîne de caractère entièrement en majuscules.

Objet Math pour manipuler les fonctions mathématiques

L'objet **Math** de Javascript permet d'exécuter des fonctions mathématiques par le biais de ses nombreux attributs et méthodes.

Fonctions mathématiques à usage standard

- **Math.abs(x)**: calcule la valeur absolue de x.
- **Math.ceil(x)**: retourne l'arrondi supérieur de x. Il s'agit du nombre entier immédiatement supérieur ou égal à x.
- **Math.floor(x)**: retourne l'arrondi inférieur de x. Il s'agit du nombre entier immédiatement inférieur ou égal à x.
- **Math.round(x)**: retourne l'arrondi le plus proche (supérieur ou inférieur) de x.
- **Math.max(x,y)**: retourne le nombre plus grand parmi x et y.
- **Math.min(x,y)**: retourne le nombre plus petit parmi x et y.
- **Math.pow(x,y)**: retourne x à la puissance y.
- **Math.sqrt(x)**: retourne la racine carré de x.

Objet Math pour manipuler les fonctions mathématiques

Fonctions mathématiques spéciales

- **Math.PI**: il s'agit d'un attribut qui retourne la valeur de PI de la trigonométrie (le fameux 3.14 avec quelques chiffres en plus à droite).
- **Math.E**: cet attribut retourne la valeur du nombre d'Euler (arrondi à 2.72).
- **Math.exp(x)**: calcule l'exponentiel de x.
- **Math.sin(x)**: calcule le sinus de x.
- **Math.asin(x)**: calcule l'arcsinus de x (x doit être compris entre -1 et 1).
- **Math.cos(x)**: retourne le cosinus de x.
- **Math.acos(x)**: retourne l'arccosinus de x (x doit être compris entre -1 et 1).
- **Math.tan(x)**: calcule la tangente de x.
- **Math.atan(x)**: calcule l'arctangente de x.

Objet Math pour manipuler les fonctions mathématiques

Nombres aléatoires

En Javascript (comme en d'autres langages de programmation), il est parfois utile de générer des nombres aléatoires. La méthode **random()** (pour **Math.random()**) permet de retourner un nombre aléatoire compris entre 0 et 1. Ce nombre généré contient plusieurs chiffres après la virgule. Pour générer un nombre aléatoire entier une technique très simple est utilisée.

Supposons que nous voulons générer un nombre aléatoire compris entre 0 et 10 que l'on va affecter à la variable **nbrAltr**. Dans ce cas le code sera:

```
nbrAltr=Math.round(Math.random()*10);
```

Manipuler les dates en Javascript

Il est parfois nécessaire de manipuler les dates et les heures au sein d'une application Web. Par exemple, créer un calendrier pour pouvoir sélectionner la date de départ d'un train plus facilement, ou afficher l'horloge sur sa page Web, ou encore calculer la durée de séjour d'un client qui compte réserver une chambre d'hôtel sur un site d'hébergement.

Cependant, il ne faut pas trop se fier à la date du client, car celle-ci peut ne pas être précise. Mais, il est utile de savoir comment manipuler les dates en Javascript.

Objet Date

Pour créer un objet Date on fait comme ceci:

```
d=new Date();
```

d est l'objet résultatnt de l'objet (prototype) Date() de Javascript. Le mot clé **new** signifie "nouvelle" date dans ce cas.

Manipuler les dates en Javascript

Si les parenthèses de **Date()** sont vides, cela signifie qu'on veut créer l'objet date à partir de la date courante du système. Par contre on peut toujours faire référence à une date différente en la précisant entre les parenthèses comme ceci:

```
Date = new Date("Mar 20, 2021 19:00:00")
```

Méthodes de l'objet Date

- **getFullYear()**: permet de retourner l'année sur 4 chiffres.
- **getDay()**: retourne un indice numérique qui représente le jour de la semaine. 0 pour dimanche, 1 pour lundi, ... , 6 pour samedi.
- **getMonth()**: retourne un indice numérique qui représente le mois. 0 pour janvier, 1 pour février, ... , 11 pour décembre.
- **getHours()**: retourne l'heure (sans zéro initial). Si la date crée fait référence à 9h du matin, alors cette méthode retournera 9 et non pas 09.
- **getMinutes()**: retourne les minutes (sans zéro initial).
- **getSeconds()**: retourne les secondes (sans zéro initial).
- **getMilliSeconds()**: retourne les millisecondes (rarement utilisé).
- **getTime()**: retourne ce que l'on appelle le Timestamp UNIX (ou Timestamp POSIX) en millisecondes. C'est à dire le nombre de millisecondes écoulées depuis 01/01/1970 00:00:00. C'est la date où le premier système UNIX a été lancé.

Expressions régulières

Les expressions régulières sont des techniques qui permettent de chercher des modèles (ou des occurrences) de formes qui peuvent être complexes, au sein des chaînes de caractères.

Pour simplifier, imaginez qu'on invite un client à saisir une adresse email dans une zone de texte. Afin de s'assurer si ce qu'il a entré respecte la forme usuelle d'un email, on peut vérifier certains points:

- Y a-t-il un arobas dans l'email?
- Y a-t-il un point?
- Est-ce qu'il y a au moins un caractère avant l'arobas?
- Le nom de domaine est-il valide?
- Est-ce que l'email ne contient pas de caractères non autorisés comme les espaces?

Expressions régulières

Objet RegExp

C'est grâce à l'objet **RegExp** que l'on peut créer un model (ou occurrence). La syntaxe ressemble à ceci:

```
expression=new RegExp ("occurrence");
```

expression est l'objet qui contient le model qu'on a créé à l'aide de l'objet **RegExp**. En paramètres de celui ci, nous déclarons une chaîne de caractères qui représente l'expression régulière elle même.

caractères spécial	Signification
.	Un seul caractère quelconque
+	Le motif précédent doit figurer au moins une fois, le max n'est pas déterminé
*	Le motif précédent doit figurer au moins 0 fois, le max n'est pas déterminé
()	Pour grouper un ensemble de caractères en tant que motif unique
[]	Pour désigner un ensemble de motifs dont , au moins, l'un d'entre eux doit figurer
-	Désigne un intervalle s'il est déclaré entre les crochets. Exemple: [a-z] signifie l'alphabet minuscule
?	Le motif précédent doit figurer 0 ou une fois.
^	Le motif suivant doit figurer au début de la chaîne
\$	Le motif précédent doit figurer à la fin de la chaîne
{ }	Pour spécifier le minimum et le maximum de fois où l'occurrence doit figurer. Exemple: {1,3} désigne que l'occurrence figure entre 1 et 3 fois.

Expressions régulières

il faut déspécialiser les caractères spéciaux si on souhaite les chercher en tant que motif (ou occurrence).

```
expression=new RegExp ("\^") ;
```

Méthodes de l'objet RegExp

La méthode la plus utilisée pour l'objet RegExp s'appelle **test()**. La méthode **test()** permet de tester si la chaîne de caractères passée en paramètre (dans les parenthèses) correspond à l'expression régulière. Si oui, elle retourne la valeur booléenne **true**, sinon elle retourne **false**. On peut donc la manipuler grâce à la structure de contrôle **if else**.

Expressions régulières

Méthodes de l'objet RegExp

```
email="user@domaine.tld";
emailValide=new RegExp ("^ .+ @ .+ \. .+ " );
if(emailValide.test (email) )
    alert ("Email valide .");
else
    alert ("Email invalide .");
```

Début de la chaîne

Un caractère

Répéter

Eliminer la signification
spéciale du point

L'objet window: le parent de tous les objets de la page

Redirection: attribut location

Le fait que les liens hypertextes représentent le moyen le plus évident pour rediriger le navigateur vers une autre page relève de l'unanimité. Or, pour changer de page, il faut absolument que l'internaute clique sur le lien. Mais imaginons un instant qu'on veut rediriger le navigateur vers une nouvelle page sans cliquer sur aucun lien (suite à un événement ou un traitement spécial par exemple). Il nous faut donc une autre solution.

En Javascript l'attribut **location** permet de rediriger le navigateur vers la page dont le nom lui est affecté en tant que chaîne de caractères. Pour faire simple, imaginons que nous voulons aller vers la page nommée "contact.html" via Javascript. Le code ressemblera à ceci:

L'objet window: le parent de tous les objets de la page

```
window.location="contact.html";
```

Ecrire dans la barre d'état: attribut status

La barre d'état est la petite barre qui se trouve (d'une manière permanente ou temporaire) en bas du navigateur et qui affiche souvent des informations comme la cible d'un lien hypertexte survolé ou l'état de chargement des éléments de la page.

Javascript permet de personnaliser le message à afficher sur la barre d'état grâce à l'attribut **status**. Le message souhaité lui est affecté en tant que chaîne de caractères.

Exemple:

```
window.status="Barre d'état dont le message a été personnalisé.;"
```

Méthodes de boites de dialogue: alert(), confirm() et prompt()

Les méthodes de boites de dialogue figent le navigateur. En effet, le fait d'afficher un message dans une boite de dialogue n'est qu'une tâche complémentaire de ces méthodes. Leur rôle principal c'est de figer le navigateur et l'empêcher de poursuivre l'exécution du reste du code (que ce soit HTML, CSS ou Javascript lui même). Cependant, il faut dire à l'internaute pourquoi son navigateur est figé. C'est à cela que sert le message affiché.
Le navigateur sera de nouveau libéré si le client clique sur le bouton (ou les boutons) qui se présente sur la boite de dialogue.

Méthodes de boites de dialogue: alert(), confirm() et prompt()

- **alert()**: Cette méthode permet d'afficher un message dans une boite de dialogue en figeant le navigateur. Elle affiche aussi un bouton "OK". En cliquant dessus, la boite disparaît et le navigateur est libéré.
- **confirm()**: Cette méthode permet d'afficher un message dans une boite de dialogue menue de deux boutons: "OK" et "Annuler". En cliquant sur "OK", la méthode confirm() retourne la valeur booléenne **true**. Le bouton "Annuler" retourne **false**. Cette méthode sert principalement à faire confirmer une action par le visiteur.
- **prompt()**: La méthode prompt() permet d'afficher un message dans une boite de dialogue. Cette boite contient aussi une zone de texte et deux boutons "Ok" et "Annuler". Si on clique sur "OK" la chaîne de caractères saisie dans la zone de texte est retournée par la méthode. En cliquant sur "Annuler" la valeur "null" (qui signifie "variable vide") est retournée. La méthode prompt() peut accueillir deux paramètres. Le premier est affiché dans la boite de dialogue et le deuxième constitue le texte par défaut de la zone de texte.

```
<html>
  <head>
    <meta charset="UTF-8" />
    <script language="javascript">
      function f1() {
        alert("Bouton Alert actionné.");
      }
      function f2() {
        if(confirm("Voulez-vous continuer?"))
          alert("Vous avez accepté de continuer.");
        else
          alert("Vous avez refusé de continuer.");
      }
      function f3() {
        alert("Votre nom est: "+prompt("Veuillez entrer votre nom svp", "Votre nom ici"));
      }

    </script>
  </head>
  <body>
    <input type="button" value="Alert" onClick="f1()" />
    <input type="button" value="Confirm" onClick="f2()" />
    <input type="button" value="Prompt" onClick="f3()" />
  </body>
</html>
```

Pop-Up: méthodes open() et close()

Les pop-up sont des fenêtres qui s'ouvrent indépendamment de la fenêtre principale. Elle contiennent généralement de la publicité ou des informations complémentaires à la page principale. De nombreux navigateurs ont tendance à bloquer l'ouverture par défaut des pop-up car, parfois, leur ouverture intempestive importune l'internaute.

Pour ouvrir une fenêtre on utilise la méthode **open()** et pour la fermer, en plus du bouton de fermeture de la fenêtre par défaut, on peut faire appel à la méthode **close()**.

La méthode **open(URL, Nom de la fenêtre, Options)** reçoit trois paramètres:

- **URL:** Il s'agit du chemin de la page à ouvrir dans la fenêtre. Il peut être absolu ou relatif.
- **Nom de la fenêtre:** Il s'agit d'un nom optionnel que l'on donne à la fenêtre. Il est utile quand plusieurs fenêtres sont ouvertes. S'ils disposent de noms différents, alors on se retrouvera avec autant de fenêtre que d'appels. S'ils ont le même nom, alors un nouvelle fenêtre ouverte écrase l'ancienne.
- **Options:** Les options permettent de personnaliser l'apparence de la fenêtre. Les options possibles sont:

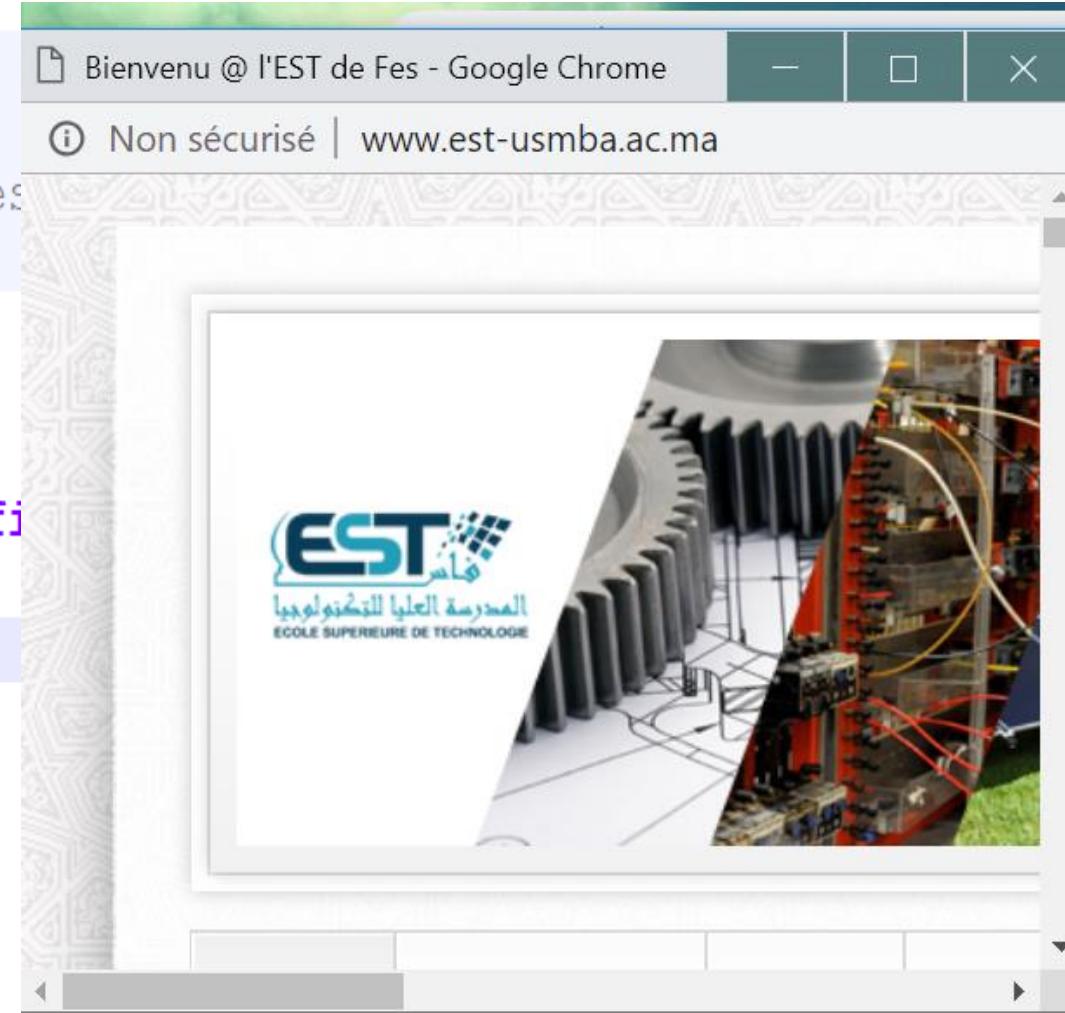
- **width**: spécifie la largeur de la fenêtre en pixel (l'unité n'est pas spécifiée).
- **height**: hauteur de la fenêtre.
- **location**: peut avoir **yes** ou **no** pour autoriser ou non l'apparition de la barre d'adresse.
- **resizable**: peut avoir **yes** ou **no** pour autoriser ou non le redimensionnement de la fenêtre.
- **menubar**: peut avoir **yes** ou **no** pour autoriser ou non l'apparition du menu du haut de la fenêtre.
- **scrollbar**: peut avoir **yes** ou **no** pour autoriser ou non l'apparition des barres de défilement
- **toolbar**: peut avoir **yes** ou **no** pour autoriser ou non l'affichage de la barre d'outils.
- **status**: peut avoir **yes** ou **no** pour autoriser ou non l'apparition de la barre d'état.

Les options doivent être séparées d'une virgule (sans espaces). Selon le navigateur, toute option non déclarée reçoit implicitement la valeur **no**.

Chapitre 3: Le langage Javascript

Objet window

```
<html>
  <head>
    <meta charset="UTF-8" />
    <script language="javascript">
      function popup() {
        window.open("http://www.es")
      }
    </script>
  </head>
  <body>
    <input type="button" value="Afficher" onclick="popup()" />
  </body>
</html>
```



ursConceptionWeb/testPopUp.htm

Accueil - Portail Ma...

Chapitre 3: Le langage Javascript

Objet window

Temporisateur: méthodes `setTimeout()` et `clearTimeout()`

Il faut dire que la méthode `setTimeout()` est largement utilisée en Javascript.

```
<html>
  <head>
    <meta charset="UTF-8" />
    <script language="javascript">
      function f() {
        alert("Execution retardée de 2 secondes.");
      }
      function g() {
        setTimeout("f()",2000);
      }
    </script>
  </head>
  <body>
    <input type="button" value="Actionner" onClick="g()" />
  </body>
</html>
```

Chapitre 3: Le langage Javascript

Objet event

```
<html>
  <head>
    <meta charset="UTF-8" />
  </head>
  <body>
    <section>
      <div>
        <a href="#" onClick="f ()">Hyperlien</a>
      </div>
    </section>
  </body>
</html>
```

L'objet **Document** est hiérarchiquement placé sous l'objet **window**. Il permet de manipuler tous les objets qui sont inclus dans le documents HTML (images, textes, formulaires...).

Quand on parle de l'objet **document** il y a un autre terme qui surgit automatiquement, c'est le **DOM**.

Document Object Model: DOM

DOM qui signifie **Document objet Model** est un outil qui permet d'accéder au document. Il s'agit plus précisément d'une interface qui standardise la manière d'accès aux document structurés en balises comme HTML ou XML. D'ailleurs, chaque élément du document fait partie du model DOM.

Document Objet Model: DOM

Arborescence et nœuds

Nous avons déjà parlé de la hiérarchie des objets en Javascript. Il s'agit en fait de la manière dont sont présentés les différents éléments constituant le contenu (un objet parent qui contient des objets enfants qui, à leurs tours, contiennent d'autres objets et ainsi de suite).

L'ensemble des ces objets constitue une arborescence (connue aussi sous le nom **arbre HTML**, si le document en question est HTML). Cet arbre est en fait une structure d'éléments qui représentent des **nœuds**.

Il existe deux types de nœuds, nœuds éléments et nœuds texte. Les nœuds éléments sont tout simplement les balises et les nœuds textes sont les textes contenus dans un document HTML.

```
<div>Bonjour <span>à tous</span></div>
```

Document Objet Model: DOM

Méthodes et sous-objets

Méthodes de base de l'objet document

Il existe de nombreux méthodes (et attributs) qui s'appliquent sur l'objet **document**.

- **write()**: La méthode **write()** permet d'écrire la chaîne de caractères passée en paramètre directement dans le document.

```
document.write("Bonjour") ; // Affiche "Bonjour" dans la page.
```

```
document.writeln("Bonjour") ; // Affiche "Bonjour" suivi d'un retour à la ligne dans la page.
```

Sous objets de document

Trois sous-objets sont fréquemment utilisés:

- **images**: Cet objet retourne un tableau qui contient tous les objets de type image (c'est à dire les balises ``). La première image du document est indexée par 0, la deuxième par 1 et ainsi de suite. L'attribut le plus utilisé de cet objet est **src** qui permet de changer la source de l'image dynamiquement.
- **forms**: Cet objet retourne un tableau qui contient tous les formulaires du document (les balises `<form>`).
- **body**: Cet objet fait référence à la balise `<body>`.

Sous objets de document

```



<script language="javascript">
    for(i=0;i<document.images.length;i++)
        document.images[i].src="image4.jpg";
</script>
```

Exemple 2:

```
<form>
    <input type="text" name="login" />
</form>
<script language="javascript">
    document.forms[0].login.value="ABC";
</script>
```

Autres façons pour accéder aux sous objets de document

Jusqu'ici nous avons vu comment écrire directement dans le document et comment accéder aux images et formulaires déclarés. Cependant, il existe d'autres méthodes qui permettent d'accéder plus facilement aux différents objets de tout type d'un document. Il s'agit d'ailleurs des méthodes les plus utilisées par les développeurs:

- **getElementById()**: Cette méthode permet d'accéder à un objet quelconque en l'identifiant par la valeur de son attribut **id**. Nous avons déjà eu l'occasion de voir cet attribut en CSS. Sa valeur est unique dans tout le document. C'est cette valeur qui sera déclarée dans les parenthèses de la méthode en tant que chaîne de caractères.
- **getElementsByName()**: Cette méthode permet d'accéder tous les objets correspondant à la balise (tag) déclarée entre les parenthèses en tant que chaîne de caractères. Elle retourne un tableau qui renferme tous les nœuds du type désigné.

Autres façons pour accéder aux sous objets de document

```



<script language="javascript">
    document.getElementById("im1").src="image4.jpg";
    for(i=1;i<document.getElementsByTagName("img").length;i++)
        document.getElementsByTagName("img")[i].src="image4.jpg";
</script>
```

Le premier objet image a été désigné par son identifiant "im1". Les deux derniers ont été désignés par leur nom de balise (TagName) qui correspond évidemment à "img". Puisqu'il s'agit d'un tableau, alors on a précisé les indexes des objets à cibler entre les crochets.

Autres façons pour accéder aux sous objets de document

Après avoir ciblé nos objets, que pouvons nous faire à part changer la source des images et désigner une zone de texte?

Attribut innerHTML

Jusqu'ici, nous avons pu écrire des textes dans des champs de formulaire et des boites de dialogue. Mais si on veut écrire n'importe où dans la balise **<body>** alors on peut le faire grâce à l'attribut **innerHTML**.

Autres façons pour accéder aux sous objets de document

Attribut innerHTML

```
<div id="contenu">Bonjour</div>
<script language="javascript">
    document.getElementById("contenu").innerHTML="Bonsoir";
</script>
```

Au chargement de la page, la balise **<div>** identifiée par **contenu** contient le texte "Bonjour". Une fois le script Javascript exécuté, le nœud texte "Bonjour" est remplacé par "Bonsoir" grâce à l'attribut **innerHTML**.

Autres façons pour accéder aux sous objets de document

Attribut `className`

L'attribut `className` renomme l'attribut `class` d'un élément. C'est pratique pour appliquer des styles CSS dynamiquement à l'objet en question.

Chapitre 3: Le langage Javascript

Objet DOM

```
<html>
  <head>
    <meta charset="UTF-8" />
    <style>
      .nouvelleClasse{
        color:#FFFFFF;
        font-size:18pt;
        background-color:#000000;
        padding:10px;
      }
    </style>
    <script language="javascript">
      function changer(){
        document.getElementsByTagName("h1").item(0).className="nouvelleClasse";
      }
    </script>
  </head>
  <body>
    <h1>Titre d'ordre 1</h1>
    <input type="button" value="Changer l'apparence" onclick="changer()" />
  </body>
</html>
```

Autres façons pour accéder aux sous objets de document

Sous-objet style

Le sous-objet **style** permet de définir dynamiquement un attribut de style local **style** sur la balise concernée.

Exemple:

```
<span id="citation">
    Mieux vaut tard que jamais.
</span>
<script language="javascript">
    document.getElementById("citation").style.color="#888888";
    document.getElementById("citation").style.fontStyle="italic";
</script>
```

Autres façons pour accéder aux sous objets de document

Méthode setAttribute()

Si on veut changer dynamiquement la valeur d'un attribut d'une balise alors on fait appel à l'attribut **setAttribute**.

Exemple:

```
<div align="left">
    Contenu de la DIV.
</div>
<script language="javascript">
    document.getElementsByTagName("div").item(0).setAttribute("align","center");
</script>
```

Autres façons pour accéder aux sous objets de document

Méthode **getAttribute()**

La méthode **getAttribute** permet de récupérer la valeur de l'attribut passé en paramètre.

Exemple:

```
<video src="pub.ogv" id="publicite"></video>
<script language="javascript">
    alert(document.getElementById("publicite").getAttribute("src"));
</script>
```

Création, intégration et suppression des éléments

Méthode `createTextNode()`

La méthode `createTextNode` permet de créer un nœud texte.

Exemple:

```
message=document.createTextNode("Bonjour à tous.");
```

Méthode `appendChild()`

La méthode `appendChild` ajoute un enfant à l'objet sur lequel elle est appelée. L'enfant peut être un nœud élément ou un nœud texte.

```
<html>
  <head>
    <meta charset="UTF-8" />
    <style>
      .maclasse{
        width:200px;
        height:60px;
        background-color:#EE6600;
        margin:10px;
        font:14pt verdana;
        padding:10px;
        color:#FFFFFF;
      }
    </style>
    <script language="javascript">
      function ajouter(){
        baliseDIV=document.createElement("div");
        baliseDIV.className="maclasse";
        message=document.createTextNode("Bonjour à tous.");
        baliseDIV.appendChild(message);
        document.getElementById("mdiv").appendChild(baliseDIV);
      }
    </script>
  </head>
  <body>
    <input type="button" value="Ajouter une DIV" onClick="ajouter()" />
    <div id="mdiv"></div>
  </body>
</html>
```

Méthode `insertBefore()`

La méthode `insertBefore()` ajoute un nouveau nœud enfant dans l'élément parent sur lequel elle est appelé. Si `appendChild()` ajoute le nœud à la fin, `insertBefore` quant à elle permet de spécifier avant quel élément on veut insérer le notre.

```
<style>
    .maclasse{
        width:200px;
        height:60px;
        background-color:#EE6600;
        margin:10px;
        font:14pt verdana;
        padding:10px;
        color:#FFFFFF;
    }

</style>
<script language="javascript">
    function ajouter(){
        baliseDIV=document.createElement("div");
        baliseDIV.className="maclasse";
        message=document.createTextNode("Bonjour à tous.");
        baliseDIV.appendChild(message);
        document.getElementById("madiv").insertBefore(baliseDIV,document.getElementById("monimage"));
    }
</script>
</head>
<body onLoad="decoup()">
    <input type="button" value="Insérer une DIV" onClick="ajouter()" />
    <div id="madiv">
        
    </div>
</body>
```

Chapitre 3: Le langage Javascript

Objet DOM

Méthode removeChild()

La méthode **removeChild()** supprime l'enfant passé en paramètre du parent

Exemple:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <script language="javascript">
      function supprimer() {
        document.body.removeChild(document.getElementById("monimage"));
      }
    </script>
  </head>
  <body>
    <input type="button" value="Supprimer l'image" onClick="supprimer()" /><br />
    
  </body>
</html>
```

Chapitre 3: Le langage Javascript

Objet DOM

Mot-clé **this**

Le mot clé **this** fait référence à l'objet sur lequel il est déclaré. Il permet de simplifier la désignation des objets à travers le DOM. Au lieu de cibler un élément à travers son ID ou son tagName, **this** raccourcie visiblement la syntaxe.

Supposons que nous disposons d'une image en noir et blanc qui, une fois survolé, laisse place à l'image en couleur. On peut écrire le code comme ceci:

```
<script language="javascript">
    tabImages=new Array("images/footerlogonb.png","images/footerlogo.png");
    function changerImage(indice,image) {
        image.src=tabImages[indice];
    }
</script>

```

Le mot clé **this** fait référence à l'image, car c'est sur celle-ci qu'il est déclaré et passé en paramètre de la fonction **changerImage()**. Une autre astuce consiste à placer les chemins des images dans un tableau, ce qui rend le code plus lisible et facilement modifiable.

Chapitre 4: Le langage PHP

Langage PHP

1. PHP C'est quoi?
2. De quoi aura-t-on besoin?
3. Les bases du PHP
4. Les structures de contrôle
5. Les formulaires et fonctions agissant sur les variables
6. Fonctions mathématiques
7. Chaines de caractères
8. Les tableaux PHP
9. Les fonctions
10. Les sessions
11. Les fichiers
12. Accès aux bases de données

Chapitre 4: Le langage PHP

Un peu d'histoire

En 1994, et suite à un projet personnel, [Rasmus Lerdorf](#) a créé le langage **PHP** qui désignait à l'époque **Personnal Home Page**. Ce projet a été animé par son besoin de mettre à jour son CV en ligne au lieu de modifier le code source et réhéberger la page à chaque fois.

En 1997 **Zeev Suraski** et **Andi Gutmans** ont entamé des travaux d'amélioration du langage PHP. La première version officielle est alors nommée **PHP3** dont l'acronyme récursif désigne désormais **PHP Hypertext Preprocessor**. Peu de temps après, ils ont développé le moteur [Zend Engine](#) suite auquel, la version 4 de PHP est née.

En 2004, PHP5 a été créé. Il s'agit d'une version du langage qui supporte la Programmation Orientée Objet (POO) d'une manière très avancée. C'est cette version qui est actuellement utilisée par la plupart des développeurs PHP. Le successeur de PHP5 est PHP7 qui est finalisé en décembre 2015 au dépend de PHP6 qui n'a pas été officialisé.

Chapitre 4: Le langage PHP

Un site dynamique c'est quoi?

En général, on peut regrouper les sites Web en deux catégories: les sites statiques et les sites dynamiques.

Un site statique renferme un contenu figé qui ne change pas automatiquement et qui reste le même tant que le Webmaster n'est pas intervenu pour le modifier manuellement.

Les sites dynamiques, quant-à eux, sont des sites Web dont le contenu change d'une manière autonome. Celui ci peut changer en fonction de la date, le navigateur utilisé par le client, la position géographique de celui-ci, les privilèges attribués à chaque utilisateur suite à une authentification par exemple, l'historique de navigation etc...

Les sites dynamiques reposent sur des langages dits [CGI](#) (pour Common Gateway Interface) dont PHP fait parti.

Chapitre 4: Le langage PHP

Particularités de PHP

PHP est un langage de programmation coté serveur. PHP s'exécute entièrement sur le serveur qui héberge le site Web. Bien que son exécution est moins rapide que celle de Javascript en raison du temps que prend la requête pour parvenir au serveur et la réponse pour arriver jusqu'au navigateur, PHP dispose de plusieurs atouts qui se résument dans les points suivants:

- **Code source confidentiel:** Puisque PHP est un langage CGI qui s'exécute sur le serveur, alors son code source n'est jamais visible par le client, ce qui permet de manipuler des données confidentielles.
- **Open source:** PHP est un langage de programmation libre de droit.
- **Multi-plateforme:** PHP s'exécute sur des serveurs d'applications que l'on peut installer sur de nombreux systèmes d'exploitation (Unix/Linux, Windows, Mac OS, BSD...)
- **Syntaxe simple et intuitive:** Les personnes ayant déjà programmer en C ou en Javascript trouvent PHP facile à manipuler.

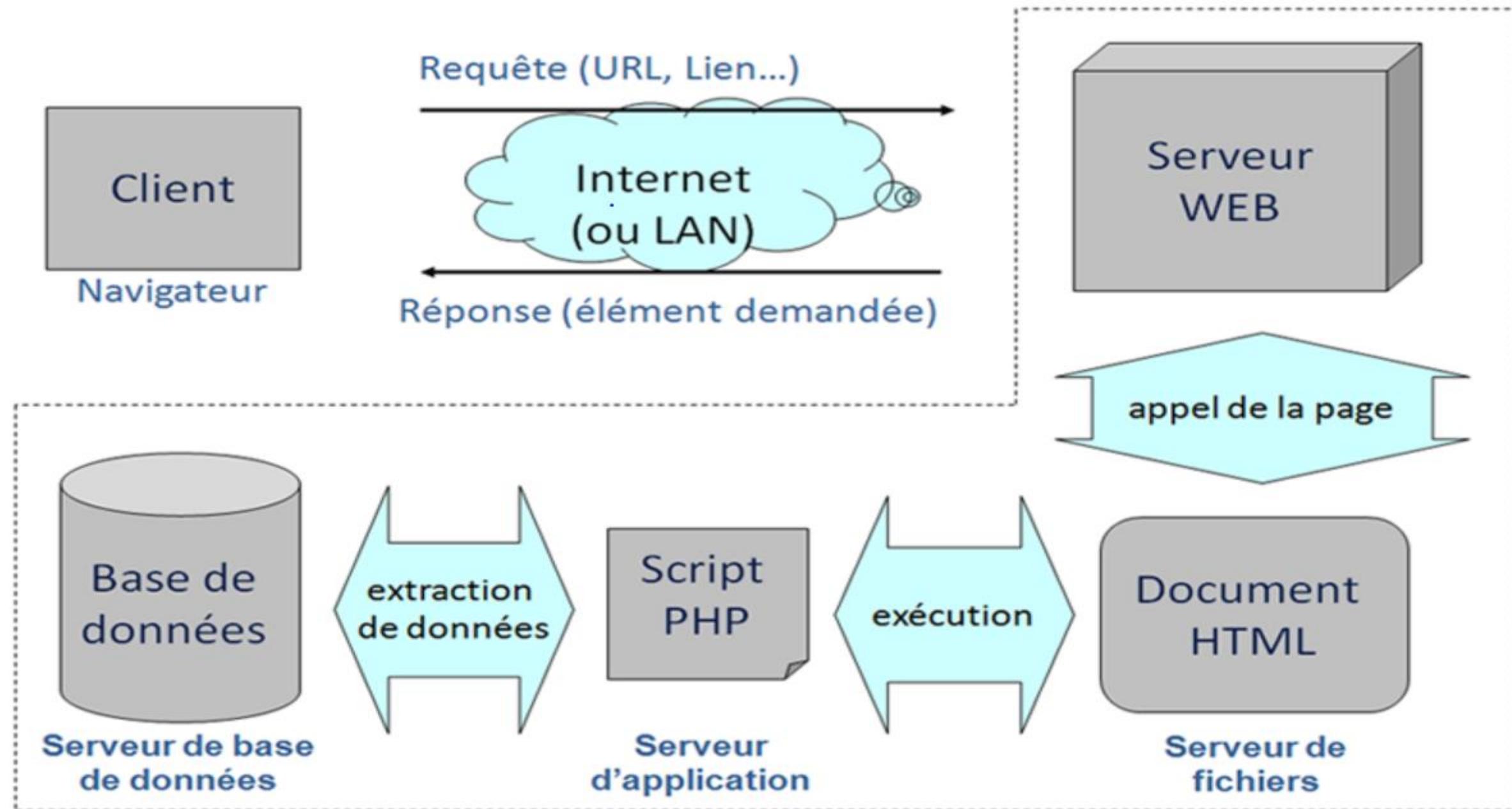
Chapitre 4: Le langage PHP

Particularités de PHP

- **Interfaçage simple avec les bases de données:** Bien que PHP est souvent associé au [SGBD MySQL](#), il peut cependant interagir avec presque tous les SGDB connus, d'autant plus à l'aide de l'interface [PDO](#).
- **Richesse fonctionnelle:** PHP prend en charge de nombreuses bibliothèques qui permettent de réussir des traitements de nature différente comme la manipulation du DOM, la programmation réseau ou le traitement d'images...
- **Supports disponibles:** La documentation de PHP est très abondante et diversifiée surtout grâce aux postes d'une très large communauté qui s'est formée autour de lui. Cependant le support PHP officiel est disponible sur le site [www.php.net](#). Si vous cherchez des informations concernant une fonction, il suffit de taper son nom après l'URL. Par exemple, pour obtenir des informations sur la fonction **echo** de PHP, il suffit de saisir l'URL suivante: [www.php.net/echo](#).

Chapitre 4: Le langage PHP

De quoi aura-t-on besoin pour coder en PHP?



Préparer son serveur

Désormais, il n'est plus question de mettre nos pages Web n'importe où sur notre ordinateur. En effet, il y aura un emplacement qui sera dédié à l'hébergement et qui sera géré par le logiciel faisant office de serveur Web.

Parlant justement de serveur Web, il faut d'abord savoir que ce serveur exécute les requêtes [HTTP](#) (HyperText Transfer Protocol) qui est le protocole de communication utilisé pour transférer les données hypertextes entre le client et le serveur.

Serveur Apache

[Apache](#) est un serveur Web libre distribué sous la [licence Apache](#). Il est le serveur Web le plus populaire et le plus utilisé par les hébergeurs à travers la toile.

Il existe plusieurs logiciels qui font office de serveur Web et qui supportent automatiquement le langage PHP. [Wamp](#)

Préparer son serveur

Une fois WAMP Server démarré, vous disposez désormais d'un serveur Web local à votre machine. Pour interroger ce serveur vous pouvez taper sur l'URL de votre navigateur préféré l'adresse `http://localhost` ou `http://127.0.0.1`. Ces deux adresses font référence au serveur local (ou boucle locale dans le jargon des réseaux informatiques). Normalement, vous devez obtenir la page propre au serveur qui contient des informations sur WAMP Server.



WampServer

Où faut-il placer nos documents PHP?

Si vous voulez exécuter du code PHP, alors il faut faire appel au serveur d'application grâce à l'URL **http://localhost**. Il est donc hors de question de cliquer sur un document PHP deux fois pour l'exécuter dans un navigateur comme on avait l'habitude de faire avec HTML ou Javascript.

Tout d'abord, il faut noter qu'un document PHP est toujours suffixé par l'extension **.php**. C'est de cette manière que le serveur Web sait qu'il doit faire exécuter le document PHP par le serveur d'application, qui renferme le moteur PHP, avant de le renvoyer au client. Cependant, il faut placer nos documents PHP (et les autres documents du site Web d'ailleurs) dans un endroit bien précis. Cet endroit s'appelle le **Document Root** (littéralement **la racine du serveur Web**).

Sur WAMP Server le Document Root est le dossier du nom de **www**, il est par défaut situé dans le dossier d'installation du logiciel (généralement **C:/wamp/www**). Il est donc important de placer tous les documents PHP que nous allons créer dans ce répertoire.

A quoi ressemble un document PHP?

Une page PHP est suffixée par l'extension **.php**, mais cela ne veut pas dire qu'elle contient uniquement du code PHP. En effet, elle peut renfermer toutes les syntaxes que nous avons vu jusqu'ici à savoir: HTML, CSS et Javascript.

Il est donc tout à fait possible que votre page PHP contienne 4 langages différents à la fois.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <style>
            * {
                /* Code CSS */
                font-family:verdana;
                font-size:10pt;
            }
        </style>
    </head>
    <body>
        <div>C'est du HTML.</div>
        <script language="javascript">
            document.write("C'est du Javascript.");
        </script><br />
        <?php
            echo "Et là, du PHP!";
        ?>
    </body>
</html>
```

Chapitre 4: Le langage PHP

Intégration du code PHP

Si un document suffixé par **.php** est demandé par le client, le serveur d'application l'analyse d'abord pour chercher les scripts PHP qu'ils contient et les exécute sur place. Le résultat obtenu est un document où le code PHP a été remplacé par le résultat de son l'exécution. Ce document est renvoyé au navigateur du client qui, lui, exécute les codes restants qui ne sont rien d'autre que des codes côté client comme HTML, CSS et Javascript.

Si on souhaite consulter le code source sur le navigateur du client on obtient ceci:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <style>
            * {
                /* Code CSS */
                font-family:verdana;
                font-size:10pt;
            }
        </style>
    </head>
    <body>
        <div>C'est du HTML.</div>
        <script language="javascript">
            document.write("C'est du Javascript.");
        </script><br />
        Et là, du PHP!
    </body>
</html>
```

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <style>
            * {
                /* Code CSS */
                font-family:verdana;
                font-size:10pt;
            }
        </style>
    </head>
    <body>
        <div>C'est du HTML.</div>
        <script language="javascript">
            document.write("C'est du Javascript.");
        </script><br />
        <?php
            echo "Et là, du PHP!";
        ?>
    </body>
</html>
```

Délimiteurs PHP

Pour indiquer au serveur où est intégré le code PHP dans la page, il faut le placer dans des délimiteurs (ou balises) spéciaux. Tout ce qui est déclaré entre ces délimiteurs sera exécuté par le serveur avant de renvoyer la page au client.

- **Délimiteur long:** il s'agit du délimiteur classique de PHP. Le code PHP doit être placé entre `<?php` et `?>`. C'est le plus populaire et le plus recommandé.

Les commentaires

- En PHP on peut intégrer des commentaires qui seront ignorés lors de l'exécution du script par le serveur. Comme pour Javascript (ou pour le langage C) les commentaires en PHP peuvent avoir deux formes:

Commentaire de fin de ligne: il s'agit d'un commentaire qui s'étend jusqu'à la fin de la ligne à partir du symbole double slash (//).

- **Commentaire sur plusieurs lignes:** il s'agit d'un bloc qui peut contenir plusieurs lignes comprises entre les symboles /* et */.

```
<?php
    // Commentaire de fin de ligne
    /*
        Bloc entier
        vu comme un
        commentaire
    */
?>
```

Variables PHP

Comme pour les autres langages de programmation, en PHP les variables servent à stocker des valeurs qui peuvent changer au cours du programme. A l'instar de Javascript, nous n'avons pas besoin de déclarer le type de la variable avant de s'en servir et on peut lui affecter des valeurs de types différents tout au long du programme sans recourir au CAST.

En PHP, les noms de variables sont prefixées par le symbole dollar (\$) et commencent par une lettre minuscule, majuscule ou le caractère souligné (_). Cependant, ils peuvent contenir des chiffres au milieu ou à la fin. Les espaces ne sont pas autorisés.

```
<?php
    $a=10;      // Juste
    $_a9=true;   // Juste
    $9a="Bonjour"; // Faux (le chiffre ne doit pas figurer au début)
    $a b=5.3;    // Faux (le nom de la variable ne doit pas contenir d'espaces)
?>
```

Variables scalaires

En PHP, il existe des variables scalaires et des variables tableau (que l'on va traiter plus loin dans ce cours). Les variables scalaires peuvent être:

- Des nombres entiers (nombres sans virgule positifs ou négatifs).
- Des nombres décimaux (de type double).
- Des chaînes de caractères (des suites de caractères quelconques).
- Des booléens (qui peuvent avoir les valeurs **true** ou **false**).

Si une variable appelée dans le script existe déjà et a déjà reçu une valeur, celle-ci sera utilisée. Sinon alors le moteur PHP lui affecte la valeur 0 par défaut.

```
<?php  
    $a=5;  
    $b=$a+1;    // $b vaut 6  
    $d=$c+1;    // $d vaut 1  
?>
```

Les constantes PHP

Les constantes servent aussi à stocker des valeurs dans un programme, mais à l'inverse des variables, leurs valeurs ne changent pas.

Fonction define()

Pour définir une constante on utilise la fonction **define(cte,val)**. **cte** représente l'identifiant de la constante à définir et **val** sa valeur.

```
<?php
    define(cte, "Bonjour");
    echo cte; // Affiche Bonjour
    define(cte, "Bonsoir");
    echo cte; // Affiche Bonjour
?>
```

Les opérateurs

Les opérateurs sont des symboles qui permettent de faire des opérations sur les variables. Les opérateurs sont souvent les mêmes dans la plupart des langages de programmation et ils sont représentés par des symboles similaires dans la plupart des cas.

En PHP on distingue 5 familles d'opérateurs:

- **Opérateurs arithmétiques.**
- **Opérateurs d'incrémentation.**
- **Opérateurs assignement.**
- **Opérateurs de comparaison.**
- **Opérateurs logiques.**

Les structures conditionnelles et boucles

Structure if else

```
<?php  
    $a=1;  
    if ($a==1)  
        echo "Un";  
    elseif($a==2)  
        echo "Deux";  
    else  
        echo "Autre";  
?>
```

```
<?php  
    $a=1;  
    if ($a==1)  
        echo "Un";  
    else{  
        if ($a==2)  
            echo "Deux";  
        else  
            echo "Autre";  
    }  
?>
```

Structure switch case

```
<?php  
    $a=1;  
    switch($a) {  
        case 1 : echo "Un"; break;  
        case 2 : echo "Deux"; break;  
        default: echo "Autre";  
    }  
?>
```

Les structures conditionnelles et boucles

Boucle for

```
<?php  
    for($i=0;$i<10;$i++) {  
        echo "Bonjour<br />";  
    }  
?>
```

Structure while

```
<?php  
    $i=0;  
    while($i<10) {  
        echo "Bonjour<br />";  
        $i++;  
    }  
?>
```

Structure do while

```
<?php  
    $i=0;  
    do {  
        echo "Bonjour<br />";  
        $i++;  
    }  
    while($i<10);  
?>
```

Les structures conditionnelles et boucles

Mots-clés break et continue

```
<?php
    for($i=0;$i<10;$i++) {
        if($i==5)
            break;
        echo "Bonjour<br />";
    }
?>
```

```
<?php
    for($i=0;$i<10;$i++) {
        if($i==5)
            continue;
        echo "$i<br />";
    }
?>
```

0
1
2
3
4
6
7
8
9

Chapitre 4: Le langage PHP

Fonctions mathématiques

Fonctions mathématiques standards

- abs(\$x)**: retourne la valeur absolue de \$x.
- ceil(\$x)**: retourne l'arrondi supérieur de \$x. Il s'agit du nombre entier immédiatement supérieur ou égal à \$x.
- floor(\$x)**: retourne l'arrondi inférieur de \$x. Il s'agit du nombre entier immédiatement inférieur ou égal à \$x.
- round(\$x,\$i)**: retourne l'arrondi le plus proche de \$x avec la précision \$i. Le nombre retourné aura \$i chiffres après la virgule.
- pow(\$x,\$y)**: retourne \$x à la puissance \$y.
- max(\$a,\$b,\$c,...)**: retourne le nombre le plus grand parmi \$a, \$b, \$c...
- min(\$a,\$b,\$c,...)**: retourne le nombre le plus petit parmi \$a, \$b, \$c...
- log(\$x)**: retourne le logarithme naturel (népérien) de \$x.
- log10(\$x)**: retourne le logarithme en base 10 de \$x.
- exp(\$x)**: retourne l'exponentiel de \$x.
- sqrt(\$x)**: retourne la racine carré de \$x.
- hexdec(\$x)**: converti \$x de la base hexadécimale à la base décimale.
- dechex(\$x)**: converti \$x de la base décimale à la base hexadécimale.
- bindec(\$x)**: converti \$x de la base binaire à la base décimale.
- decbin(\$x)**: converti \$x de la base décimale à la base binaire.

Fonctions trigonométriques

- **pi()**: retourne la valeur approximative de PI (3,14159265359).
- **sin(\$x)**: retourne le sinus de \$x.
- **cos(\$x)**: retourne le cosinus de \$x.
- **tan(\$x)**: retourne la tangente de \$x.
- **asin(\$x)**: retourne l'arc sinus de \$x.
- **acos(\$x)**: retourne l'arc cosinus de \$x.
- **atan(\$x)**: retourne l'arc tangente de \$x.

Les fonctions qui traitent les nombres aléatoires

mt_rand(): La fonction mt_rand() permet de générer un nombre aléatoire entier compris entre 0 et RANDMAX (qui représente la valeur maximale pouvant être générée). Si on désigne des paramètres entre les parenthèses, par exemple mt_rand(1,4), alors l'un des nombres 1, 2, 3 ou 4 sera généré aléatoirement.

- **mt_srand()**: Cette fonction permet d'initialiser le moteur de génération des nombres aléatoires. Il est conseillé de toujours faire suivre mt_rand() par mt_srand() pour garantir un meilleur résultat aléatoire la prochaine fois.
- **mt_getrandmax()**: Cette fonction permet de retourner le RANDMAX qui représente la valeur maximale que le moteur de génération des nombres aléatoires peut générer. Sa valeur vaut 2147483647. Si la fonction mt_rand() est déclarée sans paramètres, alors elle peut générer un nombre compris entre 0 et RANDMAX.

Les chaînes de caractères en PHP

Concaténation de chaînes de caractères

```
$str1="Bonjour";  
$str2='Bonsoir';
```

```
<?php  
    $nbr=2.3;  
    echo "L'arrondi supérieur de $nbr est ".ceil($nbr);  
?>
```

Fonctions agissant sur les chaînes de caractères

- **strlen(\$str)**: retourne un entier qui représente le nombre de caractères que contient la chaîne \$str.
- **strtoupper(\$str)**: convertir la chaîne \$str en majuscule.
- **strtolower(\$str)**: convertir la chaîne \$str en minuscule.
- **ucfirst(\$str)**: convertit le premier caractère de la chaîne \$str en majuscules.

Fonctions agissant sur les chaînes de caractères

- **trim(\$str)**: supprime les espaces de début de fin de la chaîne \$str.
- **rtrim(\$str)** ou **chop(\$str)**: supprime les espaces de fin de la chaîne \$str.
- **substr(\$str,\$deb,\$nbr)**: extrait une partie de la chaîne de caractères en commençant de la position \$deb et en retournant \$nbr caractères (Notez que la position du premier caractère de la chaîne est 0).
- **ord(\$car)**: retourne le code ASCII du caractère \$car.
- **chr(\$int)**: retourne le caractère correspondant au code ASCII \$int.
- **addslashes(\$str)**: ajoute des antislashes avant les caractères spéciaux comme l'antislash, simple cote ou double cote qui se trouvent dans la chaîne de caractères \$str.
- **stripslashes(\$str)**: supprime les antislashes qui se trouvent dans la chaîne de caractères \$str.
- **strip_tags(\$str)**: supprime les balises HTML qui se trouvent dans la chaîne de caractères \$str.
- **htmlentities(\$str)**: convertit certains caractères de \$str en mot clés HTML.

Afficher les chaînes de caractères

```
<?php  
    print "Bonjour"; // Affiche: Bonjour  
?>
```

```
<?php  
    echo "Bonjour";  
?>
```

```
<?php  
    printf("Salut à %s","tous"); // Affiche: Salut à tous  
?>
```

Les variables tableau

Les tableaux en PHP sont des variables qui peuvent contenir plusieurs valeurs à la fois. Ils peuvent être indexés ou associatifs et peuvent aussi avoir plusieurs dimensions.

Tableaux indexés

Un tableau indexé contient des indices numériques qui indexent le contenu souhaité. Ces indexées commencent par défaut de 0 et s'incrémentent de 1 à chaque fois. Pour créer un tableau indexé en PHP il existe plusieurs méthodes:

```
$tab=array("PHP", "Coté serveur", 60);
```

```
$tab=array();  
$tab[]="PHP";  
$tab[]="Coté serveur";  
$tab[]=60;
```

```
$tab=array();  
$tab[0]="PHP";  
$tab[1]="Coté serveur";  
$tab[2]=60;
```

Tableaux associatifs

Si un tableau indexé contient des indices numériques qui indexent les entrées, un tableau associatif , quant-à lui, contient des clés. Ces clés sont des chaînes de caractères qui permettent d'avoir une idée plus claire sur le contenu indexé.

Par exemple, le dernier tableau créé contient trois valeurs "PHP", "Coté serveur" et 60. Mais on ne sait pas ce que représentent réellement ces trois entrées. Essayons maintenant de les déclarer au sein d'un tableau associatif.

```
$tab=array("langage" => "PHP" , "execution" => "Coté serveur" , "heures" => 60);
```

```
<?php  
    $tab=array();  
    $tab["langage"] = "PHP";  
    $tab["execution"] = "Coté serveur";  
    $tab["heures"] = 60;  
?>
```

Tableaux à plusieurs dimensions

Les tableaux qu'on a vu précédemment contiennent une seule dimensions.

Il est toute fois possible de créer des tableaux à plusieurs dimensions, soit directement ou en déclarant les tableaux les uns dans les autres.

Pour simplifier on va voir l'exemple d'un tableau indexé à deux dimensions.

```
$tab1 = array("A" , "B" , "C");
$tab2 = array("D" , "E" , "F");
$tab3 = array("G" , "H" , "I");
```

```
$tab=array($tab1 , $tab2 , $tab3);
```

```
echo $tab[0][0]; // Affiche: A
echo $tab[1][2]; // Affiche: F
```

Parcourir un tableau: structure foreach

Pour parcourir un tableau, la solution la plus classique consiste à utiliser la boucle **for** ou la boucle **while**. Cependant, PHP inclue une structure de contrôle qui s'applique spécialement aux tableaux. Il s'agit de la structure **foreach**.

La structure **foreach** permet de parcourir un tableau élément par élément..

```
<?php
    $tab=array("PHP","Coté serveur",60);
    foreach($tab as $elem) {
        echo "$elem <br />";
    }
?>
```

PHP
Coté serveur
60

```
<?php
    $tab=array(
        "langage"=>"PHP",
        "execution"=>"Coté serveur",
        "heures"=>60
    );
    foreach($tab as $cle => $elem) {
        echo "$cle: $elem <br />";
    }
?>
```

langage: PHP
execution: Coté serveur
heures: 60

Fonctions agissant sur les tableaux

- **count(\$tab)** ou **sizeof(\$tab)**: retourne un entier qui indique le nombre d'entrées du tableau.
- **in_array(\$var,\$tab)**: vérifie si la variable \$var existe dans le tableau. Si oui la fonction **in_array()** retourne true sinon elle retourne false.
- **list(\$var1,\$var2,\$var3...)**: affecte chacune des entrées du tableau respectivement au variables \$var1, \$var2, \$var3...
- **shuffle(\$tab)**: mélange le contenu du tableau en changeant l'indexe des entrées aléatoirement.
- **sort(\$tab)**: trie dans l'ordre alphanumérique les éléments du tableau.
- **rsort(\$tab)**: trie dans l'ordre alphanumérique inverse les éléments du tableau.
- **array_rand(\$tab)**: retourne l'indexe de l'une des entrée du tableau aléatoirement.
- **array_merge(\$tab1,\$tab2,\$tab3...)**: retourne un seul grand tableau qui contient les éléments des tableaux \$tab1, \$tab2, \$tab3...
- **implode(\$sep,\$tab)** ou **join(\$sep,\$tab)**: retourne une chaîne de caractères constituée des éléments du tableau séparés par le contenu de la variable \$sep.
- **explode(\$occ,\$str)**: cette fonction s'applique sur les chaînes de caractères. Elle crée un tableau en éclatant la chaîne \$str au niveau des occurrences \$occ.

Récupération des champs de formulaire

```
<form method="post" action="">
    <input type="text" name="prenom" /><br />
    <input type="submit" name="valider" value="Vérifier" />
</form>
```

Bien entendu, c'est ce qu'il y a de plus simple en HTML. Il s'agit d'un formulaire qui contient deux champs, une zone de texte au nom de **prenom** et un bouton d'envoi du nom de **valider**. Le formulaire utilise la méthode **POST** et envoie ses valeurs à la page courante une fois le bouton d'envoi actionné.

Sur la page qui est sensée traiter le formulaire (la page courante dans ce cas), on doit d'abord récupérer les valeurs postées par celui-ci. Pour ce faire, on a recourt aux variables (tableaux) superglobales **\$_POST** et **\$_GET**.

Chapitre 4: Le langage PHP

Formulaires et fonctions agissant sur les variables

Variables `$_POST` et `$_GET`

La variable `$_POST` est en réalité un tableau associatif, c'est à dire un tableau qui utilise des clés au lieu d'indexes.

La variable `$_POST` contient la valeur du champ de formulaire dont le nom est passé en tant que clé. Par exemple, pour récupérer la valeur que l'internaute a saisi dans la zone de texte nommée `prenom`, on fait appel à la variable `$_POST["prenom"]`.

La variable `$_POST` est appelée si le formulaire en question utilise la méthode `POST`. Si le formulaire utilise la méthode `GET` alors on appelle la variable `$_GET`. Tout comme le tableau `$_POST`, le tableau `$_GET` utilise comme clé, les noms des champs de formulaires.

`$_POST` et `$_GET` sont des variables **superglobales**, c'est à dire qu'elles sont reconnues dans n'importe quel contexte (à l'intérieur des fonctions comme à l'extérieur, voir même à l'intérieur des méthodes d'une classe dans le cas de PHP5 par exemple).

Fonctions agissant sur les variables

- **empty()**: permet de vérifier si la variable passée en paramètre est vide ou non. Si la variable est vide (elle ne contient aucune valeur) elle retourne la valeur **true**, si non elle retourne **false**.
- **isset()**: permet de vérifier si la variable passée en paramètre existe ou non. Si la variable est déjà évoquée avant la fonction **isset()**, alors elle existe et cette dernière retourne **true**, sinon elle retourne **false**.
- **unset()**: permet de supprimer la variable passée en paramètre. Si après **unset()** on appelle la fonction **isset()** en leur passant la même variable, alors celle-ci retournera **false**.
- **gettype()**: permet de retourner le type de la variable passée en paramètre.
- **is_numeric()**: vérifie si la variable passée en paramètre ne contient qu'une suite de caractères numériques. Si oui elle retourne **true** sinon elle retourne **false**.
- **is_int()**: vérifie si la variable passée en paramètre est de type entier ou non. Si oui, elle retourne **true** sinon elle retourne **false**
- **is_float()**, **is_long()**, **is_double()**, **is_string()** et **is_bool()** qui vérifient chacune le type de variable associée de la même manière que **is_int()**.

```
<?php  
    @$prenom=$_POST["prenom"];  
    @$valider=$_POST["valider"];  
    $message="";  
    if(isset($valider)) { la condition if(isset($valider)) signifie si la variable $valider associée au bouton d'envoi existe.  
        if(empty($prenom))  
            $message='<font color="#FF0000">Prénom invalide.</font>';  
        else  
            $message='<font color="#00FF00">Prénom valide.</font>';  
    }  
?>  
<!DOCTYPE html>          En effet, avant de poster le formulaire, aucune des variables $prenom et $valider  
<html>                    n'existe encore, car les variables $_POST qu'on leur a affecté ne seront créés qu'une  
    <head>                  fois le formulaire (utilisant la méthode POST) est envoyé en cliquant le bouton submit.  
        <meta charset="UTF-8" />  
    </head>  
    <body>  
        <form method="post" action="">  
            <input type="text" name="prenom" /><br />  
            <input type="submit" name="valider" value="Vérifier" />  
        </form>  
        <?php  
            echo $message;  
        ?>  
    </body>  
</html>
```

Déclaration des variables

En effet, avant de poster le formulaire, aucune des variables \$prenom et \$valider n'existe encore, car les variables \$_POST qu'on leur a affecté ne seront créés qu'une fois le formulaire (utilisant la méthode POST) est envoyé en cliquant le bouton submit.

Aucune des deux variables (tableaux) \$_POST n'existe avant de poster le formulaire. Dans ce cas, au premier chargement de la page, les clés "prenom" et "valider" que contiennent les crochés ne seront pas reconnus par le compilateur PHP qui générera des erreurs de notification disant "Undefined index prenom" et "Undefined index valider".

Créer ses propres fonctions en PHP

Création et appel d'une fonction

Pour créer une fonction on fait appel au mot clé **function** comme ceci:

```
function nomDeLaFonction($arg1,$arg2...){  
    CorpsDeLaFonction;  
    return $valeurDeRetour;  
}
```

```
<?php  
    function daj () {  
        echo date("d/m/Y");  
    }  
?>
```

```
<?php  
    daj ();  
?>
```

Portés des variables

Par défaut, une variable déclarée dans une fonction PHP est **locale** à cette fonction.

Elle n'est donc pas reconnue à l'extérieur de celle ci.

De même, une variable déclarée en dehors de la fonction n'est pas par défaut accessible depuis l'intérieur.

```
<?php
    $a="Salut";
    function f() {
        $b=" à tous";
        echo $a;
    }
    f();
    echo $b;
?>
```

La variable \$a déclarée à l'extérieur de la fonction n'est pas reconnue à l'intérieur de celle-ci.
Au moment de son appel, la fonction f() essaiera d'afficher la variable locale \$a qui n'existe pas.
La variable \$b initialisée à l'intérieur de la fonction est locale à celle-ci également.
Alors rien ne s'affichera avec echo \$b;.

Pour que les variables soient accessibles de partout on va faire appel à la variable \$GLOBALS et le mot clé global.

Les inclusions

Il se trouve qu'un bout de code figure dans plusieurs documents constituant notre projet Web (site ou application). Il s'agit souvent de codes qui assurent la connexion à la base de données ou à définir des éléments qui figurent systématiquement sur toutes nos pages, comme les entêtes, bas de pages et menu de navigation.

```
<?php  
    $pdo=new PDO (  
        "mysql:host=localhost; dbname=mabase",  
        "user",  
        "pass"  
    );  
?>
```

Le fait de déclarer le même code dans plusieurs pages rend celui-ci difficile à maintenir. En effet, imaginons que nous voulons nous connecter à une base de données du nom de "mabase" installée sur le serveur local et à laquelle on peut accéder avec l'utilisateur "user" et le mot de passe "pass". Si on veut se connecter à la base de données en utilisant [PDO](#)(PHP Data Object), la syntaxe ressemblerait à ceci:

Structure include

La structure **include** permet d'appeler un fichier dans la page où elle a été déclarée. On la prend à tort pour une fonction, mais en réalité il s'agit d'une structure et par conséquent, les parenthèses ne sont pas obligatoire.

La syntaxe de la structure **include** ressemble à ceci:

```
<?php  
    include "fichier_à_inclure";  
    // ou bien  
    include ("fichier_à_inclure");  
?>
```

Structure require

La structure **require** fonctionne pratiquement comme **include**. Il s'agit aussi d'une structure mais on peut la déclarer avec des parenthèses. La principale différence entre les deux structures c'est qu'à l'inverse de **include** qui se contente d'afficher une notification si le fichier appelé n'est pas accessible, **require** quant-à-elle arrête nettement l'exécution du programme.

Les arrêts prématurés

Dans le cas normal, un programme s'exécute du début à la fin. L'arrêt de l'exécution après avoir passé en vue toutes les instructions est donc tout à fait logique. Cependant, il se peut qu'un programme s'arrête avant d'atteindre la fin prévue et cela est du principalement à l'une des raisons suivantes:

- **Erreur fatale:** il s'agit d'une erreur stricte qui empêche le programme de se poursuivre. Néanmoins, ce genre d'erreur pousse le compilateur à ne rien exécuter du tout car le langage PHP est compilé.
- **Arrêt prématué:** il s'agit d'un arrêt programmé par le développeur. Dans ce cas, même si le programme ne s'exécute pas en entier, son arrêt est considéré comme normal car il a été prévu.

Arrêt prématué avec `exit()` et `die()`

Les fonctions `exit()` et `die()` sont similaires (des alias). Elles arrêtent le programme à l'endroit où elle sont déclarées en affichant le message passé en argument (en tant que chaîne de caractères).

Arrêt prématué avec `exit()` et `die()`

Les fonctions `exit()` et `die()` sont similaires (des alias). Elles arrêtent le programme à l'endroit où elles sont déclarées en affichant le message passé en argument (en tant que chaîne de caractères).

```
<?php
    for ($i=1;$i<=10;$i++) {
        if ($i>5)
            die("Fin");
        echo "Ligne $i <br />";
    }
?>
```

Ce qui donne:

Ligne 1
Ligne 2
Ligne 3
Ligne 4
Ligne 5
Fin

Principe des sessions

Les sessions constituent un moyen de stocker les données sur le serveur. En plus, ces données sont propres à chaque utilisateur qui se connecte à celui-ci et sont associées à un identifiant de session unique. Ils sont très utiles car ils permettent de faire persister des informations aussi longtemps que le client est connecté voir même après qu'il ait quitté le site Web. Ces informations sont accessibles par toutes les pages visitées par le client (d'où le nom de session).

Quand une session est créée sur le serveur, ce dernier envoie son identifiant (unique) au client sous forme d'un cookie.

Démarrer ou reprendre une session: session_start()

```
<?php
    session_start();
    @$login=$_POST["login"];
    @$pass=$_POST["pass"];
    @$valider=$_POST["valider"];
    $bonLogin="user";
    $bonPass="1234";
    $erreur="";
    if(isset($valider)) {
        if($login==$bonLogin && $pass==$bonPass) {
            $_SESSION["autoriser"]="oui";
            header("location:session.php");
        }
    else
        $erreur="Mauvais login ou mot de passe!";
    }
?>
```

1. On lance une session avec session_start().
2. On récupère le contenu saisi dans le formulaire d'authentification.
3. Une fois, l'internaute clique sur le bouton, le test du mot de passe et le login se lance.
4. Si le login et le pass sont conformes, avec header, on dirige notre internaute vers sa page web.
Si non un message d'erreur sera affiché.

Reprendre une session: `session_start()`

```
<?php
    session_start();
    if($_SESSION["autoriser"] != "oui") {
        header("location:login.html");
        exit();
    }
?>
```

Détruire une sessions: `session_destroy()`

```
<?php
    session_start();
    session_destroy();
    header("location:login.html");
?>
```

Dans chaque page qui constitue le compte d'un internaute, on reprend la session avec `session_start()`.

Si la variable autoriser déclarer le tableau `$_SESSION` ne correspond pas à oui. Donc les données récupérées du formulaire ne sont pas conformes. L'internaute avec header sera redirigé vers la première page d'authentification.

Cette page contiendra un bouton ou un lien vers `deconnexion.php`.

Au niveau de la page `deconnexion.php`; la session doit être récupéré et détruite puis l'internaute sera redirigé vers la première page d'authentification.

Manipulation des fichiers en PHP

Quand on parle de fichiers en PHP on sous entend les fichiers texte. Ces fichiers peuvent contenir des données que l'on peut afficher sur une page Web ou peuvent servir d'espace de stockage pour y enregistrer le résultat de l'exécution de nos scripts PHP.

Fonctions de test et d'évaluation de fichiers

- **file_exists(\$fichier)**: vérifie si le fichier \$fichier existe ou non. S'il existe elle retourne **true** sinon **false**.
- **filesize(\$fichier)**: retourne la taille du fichier \$fichier en octets.
- **filetype(\$fichier)**: retourne le type du fichier \$fichier.
- **is_executable(\$fichier)**: vérifie si le fichier \$fichier est exécutable ou non. Il retourne un résultat booléen.
- **is_file(\$fichier)**: vérifie si le \$fichier est un fichier et non pas un répertoire ou un lien symbolique. Il retourne un résultat booléen.
- **is_link(\$fichier)**: vérifie si \$fichier est un lien symbolique. Il retourne un résultat booléen.
- **unlink(\$fichier)**: détruit le fichier \$fichier en le supprimant de son emplacement.
- **copy(\$src,\$dst)**: copie le fichier \$src dans \$dst. \$src et \$dst peuvent être des chemins absolus ou relatifs.
- **rename(\$old,\$new)**: renomme le fichier \$old en \$new.

Fonctions d'accès aux fichiers

Si les fonctions précédentes permettent d'agir sur le fichier en un seul lot (évaluation ou lecture et écriture du contenu en entier). Les fonctions qui vont suivre quant-à-elles, agissent partiellement sur le fichier, ligne par ligne, voir lettre par lettre. Ce qui permet un meilleur contrôle de celui-ci.

Fonction fopen():

La fonction `fopen($fichier,$mode)` permet d'ouvrir le fichier spécifié par la variable `$fichier` en mode `$mode`. La fonction `fopen()` retourne un identifiant d'ouverture qui sera utile pour les fonctions qui vont suivre.

Les différents modes d'ouvertures possibles sont:

Mode d'ouverture	Signification
r	Lecture seule
w	Création et écriture seules
a	Création et écriture seules avec ajout du contenu (le pointeur sera placé à la fin du fichier)

- **Fonction fclose()**

La fonction `fclose($fp)` ferme le fichier identifié par `$fp`. `$fp` est l'identifiant de l'ouverture du fichier retourné par la fonction `fopen()`.

- **Fonction file()**

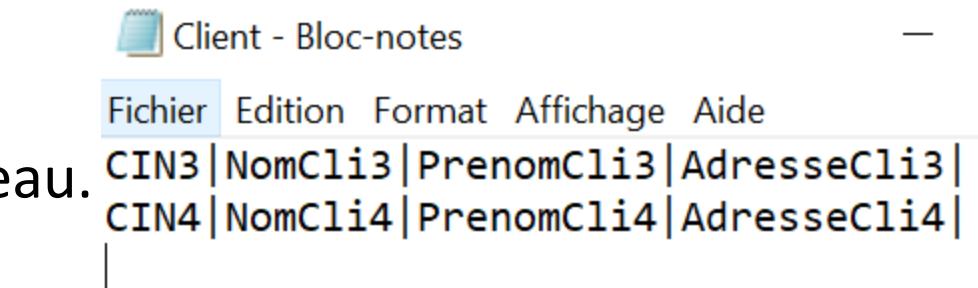
Lit le fichier et retourne son contenu sous format tableau.

- **Fonction explode()**

explode() retourne un tableau de chaînes de caractères, chacune d'elle étant une sous-chaîne du paramètre string extraite en utilisant le séparateur delimiteur.

□ Lire des données

```
$lignes=file("Client.txt");
if(file_exists($file))
{
    foreach($lignes as $ligne)
        $split=explode(' | ', $ligne);
    echo split[3];
}
```



\$lignes résultat de file contiendra les deux lignes du fichier.

Pour chaque ligne, on applique \$split pour récupérer chaque données à la fois en fonction du délimiteur mentionné.

Ici split[0] va contenir le CIN ; split[1] les noms; split[2] les prénoms et split[3] les adresse.

- **Fonction fputs()**

La fonction fputs(\$fp,\$chaine) permet d'écrire dans le fichier la chaîne de caractère \$chaine à partir de la position actuelle du pointeur. S'il y a déjà du contenu à l'endroit où l'on souhaite écrire, alors il sera écrasé lettre par lettre jusqu'à ce que l'écriture de \$chaine finisse.

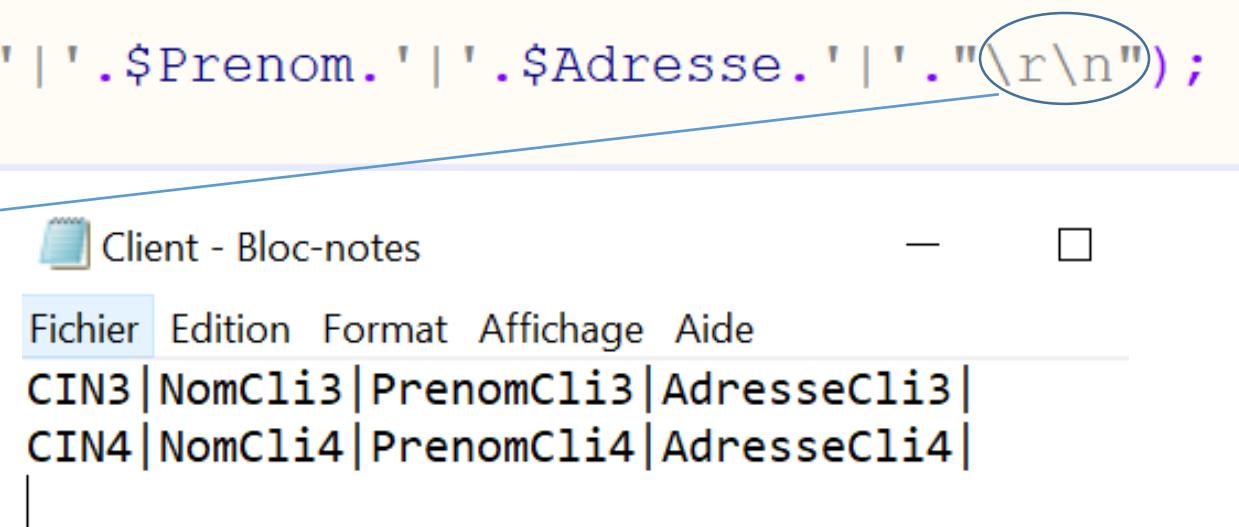
□ Ecrire des données

```
$file=fopen('Client.txt','a+');
if ($file!=null)
{
    fputs($file,$CIN.'|'.$Nom.'|'.$Prenom.'|'.$Adresse.'|'."
\r\n");
}
```

Ouverture du fichier en mode ajout

Ecriture des données avec le délimiteur |

Assurer le retour à la ligne pour chaque enregistrement.



□ Stockage de données

La **base de données** (BDD) est un système qui enregistre des informations. Ce qui est très important ici, c'est que ces informations sont toujours **classées**.

Les **SGBD (Système de Gestion des Bases de Données)** sont les programmes qui se chargent du stockage des données. Les plus connus sont, pour rappel :

- **MySQL** : libre et gratuit, c'est probablement le SGBD le plus connu. Nous l'utiliserons dans cette partie ;
- **PostgreSQL** : libre et gratuit comme MySQL, avec plus de fonctionnalités mais un peu moins connu ;
- **SQLite** : libre et gratuit, très léger mais très limité en fonctionnalités ;
- **Oracle** : utilisé par les très grosses entreprises ; sans aucun doute un des SGBD les plus complets, mais il n'est pas libre et on le paie le plus souvent très cher ;

□ Stockage de données

- les ordres au SGBD sont donnés en langage SQL.

Vous allez devoir communiquer avec le SGBD pour lui donner l'ordre de récupérer ou d'enregistrer des données. Pour lui « parler », on utilise le langage SQL.

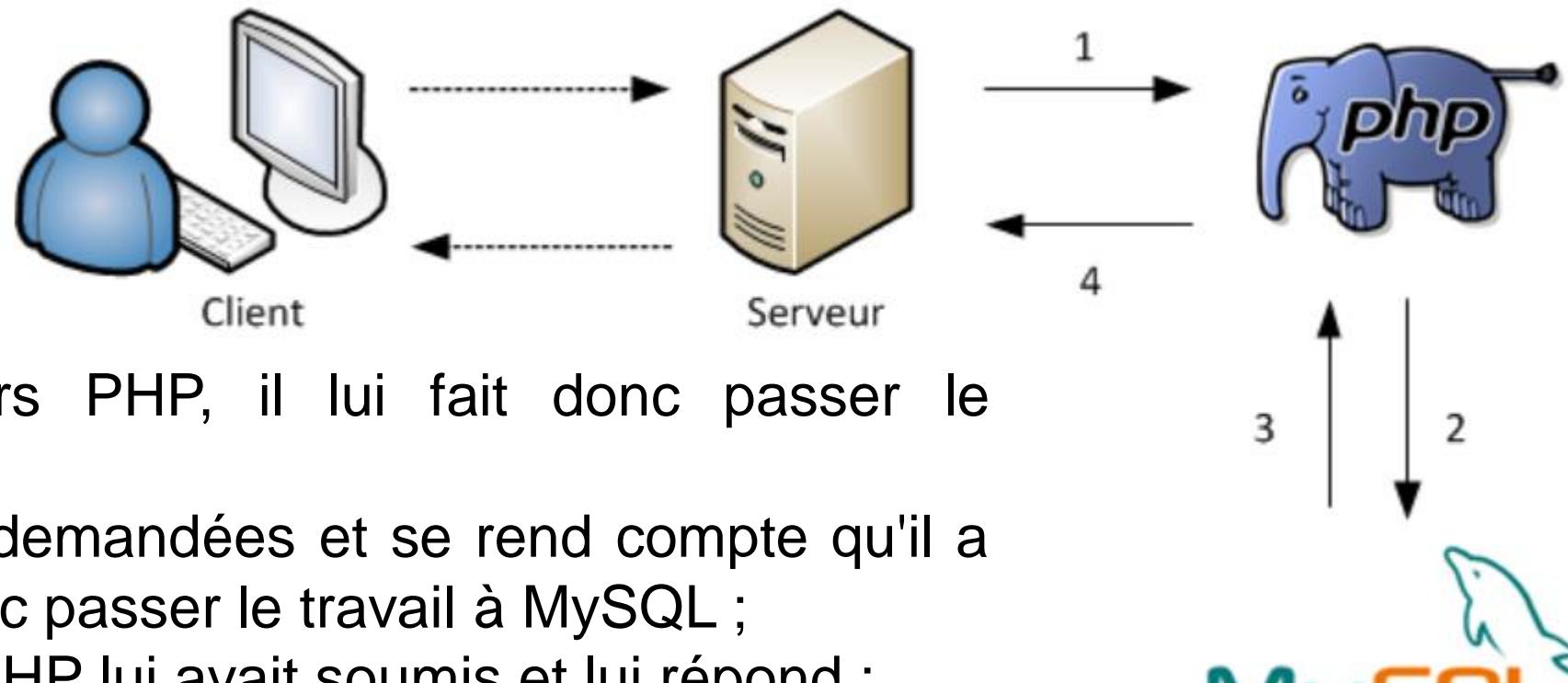
La bonne nouvelle, c'est que le langage SQL est un standard, c'est-à-dire que quel que soit le SGBD que vous utilisez, vous vous servirez du langage SQL. La mauvaise, c'est qu'il y a en fait quelques petites variantes d'un SGBD à l'autre, mais cela concerne généralement les commandes les plus avancées.

```
select numCompte from compte;
```

```
Select CIN from client;
```

□ Stockage de données

- Le PHP est l'intermédiaire entre vous et mysql



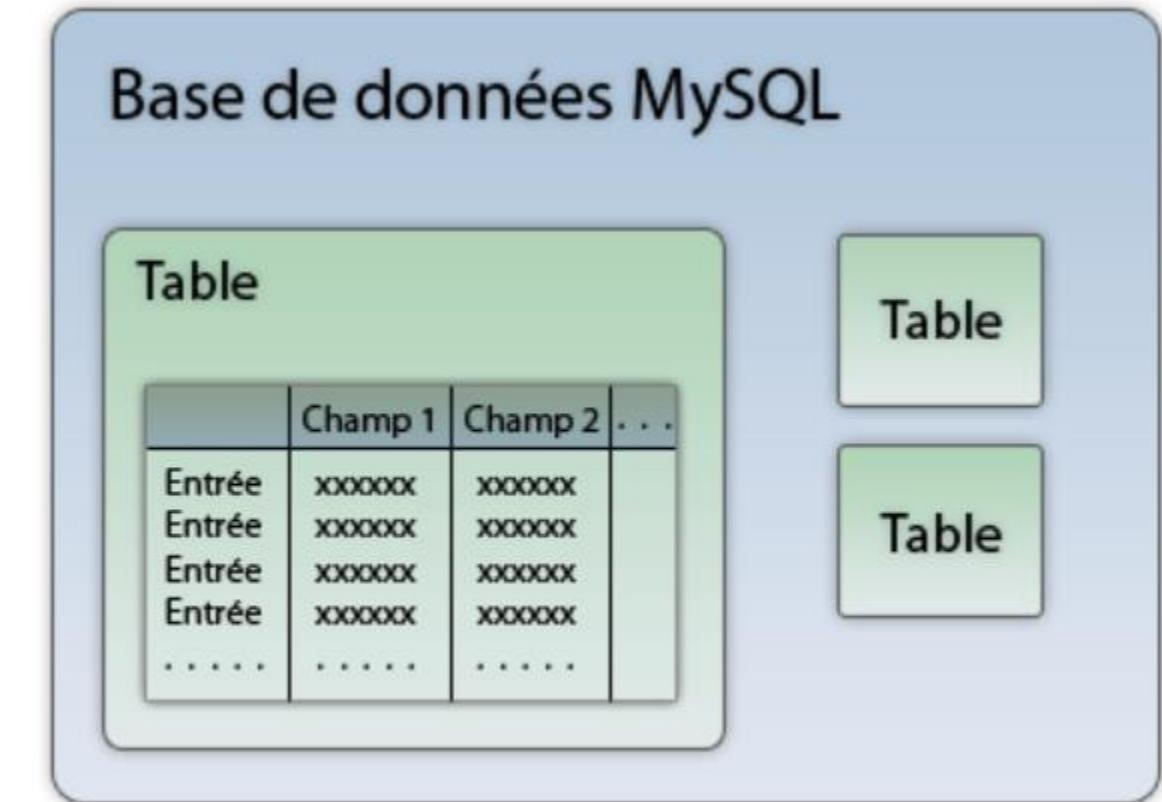
- 1.le serveur utilise toujours PHP, il lui fait donc passer le message ;
- 2.PHP effectue les actions demandées et se rend compte qu'il a besoin de MySQL. Il fait donc passer le travail à MySQL ;
- 3.MySQL fait le travail que PHP lui avait soumis et lui répond ;
- 4.PHP renvoie au serveur que MySQL a bien fait ce qui lui était demandé.

□ Stockage de données

- Structure d'une base de données

Une base de données est un ensemble de tables liées. Chaque table contient un ensemble de champs.

Au lieu maintenant de stocker nos enregistrements dans des fichiers nous allons directement les stockées dans les tables qui leurs correspondent.



Organisation d'une base de données MySQL

Chapitre 4: Le langage PHP

Les bases de données mysql

□ phpMyAdmin

The screenshot shows the phpMyAdmin interface running on a local server. The browser address bar displays `http://localhost/phpmyadmin/index.php`. The main menu bar includes options like Bases de données, SQL, État, Comptes utilisateurs, Exporter, Importer, Paramètres, RéPLICATION, and Plus. On the left, a sidebar lists databases: Nouvelle base de données, base_php, information_schema, ma, mabase, mysql, performance_schema, and sys. The central area contains two main sections: "Paramètres généraux" (General parameters) and "Paramètres d'affichage" (Display parameters). The "Paramètres généraux" section includes fields for "Modifier le mot de passe" and "Interclassement pour la connexion au serveur" set to "utf8mb4_unicode_ci". The "Paramètres d'affichage" section includes "Langue - Language" set to "Français - French", "Thème" set to "pmahomme", and "Taille du texte" set to "82%". To the right, a "Serveur de base de données" panel provides server status: MySQL (127.0.0.1 via TCP/IP), Type de serveur: MySQL, Connexion au serveur: SSL n'est pas utilisé, Version du serveur: 5.7.24 - MySQL Community Server (GPL), Version du protocole: 10, Utilisateur: root@localhost, and Jeu de caractères du serveur. A vertical status bar on the right, titled "WAMP SERVER 3.1.7", shows "Made in France by Otomatic" and a list of services: Localhost (4.8.4), phpMyAdmin (4.8.4), Adminer (4.7.0), Vos VirtualHosts, Répertoire www (2.4.37), Apache (2.4.37), PHP (7.2.14), MySQL (5.7.24), and MariaDB (10.3.12). It also includes links for Démarrer les services, Arrêter les services, and Redémarrer les services.

□ phpMyAdmin

1- Création d'une base de données

Base de données	Interclassement	Action
base_php	utf8_general_ci	Vérifier les privilèges
information_schema	utf8_general_ci	Vérifier les privilèges
mabase	utf8mb4_bin	Vérifier les privilèges
ma_base	utf8_bin	Vérifier les privilèges
ma_base_php	utf8_bin	Vérifier les privilèges
mysql	latin1_swedish_ci	Vérifier les privilèges
performance_schema	utf8_general_ci	Vérifier les privilèges
sys	utf8_general_ci	Vérifier les privilèges

Total: 8 latin1_swedish_ci

□ phpMyAdmin

1- Crédation des tables

The screenshot shows the phpMyAdmin interface for managing MySQL databases. On the left, a sidebar lists databases: Nouvelle base de données, base_php, information_schema, ma, mabase, mysql, performance_schema, and sys. The base_php database is selected, showing its structure with tables client and compte. The main area displays a table of existing tables:

Table	Action	Lignes	Type	Interclassement	Taille	Perte
client	Parcourir Structure Rechercher Insérer Vider Supprimer	2	MyISAM	utf8_general_ci	2,1 kio	-
compte	Parcourir Structure Rechercher Insérer Vider Supprimer	2	MyISAM	utf8_general_ci	3,1 kio	-
2 tables	Somme	4	MyISAM	utf8_general_ci	5,1 kio	0

Below the table, there are buttons for "Tout cocher" (Select all) and "Avec la sélection" (With selection). At the bottom, there are links for "Imprimer" (Print) and "Dictionnaire de données" (Dictionary of data), and a form for creating a new table:

Nouvelle table

Nom : Nombre de colonnes :

Exécuter

□ phpMyAdmin

1- Création des tables

Nom de table: Ajouter colonne(s)

Structure							
Nom	Type	Taille/Valeurs*	Valeur par défaut	Interclassement	Attributs	Null	Index
	INT		Aucun(e)				
	INT		Aucun(e)				
	INT		Aucun(e)				
	INT		Aucun(e)				

Commentaires de table :

Interclassement :

Moteur de stockage :

Définition de PARTITION :

Partitionner par : ()

Partitions :

□ phpMyAdmin

Insertion/modification/suppression d'enregistrements

phpMyAdmin

Serveur courant : MySQL

Récentes Préférées

- Nouvelle base de données
- base_php
 - Nouvelle table
 - client
 - compte
- information_schema
- ma
- mabase
- mysql
- performance_schema

Serveur: MySQL:3306 » Base de données: base_php » Table: client

Parcourir Structure SQL Rechercher Insérer Exporter

Affichage des lignes 0 - 1 (total de 2, traitement en 0,0000 seconde(s).)

```
SELECT * FROM `client`
```

Tout afficher Nombre de lignes : 25 Filtrer les lignes : Chercher dans

+ Options

nom	prenom	adresse	CIN
Client1	Client1	Adresse	EE123456

Éditer Copier Supprimer Tout cocher Avec la sélection : Éditer Copier Supprimer

Chapitre 4: Le langage PHP

Les bases de données mysql

□ **phpMyAdmin**

Ordre SQL

The screenshot shows the phpMyAdmin interface for managing a MySQL database named 'base_php'. The current table is 'client'. The SQL tab is selected, displaying the following code:

```
1 | DELETE FROM `client` WHERE 0
```

To the right of the SQL input field, there is a 'Colonnes' (Columns) panel listing the columns of the 'client' table:

- nom
- prenom
- adresse
- CIN

Below the SQL input field, there are several buttons: SELECT*, SELECT, INSERT, UPDATE, DELETE, Effacer (Delete), and Format. There is also a link to 'Récupérer la requête auto-sauvegardée' (Recover auto-saved query). A checkbox for 'Lier les paramètres' (Bind parameters) is present. At the bottom, there are options for 'Délimiteur' (Delimiter), checkboxes for 'Afficher à nouveau la requête après exécution' (Show query again after execution), 'Conserver la boîte de requêtes' (Keep query box), 'ROLLBACK à la fin' (Rollback at end), 'Activer la vérification des clés étrangères' (Enable foreign key check), and two buttons: 'Simuler la requête' (Simulate query) and 'Exécuter' (Execute).

Etablir la connexion entre PHP et la BDD

L'extension PDO : c'est un outil complet qui permet d'accéder à n'importe quel type de base de données. On peut donc l'utiliser pour se connecter aussi bien à MySQL que PostgreSQL ou Oracle.

Le gros avantage de PDO est que vous pouvez l'utiliser de la même manière pour vous connecter à n'importe quel autre type de base de données (PostgreSQL, Oracle...).

□ Etablir la connexion entre PHP et la BDD

```
try{
$bdd= new PDO('mysql:host=localhost; dbname=base_php; charset=utf8','root','');
}
catch(Exception $e)
{
    die('erreur: '.$e->getMessage());
}
```

La ligne de code crée un objet \$bdd. C'est un objet qui représente la connexion à la base de données. On crée la connexion en indiquant dans l'ordre les paramètres suivants:

- le nom d'hôte (localhost) ;
- la base de données (base_php) ;
- le login (root) ;
- le mot de passe (sous WAMP il n'y a pas de mot de passe, j'ai donc mis une chaîne vide, sous MAMP le mot de passe est root).

□ Récupérer des données

```
$query= $bdd->query("Select CIN from client;");  
while($data=$query->fetch())  
{  
    echo $data['CIN'];  
}  
  
$query->closeCursor();
```

- La méthode query est appelé via l'objet \$bdd récupéré comme résultat de la connexion via PDO (diapo précédente).
- Cette méthode permet d'interroger votre base de données en utilisant la requête SQL spécifiée.
- La méthode fetch permet de récupérer le résultat ligne par ligne.
- \$data est un tableau associatif qui contient tous les enregistrements obtenus via la requête SQL lancée.
- Ces cases disposent du même nom spécifié dans votre table.
- Ici \$data est un tableau à une colonne nommée CIN.

□ Récupérer des données

```
$q1=$bdd->query("select nom, prenom from client, compte where client.CIN= compte.CIN and compte.montant<0");  
  
while($d1=$q1->fetch())  
{  
    echo "  
        <table bgcolor='#FFF0F5' border=0>  
        <tr>  
        <td width='170px'>Nom: ".$d1['nom']."</td>  
        <td width='153px'>Prénom: ".$d1['prenom']."</td>  
        </tr> </table>";  
}  
  
$q1->closeCursor();
```

Le code suivant permet de sélectionner le nom, le prénom des clients dont le montant dans leurs comptes bancaires est inférieur à 0. (des débiteurs).

Les résultats obtenus sont affichés sous format d'un tableau html.

\$d1 contient deux colonnes nom et prénom et l'ensemble des lignes résultats de la requête.

Chaque ligne de \$d1 sera affichée au niveau d'un tr et deux td du tableau html.

□ Ecrire des données en utilisant des marqueurs

```
$query=$bdd->prepare("insert into client(nom,prenom,adresse,CIN) values(?, ?, ?, ?);");
$query->execute(array($nom,$prenom,$adresse,$CIN));
$query->closeCursor();
```

Les marqueurs nous permettent de signaler à la requête à exécuter qu'ils seront remplacés par la valeur des variables passées au niveau de la méthode execute.

C bout de code permet d'insérer un nouveau client dont les informations sont stockées dans les variables php \$nom, \$prenom, \$adresse et \$CIN. Ces données peuvent être récupérées via \$_POST au niveau d'un formulaire.

□ Modifier des données en utilisant des marqueurs

```
$query= $bdd->prepare("Update compte set montant = ? where numCompte=?;");  
$query->execute(array($m,$numC));  
  
$query->closeCursor();
```

Ce bout de code permet de modifier le montant du compte dont le numéro est stocké dans \$numC.

La nouvelle valeur du montant est stockée dans \$m.

□ Supprimer des données en utilisant des marqueurs

```
$q1=$bdd->prepare("delete from compte where CIN=?;");  
$q1->execute(array($CIN));  
  
$q1->closeCursor();
```

Ce bout de code permet de supprimer le compte du client dont le CIN est \$CIN.