# Développement d'applications Web

PHP & MYSQL

# Plan

- Introduction  aux World Wide Web
- Partie1: langage de programmation pour le web coté client
- Partie2: langage de programmation coté serveur(PHP)

# Introduction aux World Wide Web

# Qu'est-ce que le Web (ou World Wide Web, Toile, WWW, W3) ?

- Système hypertexte public : système contenant des documents liés entre eux par des hyperliens permettant de passer automatiquement d'un document à l'autre.
- Le concept du Web a été mis au point au CERN (Centre Européen de Recherche Nucléaire) en 1991 par une équipe de chercheurs à laquelle appartenaient Tim-Berners LEE, le créateur du concept d'hyperlien, considéré aujourd'hui comme le père fondateur du Web.

# Principe du web

- Le principe de web repose sur l'utilisation d'hyperliens pour naviguer entre des documents (appelés «**pages web**») grâce à un logiciel appelé **navigateur** (parfois également appelé *fureteur* ou *butineur* ou en anglais *browser*).
- Une page web est ainsi un simple fichier texte écrit dans un langage de description (appelé HTML), permettant de décrire la mise en page du document et d'inclure des éléments graphiques ou bien des liens vers d'autres documents à l'aide de balises.

# Différence entre le Web et Internet ?

- Internet : réseau mondial d'ordinateurs permettant aux utilisateurs de communiquer (courrier électronique), de publier des informations (Web), de transférer des données (FTP), de travailler à distance (telnet et ssh). . .
- Web : un aspect d'Internet, une partie des contenus circulant sur l'Internet

# Architecture client-serveur du Web

- Le client (navigateur : Internet Explorer, Firefox, Safari. . .)

  - demande au serveur des informations

  - affiche des pages pour l'utilisateur

- Le serveur (Apache, Microsoft IIS. . .)

  -reçoit en permanence les requêtes des clients

  -renvoie les documents correspondants

Le serveur Web : soit le logiciel qui répond aux requêtes d'un client, soit la machine sur lequel fonctionne ce serveur.

# Qu'est-ce qu'un site web ?

- Un site web (aussi appelé site internet ou *page perso* dans le cas d'un site internet à but personnel) est un ensemble de fichiers HTML (page web) stockés sur un ordinateur connecté en permanence à internet et hébergeant les pages web (serveur web).
- Un site web est habituellement architecturé autour d'une page centrale, appelée «**page d'accueil**» et proposant des liens vers  d'autres pages hébergées sur le même serveur, et parfois des liens dits «externes», c'est-à-dire de pages hébergées par un autre serveur.

# HTTP

- Protocole http :  est un ensemble normalisé de règles décrivant la manière de transmettre des informations
- HTTP HyperText Transfer Protocol, le plus utilisé des protocoles de communication sur le World Wide Web. Permet à un client Web d'indiquer quelle page il veut obtenir, et au serveur Web de lui répondre en lui donnant cette page.

# URL

- URL : *Uniform Resource Locator*
- Identifie l'endroit où se trouve une ressource(document) sur le Web.
- Une URL se présente sous la forme suivante : http://www.commentcamarche.com
- **http://** indique que nous souhaitons naviguer sur le web au moyen du protocole HTTP
- **www.commentcamarche.com** correspond à l'adresse du serveur qui héberge les pages web. Par convention les serveurs web possèdent un nom commençant par *www*. La seconde partie de cette adresse est appelée nom de domaine.

# Le << Web 1.0 >> et Le << Web 2.0 >>

- Web 1.0: ce sont les organisations qui détiennent des sites qui décident de l'information qui y figure

✓ Communication de type « one-to-many » (= diffusion)

Web 2.0 : donner le contrôle de l'information aux utilisateurs

✓ faire émerger des « réseaux sociaux »

✓ chacun peut déposer des contenus

-pour donner son avis sur un sujet donné (blogs)

- pour partager ses documents (images, vidéos … )

- pour étiqueter ( « tagger ») des contenus existants

- Exemples de sites« Web 2.0 » >" Wikipédia, Youtube,…

# Types de sites Web

- il existe deux types de sites Web : les sites dits **« statiques »** et les sites « dynamiques ».
- Les sites statiques, c'est très simple, sont des sites qui ne sont réalisés qu'avec l'aide du HTML et du CSS. On les appelle statiques car sans l'intervention d'une personne(webmaster) ces derniers ne peuvent s'actualiser. Ils sont également statiques au sens où ils ne permettent pas de véritable échange avec les visiteurs.
- Les sites dynamiques, au contraire, s'actualisent tout seuls (ou presque) et permettent véritablement d'interagir avec les visiteurs. Par exemple, dès qu'il y a un forum, un espace commentaire ou un espace de connexion pour les visiteurs, votre site est dynamique.

# Types de sites Web (suite)

- Dans le cas d'un site statique, la relation client / serveur est simplifié à son maximum : le client demande au serveur, via son navigateur généralement, à accéder à une page Web et le serveur lui répond en lui envoyant la page demandée. La page envoyée est toujours la même et aucun traitement n'est fait.
- Dans le cas d'un site dynamique, ça se complique légèrement. Cette fois-ci, le client demande à accéder à une page Web mais le serveur va générer une page potentiellement différente pour chaque visiteur la lui demandant.
- Pour en être convaincu, pensez aux utilisateurs se connectant à leur compte sur un site : la page est bien générée pour chaque visiteur et est également différente pour chacun d'entre eux. Une fois la page générée à la volée, le serveur renvoie la page demandée au client.

# Partie1: langage de programmation pour le web coté client

HTML5 &CSS3

# HTML

# Définitions et rôles du HTML

- HTML est l'abréviation de HyperText Markup Language, soit en français « langage hypertexte de balisage ». Ce langage a été créé en 1991 et a pour fonction de structurer et de donner du sens à du contenu.
- Le HTML sert, entre autres choses, à indiquer aux navigateurs quel texte doit être considéré comme un paragraphe, quel texte doit être considéré comme un titre, que tel contenu est une image ou une vidéo.
- Il ne décrit pas la présentation (ce sera le rôle de CSS)
- Il ne décrit pas de comportement dynamique (ce sera le rôle de JavaScript et des langages côté serveur)
- La version actuelle du HTML est HTML5

# L'éditeur de texte

- Pour coder en HTML, c'est très simple : nous n'avons besoin que d'un éditeur de texte. Il existe des centaines et des centaines d'éditeurs de texte. Certains sont gratuits, d'autres sont payants.
- Voici quelques exemples des éditeurs:

• NotePad++, si vous êtes sous Windows;

• Komodo, pour Mac ou Linux

• TextWrangler, pour Linux

# Eléments, balises et attributs

- Balises

Leur syntaxe est :

**<balise attributs>** contenu **</balise>** ou (balise sans contenu)

-balise: mot clé désignant un élément particulier

-contenu: peut contenir du texte ou d'autres balises

-attributs: représente les différents paramètres associés à l'élément, sous la forme d'une liste de nom="valeur" ou nom='valeur' , séparés par des espaces (les guillemets ne sont pas toujours indispensables, mais elles le deviennent dès que valeur contient des caractères exotiques)

# Eléments, balises et attributs

- Les noms des éléments et des attributs sont souvent écrits en minuscule, mais <head> et <HeAd> sont équivalents.
- Les balises sont ouvertes et refermées dans l'ordre (  et non  ).

**Exemples**

- `<title>coucou</title>` pour attribuer le titre *coucou* au document
- `<em>cuicui</em>` pour mettre en *emphase* le texte *cuicui* (cela sera rendu, le plus souvent, par une mise en italique).
- `<strong>cuicui</strong>` pour indiquer que le texte **cuicui** est important (cela sera rendu, le plus souvent, par une mise en gras).

# Structure de base d'une page en HTML5

```
<!DOCTYPE html>
<html>
 <head>
  <title> ma premiere page en html</title>
  <meta charset="utf-8"/>
 </head>
 <body >

 </body>
</html>
```

- **Le doctype**, comme son nom l'indique, sert à indiquer le type du document. Dans notre cas, le type de document est HTML.
- **L'élément html** va contenir toute la page.

# En-tête <head> </head>

- L'élément **head** contiendra entre autres, le titre de la page, l'encodage utilisé et des meta-data (des données invisibles pour les utilisateurs mais essentielles).
- Les deux informations les plus importantes sont :

-Le jeu de caractères de la page, à mettre tout au début de l'en-tête

```
<meta charset="utf-8"/>
```

- Le titre de la page (la seule information obligatoire à préciser) ; celui-ci sera par exemple affiché dans la barre de titre du navigateur, il n'apparaît pas dans la page elle-même.

```
<title> ma premiere page en html</title>
```

# En-tête

## Jeux de caractères

Unicode : répertoire de caractères, assignant à chaque caractère, de quelque langue que ce soit, un nombre entier.

**Exemples**

| | | | | | |
|---|---|---|---|---|---|
| A | → | 65 | $\varepsilon$ | → | 949 |
| é | → | 233 | ℵ | → | 1488 |

Jeu de caractères : moyen de représenter concrètement, par une suite de 0 ou de 1, un caractère Unicode.

**Exemples (é)**

| | | |
|---|---|---|
| iso-8859-1 | 11101001 | Seulement pour certains caractères |
| utf-8 | 11000011 10101001 | |
| utf-16 | 11101001 00000000 | |

utf-8 présente l'avantage de pouvoir représenter tous les caractères d'Unicode, de manière compatible avec l'ancien encodage ASCII.

Les balises <body> … </body> délimitent le corps du document.
Le corps est structuré en sections, paragraphes, listes, etc.

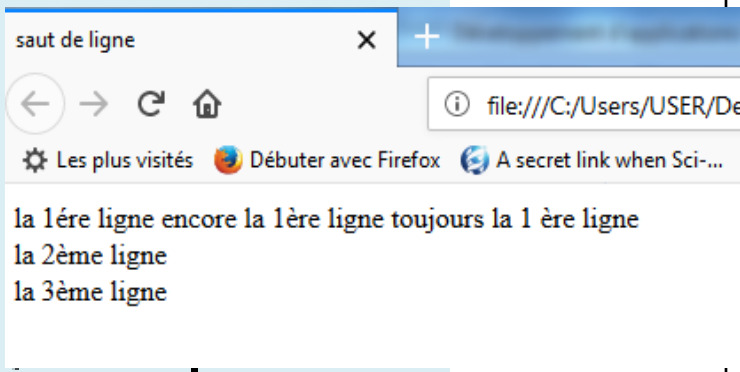
# Corps du document  <body> ... </body>

- Les balises **<body> ... </body>** délimitent le corps du document. Le corps est structuré en sections, paragraphes, listes, etc.
- Directement à l'intérieur de <body> … </body> n'apparaissent que des balises de bloc : <p> , <h1> <form> , <hr> , <ul> , <table>ainsi que la balise ** ** qui dénote un bloc sans sémantique particulière.

# Retour à la ligne : <BR/>

- <br/>:notez que, dans le cas des balises orphelines, le slash se situe après le nom de l'élément. Notez également que ce slash n'est pas obligatoire et que certains développeurs l'omettent volontairement.

```
<!DOCTYPE html>
<HTML>
 <HEAD>
  <TITLE> saut de ligne </TITLE>
  <meta charset="utf-8"/>
 </HEAD>
 <BODY>
la 1ére ligne
encore la 1ère ligne
toujours la 1 ère ligne
<BR>
la 2ème ligne <BR> la 3ème ligne
</BODY>
</HTML>
```



# Commentaires <!-- blabla -->

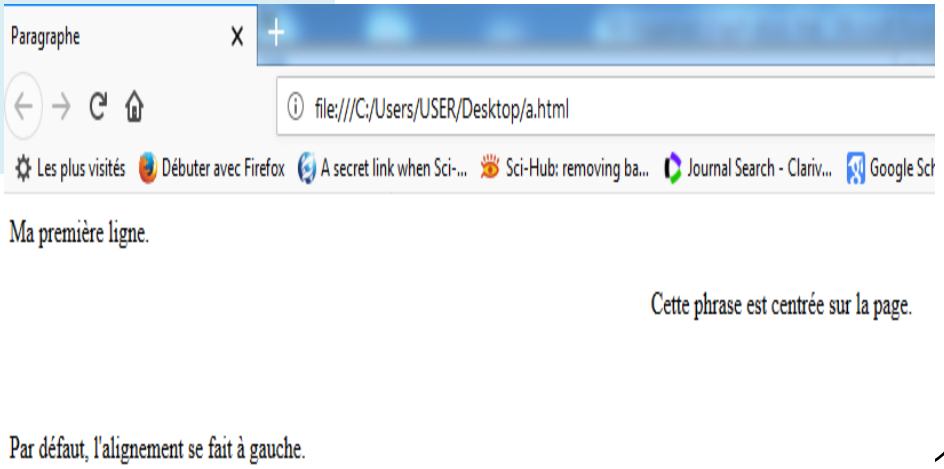
- Commenter ton code, c'est tout simplement y ajouter des commentaires. Ces commentaires sont spéciaux : il ne seront pas visibles par vos visiteurs (à moins que ceux-ci n'affichent le code source de la page).
- Voici comment on écrit un commentaire en HTML :

```
<!DOCTYPE html>
<html>
 <head>
  <title>Commentaires </title>
  <meta charset="utf-8"/>
 </head>
 <body>
  <!-- Ceci est un commentaire inséré
dans le code mais invisible pendant la
navigation -->
 </body>
</html>
```

# Les paragraphes <P> </P>

- Les balises <P> et </P> définissent le début et la fin d'un paragraphe
- Le paramètre align indique l'alignement du paragraph (left, right, center).

```
<!DOCTYPE html>
<HTML>
 <HEAD>
  <TITLE> Paragraphe </TITLE>
  <meta charset="utf-8"/>
 </HEAD>
 <BODY>
  <p> Ma première ligne. </p>
  <p align="center"> Cette phrase est
centrée sur la page. </p>
 <p align="right"> Celle-ci est alignée à
droite. </p>
 <p align="left"> Par défaut,
l'alignement se fait à gauche.
 </BODY>
</HTML>
```

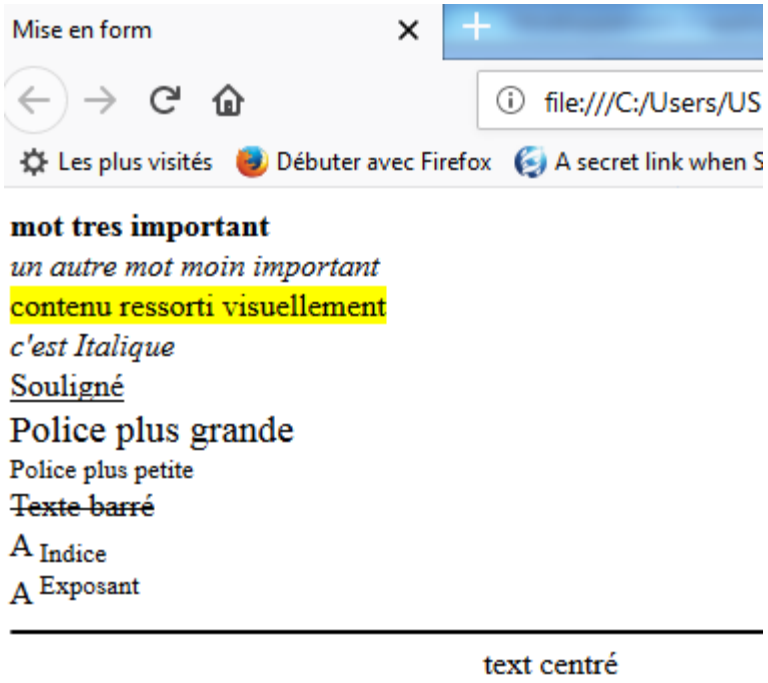


# Mise en forme de texte

- **** et **** est proche de l'élément **<STRONG> et </ STRONG >** : Il sert lui aussi à signifier qu'un contenu est important, mais moins important tout de même qu'un contenu entre des balises strong Les balises strong et em vous aident l'optimisation du référencement de votre site.
- **<mark> et </mark>:** ressortir l'élément visuellement
- **<I> et </I>:**Italique
- **<U> et </U>**:souligné
- **<BIG> et </BIG> :**Police plus grande
- **<SMALL> et </SMALL>**:Police plus petite
- **<STRIKE> et </STRIKE>** ou **<S> et </S>**:Texte barré
- **<SUB>** et **</SUB>**: Indice
- **<SUP>** et **</SUP>**:Exposant
- ** et  :** centrer un text
- **<HR>:** Ligne horizontale

# Mise en forme

```
<html>
<head>
<title> Mise en form </title>
<meta charset="utf-8"/>
</head>
<body >
<strong> mot tres important </strong> <br>
<em> un autre mot moin important</em> <br>
<mark> contenu ressorti visuellement</mark>
<br> <I> c'est Italique </I> <br>
<U> Souligné </U> <br>
<BIG> Police plus grande </BIG> <br>
<SMALL> Police plus petite </SMALL> <br>
<STRIKE> Texte barré </STRIKE> <br>
A<SUB> Indice </SUB> <br>
A<SUP> Exposant </SUP>
<HR SIZE=3 WIDTH=50% ALIGN=left COLOR=BLACK>
<center> text centré </center>
</body>
</html>
```



# Mise en forme de texte (suite)

- On définit la police de caractères, taille, et couleur comme suit :

**<FONT FACE="Arial" SIZE="5" COLOR="red"> et </FONT>**

-la police pouvant être : Arial, Arial Black, Comic sans MS, Courier New, Impact, Times new roman, Verdana, et

-size de 1 à 7, et color étant une couleur (en anglais) ou un code RGB (#004422).

```
<!DOCTYPE html>
<html>
<head>
<title> Mise en form du texte </title>
<meta charset="utf-8"/>
</head>
<body >
<FONT FACE="Arial" SIZE="5" COLOR="red">
Un texte Arial Rouge
</FONT>
<br>
<FONT FACE="times" SIZE="3" COLOR="blue">
Un texte Time Times new roman
</FONT>
</body>
</html>
```

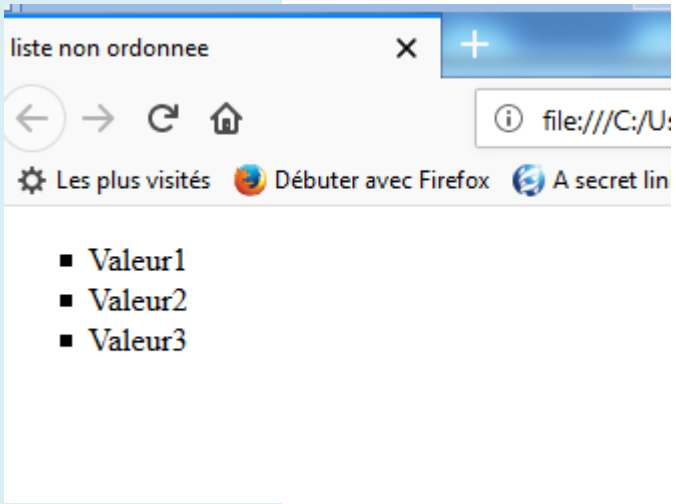
# Titres

- Pour créer des titres, on va utiliser les éléments h1, h2, h3, h4, h5 et h6. Pourquoi autant d'éléments différents ?
- Pour pouvoir créer des titres de diverses importances. Ainsi, un titre h1 sera considéré comme un titre très important tandis qu'un titre h6 sera considéré comme très peu important.

```
<!DOCTYPE html>
<html>
<head>
<title> Titres </title>
<meta charset="utf-8"/>
 </head>
<body >
<H1> Titre 1 </H1>
<H2> Titre 2 </H2>
<H3> Titre 3 </H3>
<H4> Titre 4 </H4>
<H5> Titre 5 </H5>
<H6> Titre 6 </H6>
</body>
</html>
```

# Listes non ordonnées

```
<!DOCTYPE html>
<html>
<head>
<title> liste non ordonnee </title>
<meta charset="utf-8"/>
</head>
<body >

<UL type="square">
<LI> Valeur1 </LI>
<LI> Valeur2 </LI>
<LI> Valeur3 </LI>
</UL>

</body>
</html>
```




- **On peut préciser le type de « puces » (disc, circle, square)**

# Listes Ordonnées

- On peut préciser le type (A, a, 1, i, I). Exemple : **<OL TYPE="A">**
- On peut faire démarrer à une valeur précise. **Exemple : <OL START=2000>**

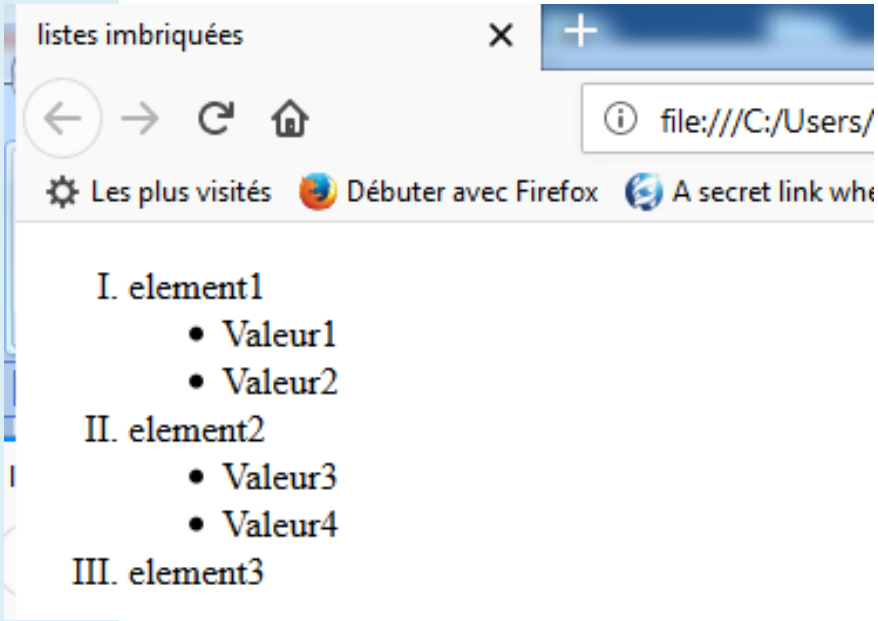
```
<!DOCTYPE html>
<html>
<head>
<title> liste ordonnee </title>
<meta charset="utf-8"/>
</head>
<body >

<OL>
<LI> Valeur1 </LI>
<LI> Valeur2 </LI>
<LI> Valeur3 </LI>
</OL>

<OL TYPE="A">
<LI> Valeur1 </LI>
<LI> Valeur2 </LI>
<LI> Valeur3 </LI>
</OL>

</body>
</html>
```


1. Valeur1
2. Valeur2
3. Valeur3

A. Valeur1
B. Valeur2
C. Valeur3

# Listes imbriquées

```
<!DOCTYPE html>
<html>
<head>
<title> listes imbriquées </title>
<meta charset="utf-8"/>
</head>
<body >

<OL TYPE="I">
<LI> element1 </LI>
  <UL TYPE="disc">
    <LI> Valeur1 </LI>
    <LI> Valeur2 </LI>
  </UL>
<LI> element2 </LI>
    <UL TYPE="disc">
    <LI> Valeur3 </LI>
    <LI> Valeur4 </LI>
  </UL>

<LI> element3 </LI>
</OL>

</body>
</html>
```



# INSERER DES IMAGES <img>

- L'insertion d'images en HTML va se faire au moyen de l'élément HTML **<img>.** Cet élément est représenté par une balise orpheline.
- Au sein de l'élément img, nous allons obligatoirement devoir préciser deux attributs : les attributs **src** et **alt**.
- L'attribut src (pour source) va prendre comme valeur l'adresse de l'image (adresse relative ou absolue) tandis que l'attribut alt (pour alternative) va contenir un texte alternatif décrivant l'image. Ce texte va être affiché si l'image ne peut pas l'être pour une raison ou pour une autre
- **Exemple :**

**<img src= 'image/Tulipes.jpg' alt= 'des fleurs jaunes' >**

- Les différents formats d'image: comme vous le savez certainement, vous pouvez enregistrer vos images sous différents formats. Les formats les plus utilisés sont :

  -Le JPG ou JPEG ;

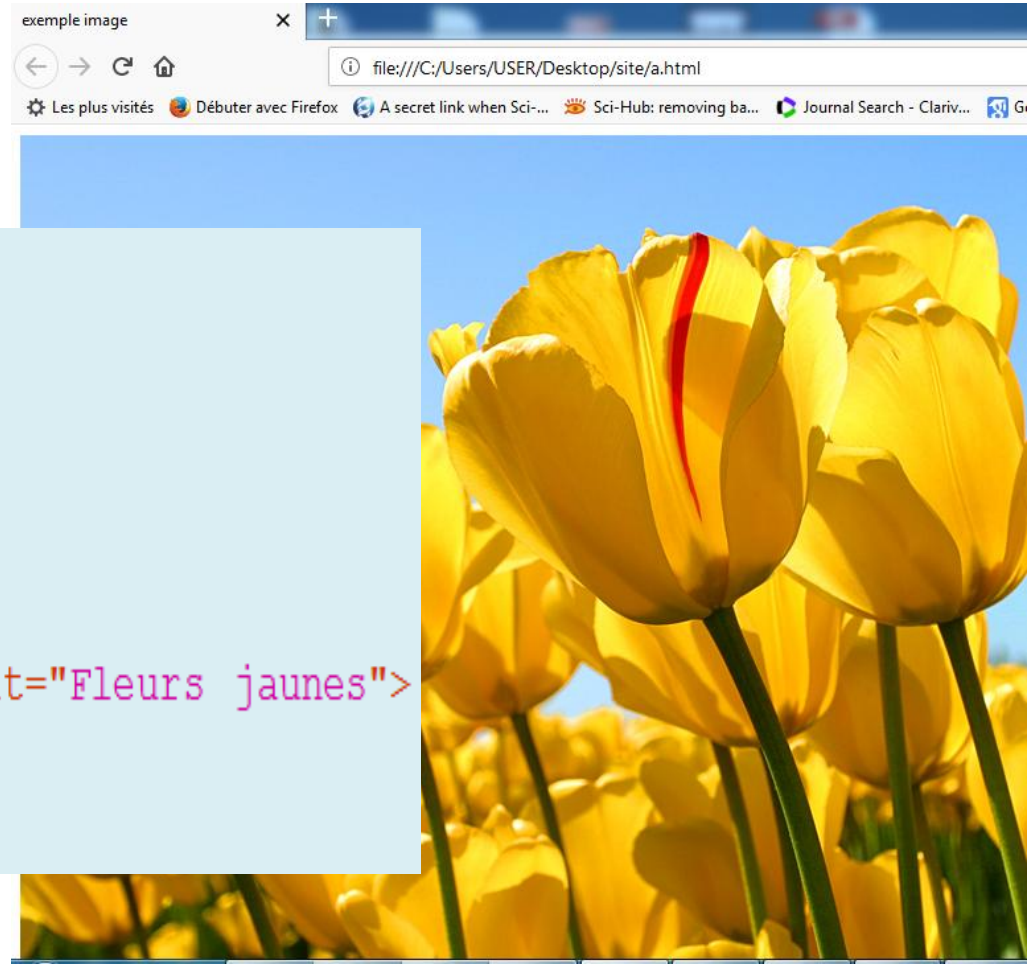
  -Le PNG ;

  -Le GIF ;

  -Le BITMAP.

# INSERER DES IMAGES EN HTML

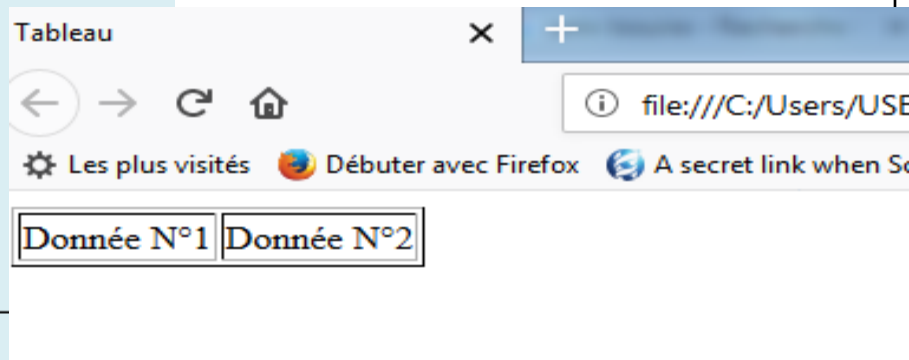
```
<!DOCTYPE html>
<html>
 <head>
<title>exemple image </title>
<meta charset="utf-8"/>
</head>
 <body bgcolor="#FFFFFF">
<img src="image/Tulips.jpg" alt="Fleurs jaunes">
 </body>
 </html>
```



# Les tableaux < TABLE> </TABLE>

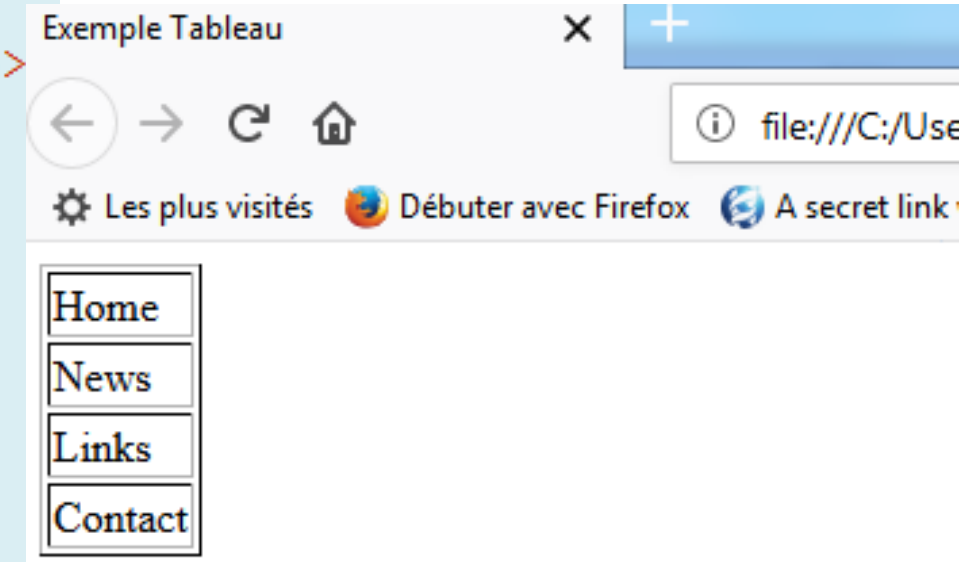
- <TR> correspond à une ligne du tableau
- <TD> correspond à une donnée (une cellule) du tableau

```
<!DOCTYPE html>
<html>
 <head>
<title>Tableau </title>
<meta charset="utf-8"/>
</head>
 <body bgcolor="#FFFFFF">

<TABLE Border=N Align="left">
<TR>
<TD>Donnée N°1</TD> <TD> Donnée N°2</TD>
</TR>
</TABLE>

</body>
 </html>
```



# Exemple Tableau

```
<!DOCTYPE html>
<html>
 <head>
<title> Exemple Tableau </title>
<meta charset="utf-8"/>
</head>
<body bgcolor="#FFFFFF">
<TABLE border=1 ALIGN="left">
<TR> <TD>Home </TD> </TR>
<TR> <TD>News </TD> </TR>
<TR> <TD>Links </TD> </TR>
<TR> <TD>Contact</TD> </TR>
</TABLE>
</body>
</html>
```



# Les hyperliens <A> </A>

- La balise **<A> </A>** sert à définir un hyperlien (une ancre hypertextuelle) dans une page Web.
- **<A HREF= "Chemin du document"> un lien vers doc </A>**
- Deux types de liens: interne et externe

✓ Externe : vers un autre site

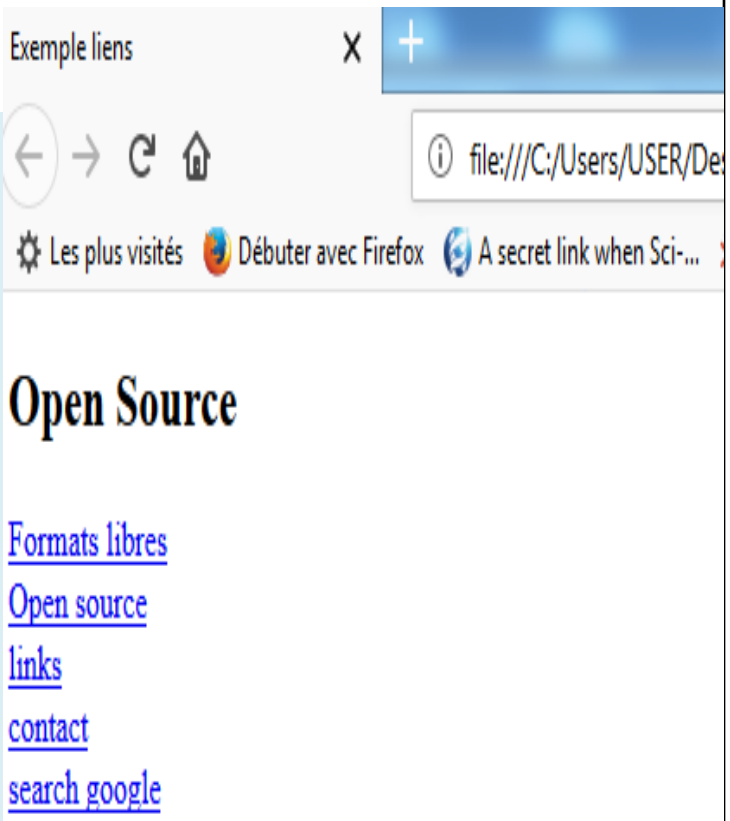
```
<A HREF="http://www.openxnet.com.dz">Openxnet</A>
```

✓Interne : vers un fichier situé dans un dossier parent

```
<A HREF="formation.html"> Formation </A>
```

# Les hyperliens

```
<!DOCTYPE html>
<html>
 <head>
<title> Exemple liens </title>
<meta charset=”utf-8”/>
</head>
<body bgcolor="#FFFFFF">
<h2> Open Source </h2>
<a href="format.html" >Formats libres</a> <br>
<a href="initiative.html"> Open source </a> <br>
<a href="link.html" >links</a><br>
<a href="contact.html">contact</a><br>
<a href="http://www.google.com">search google</a>
<br>
</body>
</html>
```



# Formulaires <form> </form>

- Permettent d'interagir avec l'utilisateur en lui proposant d'entrer des informations
- En HTML : uniquement l'interface de formulaire
- L'essentiel du travail sera fait par le script qui traitera la soumission du formulaire (PHP & MYSQL)
- Cet élément **form** va avoir besoin de deux attributs pour fonctionner normalement : les attributs **method** et **action**.
- L'attribut **action** va lui nous servir à préciser l'adresse relative de la page dans laquelle les données doivent être traitées. Ce sera généralement un fichier PHP.
- L'attribut **method** va indiquer comment doivent être envoyées les données saisies par l'utilisateur. Cet attribut peut prendre deux valeurs : **get** et **post**.

```
<!DOCTYPE html>
<html>
 <head>
<title> formulaire </title>
<meta charset="utf-8"/>
</head>
<body bgcolor="#FFFFFF">

<form action="action.php" method="post">
// ici nous allons creer notre formulaire
</form>

</body>
</html>
```

# Les éléments de formulaire

| Elément | Définition |
|---|---|
| form | Définit un formulaire |
| input | Définit un champ de données pour l'utilisateur |
| label | Définit une légende pour un élément input |
| textarea | Définit un champ de texte long |
| select | Définit une liste de choix |
| optgroup | Définit un groupe d'options dans une liste |
| option | Définit une option dans une liste |
| fieldset | Permet de regrouper les éléments d'un formulaire en différentes parties |
| legend | Ajoute une légende à un élément fieldset |

# L'attribut name

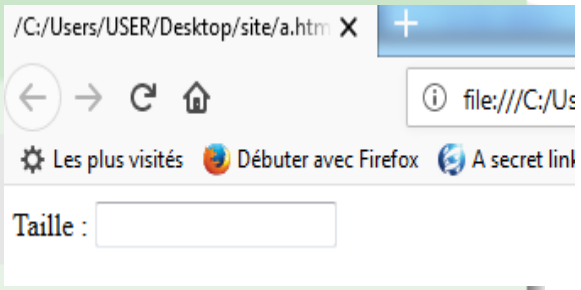
- Nous allons également devoir préciser un attribut name pour chaque élément de notre formulaire demandant des informations à un utilisateur.
- Cet attribut name va nous permettre, par la suite, de reconnaître le contexte de chaque donnée envoyée par l'utilisateur afin de pouvoir les traiter efficacement.
- En effet, sans attribut name, nous recevrions des données mais ne saurions pas quoi en faire, ne sachant pas à quel champ elles appartiennent.
- Cet attribut est donc indispensable. Vous pouvez une nouvelle fois lui attribuer la valeur souhaitée. Seule restriction : cette valeur doit être unique afin de bien pouvoir identifier chaque champ.

# Étiquettes <label> </label>

- La plupart des champs sont naturellement accompagnés d'une étiquette ( `<label>` ).
- On peut la placer où on veut, en général juste à gauche ou à droite du champ.
- Son attribut `for` référence l'attribut `id` du champ correspondant.
- Dans les navigateurs graphiques, un clic sur l'étiquette d'un champ permet en général de sélectionner le champ.

## Example (Étiquette)

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



# Champs de saisie

- La balise `<input>` a une utilisation très vaste dans les formulaires. Elle représente un champ de saisie.
- L'attribut `type` détermine le type (texte, mot de passe, liste, etc.) du champ.
- L'attribut `name` (nom du paramètre de la requête HTTP) est obligatoire (sauf pour les types `"reset"` et `"submit"`) ; il permet de préciser au serveur à quelle saisie on fait référence.

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

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

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

**Exemple**

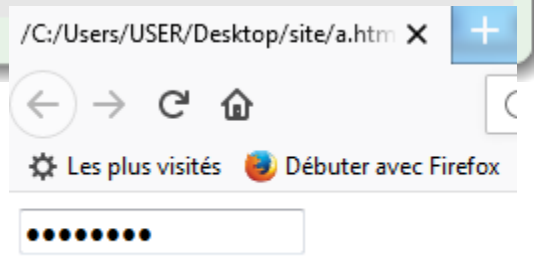
```
<input type="text" name="prenom" value="Jordy" maxlength="50">
```

# Saisie d'un mot de passe

- `type ="password"` est utilisé pour la saisie d'un texte dont les caractères sont remplacés par des astérisques : c'est généralement utilisé pour la saisie des mots de passe. Le mot de passe est quand même transmis en clair au serveur !
- L'attribut `value` permet de préciser la valeur par défaut (facultatif).
- La taille maximale de la chaîne de caractères à saisir peut être spécifiée à l'aide de l'attribut `maxlength` (facultatif).

**Exemple**

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

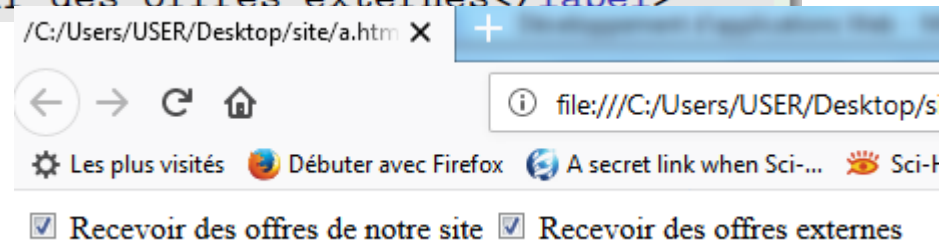


# Choix multiples parmi une liste

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

## Exemple

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

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

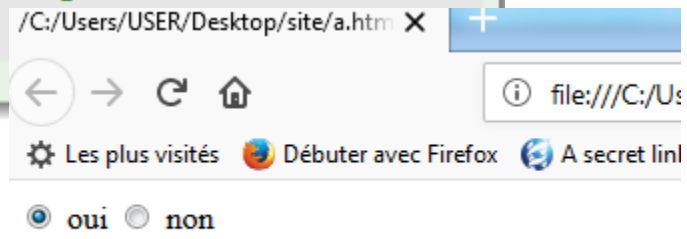


# Choix unique parmi une liste

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

## Exemple

```
Recevoir de la pub:
<input type="radio" name="pub" value="oui" id="pub-oui"
  checked="checked">
<label for="pub-oui">oui</label>

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

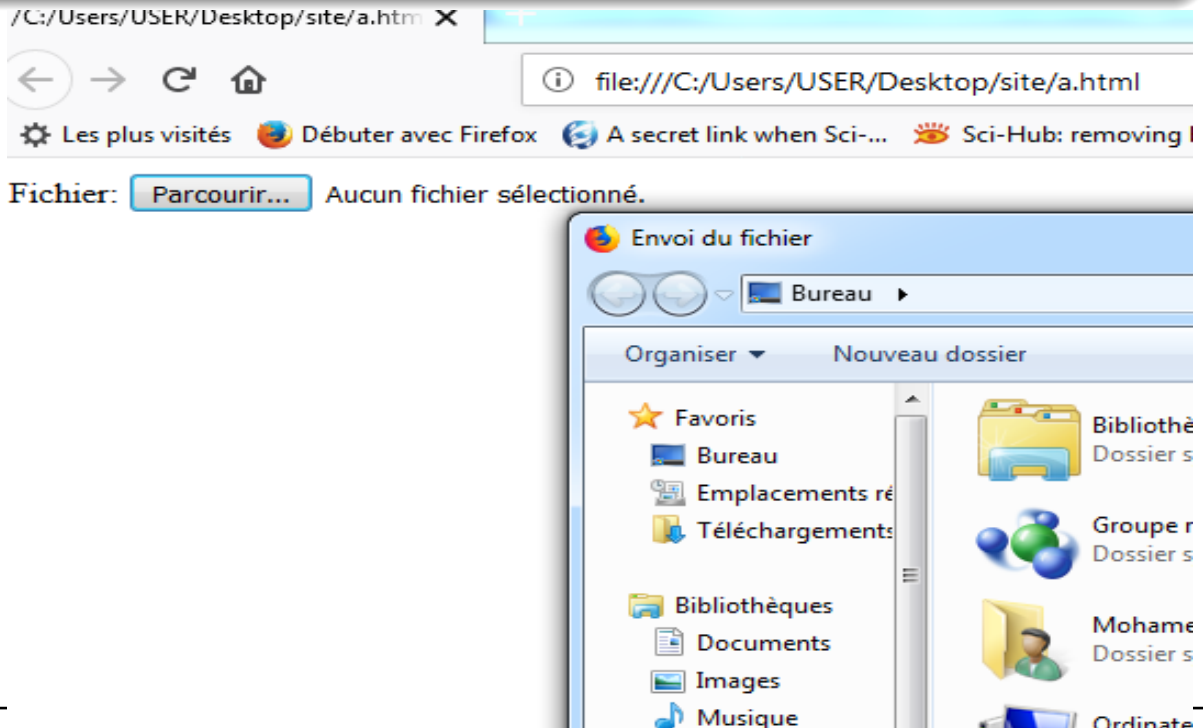


# Fichiers joints

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

**Exemple**

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

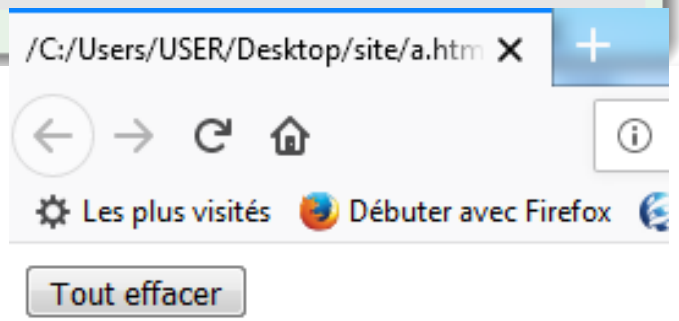


# Ré-initialisation d'un formulaire

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

**Exemple**

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

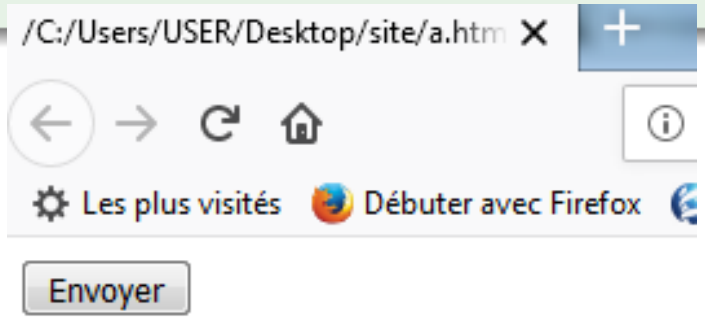


# Soumettre le formulaire

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

**Exemple**

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

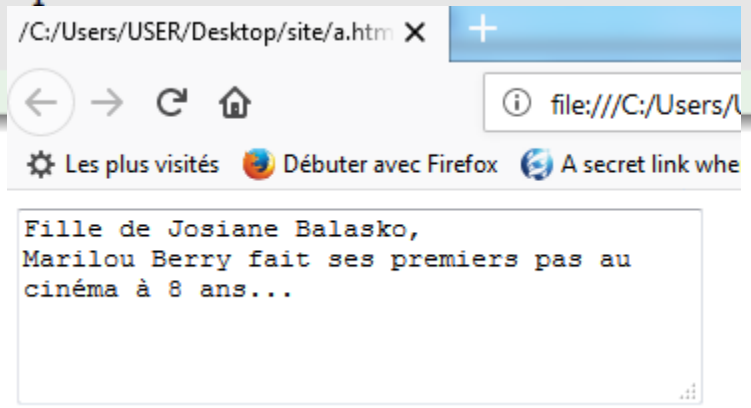


# Saisie de plusieurs lignes de texte

- Pour les saisies multiligne, on utilise la balise `<textarea>`.
- Le texte délimité par cette balise permet d'initialiser la valeur par défaut du champ.
- La balise fermante est obligatoire même si le champ est vide.
- Les attributs `rows` et `cols` (obligatoires) permettent de spécifier la taille en lignes et colonnes de la fenêtre de saisie.

## Exemple

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

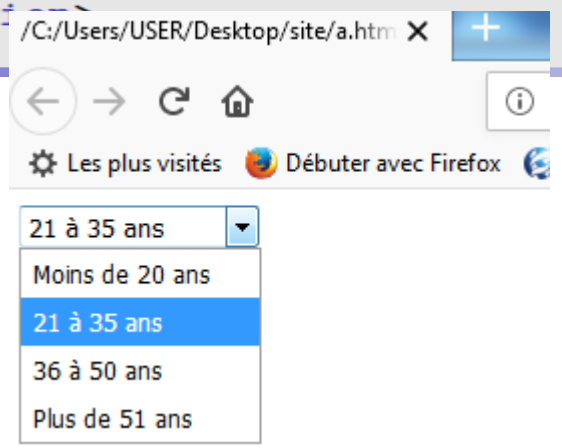


# Menus de sélection

- La balise `<select>` permet d'ajouter une liste de sélection :
  - L'attribut facultatif `size` permet de préciser le nombre de choix apparaissant sur la page Web. Par défaut, ce nombre est initialisé à 1.
  - L'attribut `multiple ="multiple"` permet d'autoriser des réponses multiples. Dans ce cas, pour PHP, on donnera toujours un nom se terminant par `[]`.
- Les choix du menu sont indiqués à l'aide de la balise `<option>` :
  - L'attribut `selected ="selected"` permet de spécifier le(s) choix par défaut.
  - L'attribut `value` permet de spécifier la valeur associée au choix.

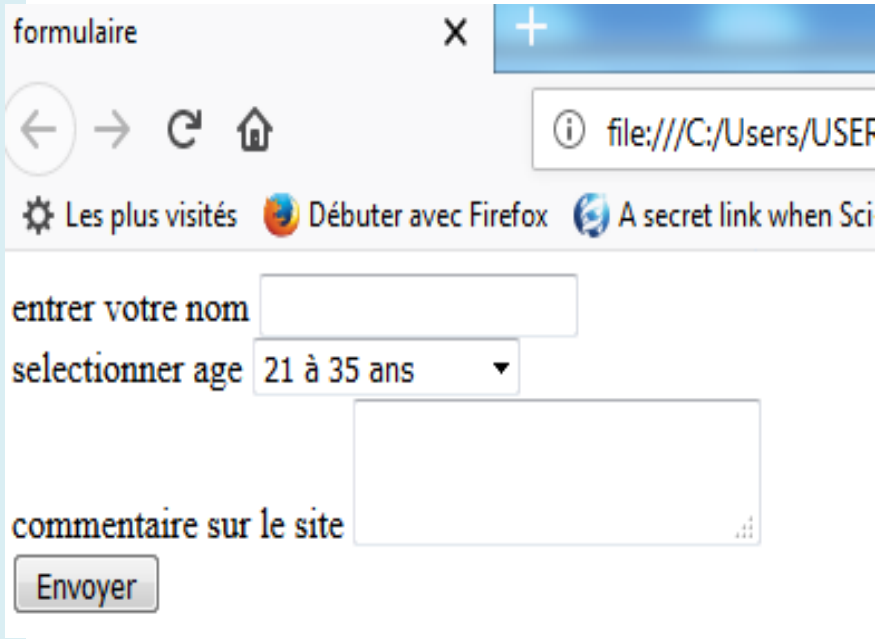
**Exemple**

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



# Forme.html

```
<!DOCTYPE html>
<html>
 <head>
<title> formulaire </title>
<meta charset=”utf-8”/>
</head>
<body bgcolor="#FFFFFF">
<form action="action.php" method="post">
<label for=’nom’ > entrer votre nom </label>
<input type="text" name="nom" id= "nom">
<label> selectionner age </label>
<select name="age">
<option value="20">Moins de 20 ans</option>
<option value="35" selected="selected">21 à 35
ans</option>
<option value="50">36 à 50 ans</option>
<option value="51">Plus de 51 ans</option>
</select>
<label> commentaire sur le site </label>
<textarea >

 </textarea>
<input type="submit" value="envoyer">
</form>
</body>
</html>
```



# Les balises structurantes de HTML5

- **<header> : l'en-tête**

**Code : HTML**

```
<header>
    <!-- Placez ici le contenu de l'en-tête de votre page -->
</header>
```

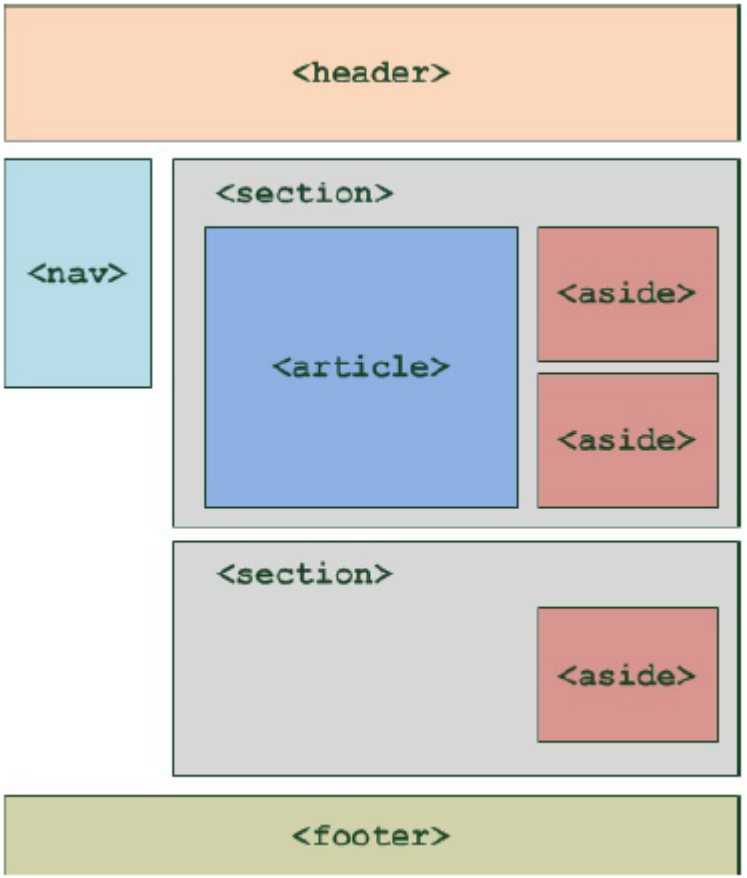
- **<footer> : le pied de page**

**Code : HTML**

```
<footer>
    <!-- Placez ici le contenu du pied de page -->
</footer>
```

- **<nav> : principaux liens de navigation**
- **<aside> : informations complémentaires**
- **<article> : un article indépendant**
- **<section>:** sert à regrouper des contenus en fonction de leur thématique. Elle englobe généralement une portion du contenu au centre de la page.

# Les balises structurantes de HTML5: résumé




Pour créer ce squelette, nous allons utiliser différentes balises HTML :

les balises structurantes de HTML5, que nous connaissons : **<header>, <section>, <nav>, etc. ;**

la balise universelle ** ** quand aucune balise structurante ne convient.

Comment je sais quelle balise utiliser moi ?

C'est à vous de décider. De préférence, utilisez une balise qui a du sens (comme les balises structurantes **<header>, <section>, <nav>)** mais, si aucune balise ne vous semble mieux convenir, optez pour la balise générique ** .**

# Un mot sur le XHTML

- XHTML signifie eXtensible HTML. Crée en 2000, il devait à l'origine succéder au HTML. XHTML 1.0 (Second Edition) est la reformulation de **HTML 4 avec XML 1.0**
- En effet, il faut savoir que le HTML base sa syntaxe sur le langage **SGML** (Standard Generalized Markup Language) tandis que le XHTML se fonde sur le **XML** (eXtensible Markup Language).
- Or, à l'époque, le XML était une véritable révolution puisqu'il permettait de faire davantage de choses que le SGML et qu'il était dit plus simple à utiliser.
- Cependant, le HTML a continué à se développer en parallèle au XHTML et le XHTML est définitivement abandonné en 2009 ou plus exactement a été en partie intégré dans ce qui allait devenir le HTML5.

# Validateur HTML et CSS

- Pour vérifier la validité d'un code HTML ou CSS, le w3c (World Wide Web Consortium), c'est-à-dire l'organisme qui gère l'évolution des langages comme le HTML et le CSS entre autres, a mis à disposition des validateurs de code, gratuits.
- Vous pouvez trouver les validateurs HTML et CSS aux adresses suivantes :
- HTML : http://validator.w3.org/
- CSS : http://jigsaw.w3.org/css-validator/

# CSS

Après avoir passé toute une première partie du cours à ne travailler que sur le HTML, nous allons maintenant découvrir le CSS. Le CSS n'est pas plus compliqué que le HTML. Il vient le compléter pour vous aider à mettre en forme votre page web.

HTML
(pas de CSS)


La même page HTML, sans et

HTML + CSS


avec CSS (www.csszengarden.com)

# un concept important

**séparation de la forme et du contenu**

il faut définir séparément la forme et le contenu

- la structure d'un document et son contenu sont décrits en HTML
- sa présentation est gérée par les CSS

1. on crée le document (contenu et structure) sans se préoccuper de sa mise en forme
2. on conçoit la (les !) mise(s) en forme

   puis éventuellement on les modifie/adapte

# avantages

- document HTML et feuille CSS peuvent être définis dans des fichiers séparés
  - création plus efficace
  - code HTML plus simple et plus lisible
  - on peut changer la feuille de style sans modifier le document
  - on peut avoir plusieurs feuilles de style pour un document
- l'homogénéité visuelle d'un site est facilitée
  - plusieurs pages peuvent partager la même feuille de style

# principe

- le langage CSS définit un ensemble de **propriétés** qui ont une influence sur l'affichage des éléments d'une page
- pour chaque propriété il existe un ensemble de **valeurs** possibles
- il est possible de fixer ces propriétés pour chacun des éléments d'un document HTML
- les propriétés agissent sur l'apparence de la **boîte** d'un élément
- les propriétés concernent
  - l'apparence du contenu (fonte, style, couleur, ...)
  - la taille de la boîte (largeur, marges, ...)
  - le positionnement de la boîte (absolu ou relatif, visibilité)
  - ...

# Règle CSS

## Règle CSS

Une **règle CSS** définit

la **valeur** d'une **propriété CSS** pour un **sélecteur** donné

```
selecteur { propriete :  valeur }
```

règle

le **sélecteur** détermine les éléments sur lesquels s'applique la règle

# syntaxe

- il est possible de regrouper plusieurs règles d'un même sélecteur les définitions sont alors séparées par des points-virgules

```
h1 {
  color : blue;
  font-size : 12px;
}
```

*« tous les éléments <h1> auront leur texte en bleu et une taille de police de 12px »*

# syntaxe

- on peut factoriser les règles partagées par des sélecteurs
  les sélecteurs sont alors séparés par des virgules

```
h1, h2 {
   color : blue;
   font-size : 12px;
}
```

*« les éléments <h1> et les éléments <h2> auront leur texte en bleu et une taille de police de 12px »*

# syntaxe

** : sélecteur universel*

**Code : CSS**

```
*
{

}
```

Sélectionne toutes les balises sans exception. On l'appelle le sélecteur universel.

# syntaxe

*A B : une balise contenue dans une autre*

**Code : CSS**

```
h3 em
{

}
```

Sélectionne toutes les balises **** situées à l'intérieur d'une balise **<h3>**. Notez qu'il n'y a pas de virgule entre les deux noms de balises.

Exemple de code HTML correspondant :

**Code : HTML**

```
<h3>Titre avec <em>texte important</em></h3>
```

# syntaxe

***A + B : une balise qui en suit une autre***

**Code : CSS**

```
h3 + p
{

}
```

Sélectionne la première balise **<p>** située après un titre **<h3>**.

Exemple :

**Code : HTML**

```
<h3>Titre</h3>

<p>Paragraphe</p>
```

# Exemples de propriétés

- `font-family` : le type de police utilisée pour le contenu,
- `font-size` : la taille des caractères (en px, em, %, etc.)
- `font-style` : *`normal`*, *`italic`*, *`oblique`*
- `font-weight` : *`normal`*, *`bold`*, *`lighter`*, etc.
- `border` : la bordure autour du contenu de l'élément (couleur, style, ...)
- `width` : largeur du contenu (%, px, em, cm)
- `color` et `background-color` : couleurs du texte et de l'arrière-plan (*`rgb(0,128,255)`*, hexa *`#AAAAAA`*, symboles prédéfinis (*`navy`*, *`white`*, ...), hsl,)

# feuille de style

**feuille de style**

une **feuille de style** CSS regroupe un ensemble de règles CSS

# intégration des règles CSS à l'HTML

- Vous avez le choix car on peut écrire du code en langage CSS à trois endroits différents :

✓ dans un fichier .css (*méthode la plus recommandée) ;*

✓ dans l'en-tête **<head> du fichier HTML ;**

✓ directement dans les balises du fichier HTML *via un attribut style (méthode la moins recommandée).*

# Dans un fichier .css (recommandé)

- on écrit le plus souvent le code CSS dans un fichier spécial ayant l'extension .css (contrairement aux fichiers HTML qui ont l'extension .html). C'est la méthode la plus pratique et la plus souple. Cela nous évite de tout mélanger dans un même fichier.
- **<link rel="stylesheet" href="style.css" />** : c'est elle qui indique que ce fichier HTML est associé à un fichier appelé **style.css** et chargé de la mise en forme.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="style.css" />
        <title>Premiers tests du CSS</title>
    </head>

    <body>
        <h1>Mon super site</h1>

        <p>Bonjour et bienvenue sur mon site !</p>
        <p>Pour le moment, mon site est un peu <em>vide</em>.
Patientez encore un peu !</p>
    </body>
</html>
```

# Dans l'en-tête <head> du fichier HTML

- Cela consiste à insérer le code CSS directement dans une balise **<style>** à l'intérieur de l'en-tête **<head>.**

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <style>
            p
            {
                color: blue;
            }
        </style>
        <title>Premiers tests du CSS</title>
    </head>

    <body>
        <h1>Mon super site</h1>

        <p>Bonjour et bienvenue sur mon site !</p>
        <p>Pour le moment, mon site est un peu <em>vide</em>.
Patientez encore un peu !</p>
    </body>
</html>
```

# Appliquer un style : class et id

- les attributs class et id sont quasiment identiques ,id fonctionne exactement de la même manière que class pour identifier une balise dans html , à un détail près : **id** ne peut être utilisé *qu'une fois dans le code*.

**Code : CSS**

```css
.introduction
{
    color: blue;
}
```

- Si vous utilisez des class, lorsque vous définirez leurs propriétés dans le fichier CSS, il faudra faire précéder le nom de class par un point (.) :

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="style.css" />
        <title>Premiers tests du CSS</title>
    </head>

    <body>
        <h1>Mon super site</h1>

        <p class="introduction">Bonjour et bienvenue sur mon site !</p>
        <p>Pour le moment, mon site est un peu <em>vide</em>. Patientez encore un peu !</p>
    </body>
</html>
```

# Appliquer un style : class et id

- Si vous utilisez des id, lorsque vous définirez leurs propriétés dans le fichier CSS, il faudra faire précéder le nom de l'id par un dièse (#) :

**Code : HTML**

```html
<img src="images/logo.png" alt="Logo du site" id="logo" />
```

**Code : CSS**

```css
#logo
{
    /* Indiquez les propriétés CSS ici */
}
```

#  et 

l'attribut `class` donne son sens aux balises génériques

 et 

-  : élément générique pour contenus phrasés
-  : élément générique pour contenus blocs

aucune sémantique particulière

avec `class` ajout d'un niveau de sémantique faible

- `<span class="celebrite">Winston Churchill</span>`
- `<div class="cursus">`

mais il faut utiliser les balises existantes quand c'est possible

-  plutôt que `<span class="important">`
- <time> plutôt que `<span class="date">`
- <h2> plutôt que `<div class="titre">`
- <header> plutôt que `<div class="entete">`

#  et 

**Code : HTML**

```html
<p>Bonjour et <span class="salutations">bienvenue</span> sur mon
site !</p>
```

**Code : CSS**

```css
.salutations
{
    color: blue;
}
```

# N'oubliez pas : HTML pour le fond, CSS pour la forme

# Partie2: langage de programmation coté serveur(PHP)

PHP&MYSQL

# Introduction

# Qu'est-ce que le PHP ?

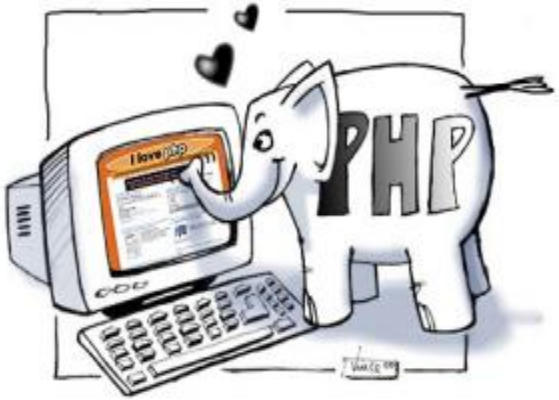

- PHP : Personal Home Pages / Hypertext PreProcessor

- PHP est un langage de programmation interprété côté serveur
- Inventé par Rasmus LEDOORF (v1 : 1994, v2 : 1996, … v5 : 2004)
- Il est dédié au Web : traitement des formulaires, communication avec des bases de données (souvent couplé à MySql)

# Qu'est-ce que le PHP ?

**Avantages**

- Il est simple (pas typé, interprété, …)
- Il est gratuit (?)
- Il est fait pour le Web
- Il est TRES répandu

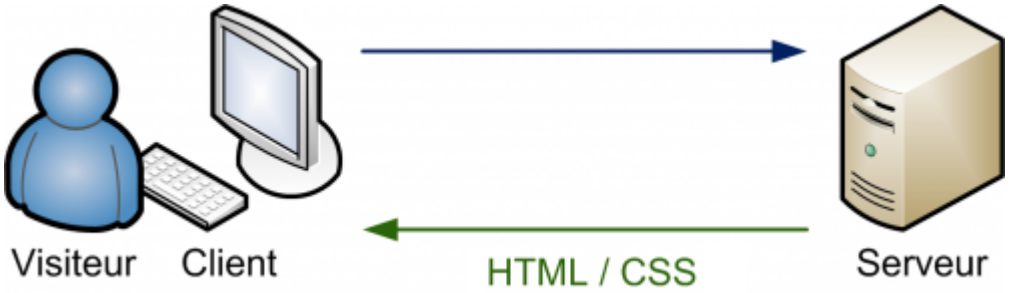




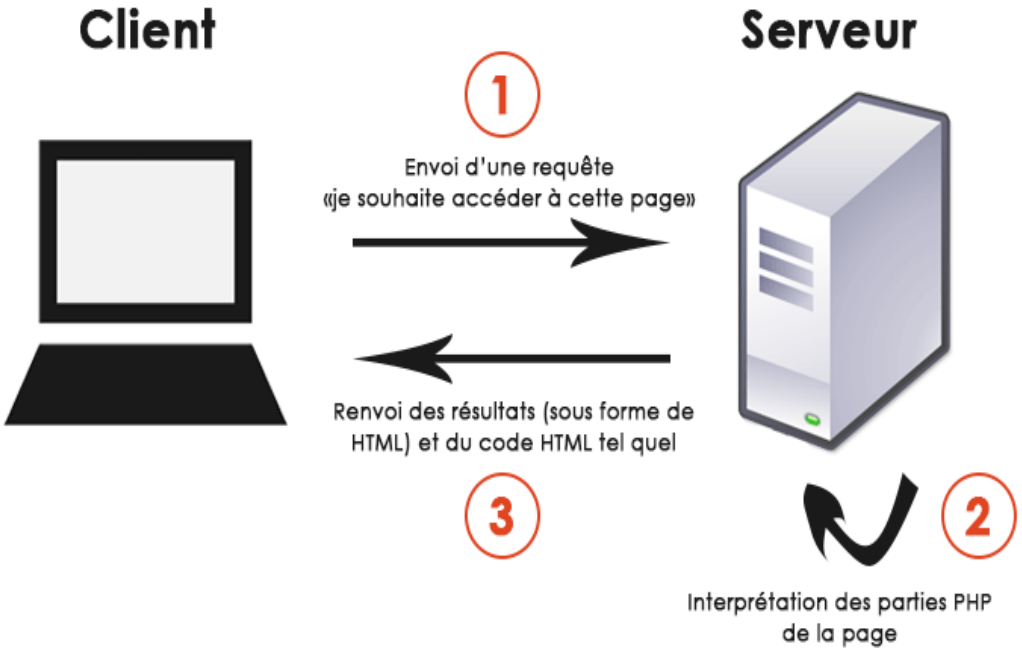
**Inconvénient**

- Code difficilement maintenable

# Comment fonctionne un site web ?




**Cas site  Statique :l**e serveur envoie du HTML et du CSS au client




**Cas site dynamique** :PHP  génère du code HTML ,le code qui est ensuite envoyé au client de la même manière qu'un site statique

# CREER UN ENVIRONNEMENT DE TRAVAIL POUR LE PHP

- **Avec un site dynamique**

Pour que votre ordinateur puisse lire du PHP, il faut qu'il se comporte comme un serveur. Rassurez-vous, vous n'avez pas besoin d'acheter une machine spéciale pour cela : il suffit simplement d'installer les mêmes programmes que ceux que l'on trouve sur les serveurs qui délivrent les sites web aux internautes.

- Ces programmes dont nous allons avoir besoin, sont:

  **Apache** : le serveur web ;

  **PHP :** le programme qui permet au serveur web d'exécuter des pages PHP ;

  **MySQL** : le logiciel de gestion de bases de données.

- Il est plus simple pour nous d'installer un paquetage tout prêt : WAMP sous Windows, MAMP sous Mac OS X ou XAMPP sous Linux.

# APPRENDRE LES BASES DU PHP

Voici une balise PHP :

```
<?php
/* Le code PHP se met ici
Et ici
Et encore ici */
?>
```

# Emplacement du code PHP

Nous allons pouvoir insérer des scripts PHP au sein de nos pages HTML. On va pouvoir écrire nos scripts **PHP à n'importe quel endroit dans nos documents.**
Cependant, généralement, nous placerons nos scripts PHP en **fin** de document, juste avant la balise fermante de l'élément **body**
Nous allons devoir placer tout notre code PHP au sein d'une balise

**<?php ?>** afin que celui-ci soit bien reconnu en tant que code PHP.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Démarrer avec PhP</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>Un titre écrit en HTML</h1>
        <p>Un paragraphe écrit en HTML</p>

        <?php
            //Du code PhP
        ?>
    </body>
</html>
```

# Insérer une balise PHP au milieu du code HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Démarrer avec PhP</title>
        <meta charset="utf-8">
        <?php
            //Un premier script PhP
        ?>
    </head>
    <body>
        <?php
            //Un autre script PhP
        ?>
        <h1>Un titre écrit en HTML</h1>
        <p>Un paragraphe écrit en HTML</p>

        <?php
            echo "Cette phrase est écrite en PhP";
        ?>
    </body>
</html>
```

# Enregistrer un fichier PHP et l'afficher

Dès lors qu'une balise PHP est ouverte dans un document HTML, celui-ci devient de fait un fichier PHP et il faut donc l'enregistrer avec l'extension .php. Cela est essentiel afin que le code PHP s'exécute correctement.

Cependant, on ne va pas pouvoir enregistrer notre fichier n'importe où sur notre ordinateur, comme on pouvait le faire avec un fichier HTML par exemple.

En effet, rappelez vous qu'un ordinateur n'est pas capable de comprendre ni de lire du code PHP à priori. C'est bien pour cela qu'on a installé WAMP, MAMP ou XAMPP.

Il vous faudra donc enregistrer votre fichier dans un sous dossier correspondant à votre logiciel afin de pouvoir exécuter et afficher celui-ci correctement.

Si vous avez installé WAMP, par exemple, il faudra enregistrer tous vos fichiers PHP dans le sous-dossier « wamp ».

# PREMIERES INSTRUCTIONS PHP

# Afficher un résultat en PHP avec echo ou print

on va devoir utiliser des guillemets ou des apostrophes pour renvoyer du texte avec **echo** ou **print**.

```php
<!DOCTYPE html>
<html>
    <head>
        <title>Démarrer avec PhP</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>Un titre écrit en HTML</h1>
        <p>Un paragraphe écrit en HTML</p>

        <?php
            echo "Résultat renvoyé par echo";
            print "Résultat renvoyé par print";
            echo 'Autre résultat renvoyé avec echo';
        ?>
    </body>
</html>
```

# Afficher un résultat en PHP avec echo ou print

**que faire si le texte que vous souhaitez afficher comporte lui même des guillemets ou des apostrophes ?**

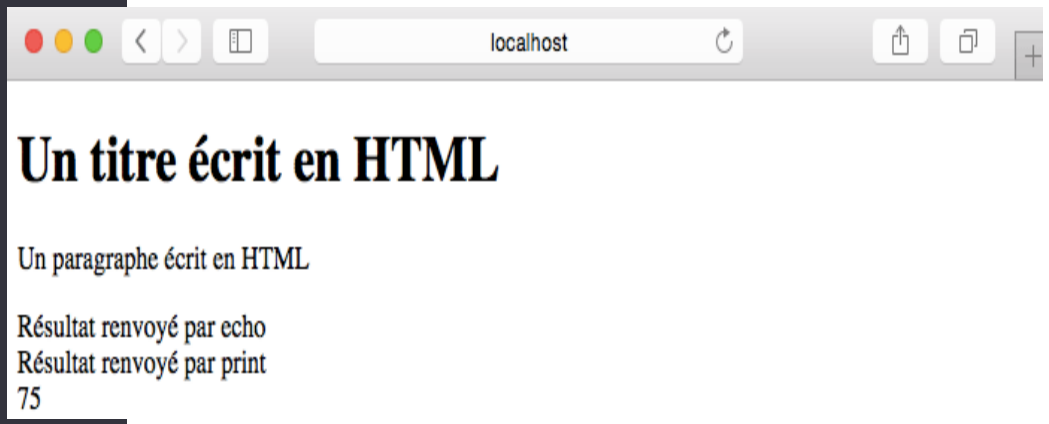
Dans ce cas là, vous devrez absolument échapper les guillemets ou apostrophes (selon ce qui a été choisi pour entourer le texte). Pour échapper des guillemets ou des apostrophes, il suffit de les faire précéder d'un antislash (\).

```
<!DOCTYPE html>
<html>
    <head>
        <title>Démarrer avec PhP</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>Un titre écrit en HTML</h1>
        <p>Un paragraphe écrit en HTML</p>

        <?php
            echo "Un premier résultat avec \"echo\"<br>";
            echo 'Un deuxième résultat avec "echo"<br>';
            echo 'Résultat affiché grâce à l\'instruction echo';
        ?>
    </body>
</html>
```

# Afficher un résultat en PHP avec echo ou print « saute de ligne »

Le navigateur reçoit donc les trois phrases qu'on a **echo** et **print** à la suite et les affiche donc l'une derrière l'autre. Pour remédier à cela, il vous faut savoir que les instructions **echo et print peuvent renvoyer bien d'autres choses que du texte simple. Ainsi, on va également pouvoir passer des nombres ou même des éléments HTML ou la contenu de variables PHP à echo et à print.**

```
<!DOCTYPE html>
<html>
    <head>
        <title>Démarrer avec PhP</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>Un titre écrit en HTML</h1>
        <p>Un paragraphe écrit en HTML</p>

        <?php
            echo "Résultat renvoyé par echo<br>";
            print "Résultat renvoyé par print<br>";
            echo 75;
        ?>
    </body>
</html>
```




```
<? php echo "Ceci est du <strong>  texte  </strong>"; ?>
```

# Les commentaires en PHP

```
<!DOCTYPE html>
<html>
    <head>
        <title>Démarrer avec PhP</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>Un titre écrit en HTML</h1>
        <p>Un paragraphe écrit en HTML</p>

        <?php
            //Un commentaire monoligne

            /*Un
            commentaire
            multilignes*/

            #Syntaxe peu utilisée de commentaire monoligne
        ?>
    </body>
</html>
```

# DECOUVERTE DES VARIABLES PHP

dernier cours

# Les variables

En PHP, les variables ne servent à stocker une information que **temporairement**. Plus précisément, une variable ne va exister que durant le temps de l'exécution du script l'utilisant.

Ainsi, on ne va pas pouvoir stocker d'informations durablement avec les variables (pour cela, nous utiliserons les **fichiers, cookies ou les bases de données** dont nous parlerons plus tard dans ce cours).

# Les variables

Dans PHP, les variables sont précédés par le signe $.

**Exemple :**

`$prenom = "toto";` la variable prenom contient une chaîne de caractère «toto».

`$nombre = 123;` La variable nombre contient un nombre.

`$faux_nombre = "123";`

La variable faux_nombre contient un nombre interprété comme une chaîne de caractères,

c'est-à-dire que l'on ne pourra pas se servir de cette variable pour des opérations mathématiques.

# Les variables

- En PHP, le seul moyen de donner un type à une variable, est d'initialiser cette variable.

**Exemple :**

```
$prenom = "toto";
```

La **variable** prenom est donc initialisée en tant que **chaîne de caractère** et **pointe vers la valeur** toto.

Donc avec une telle déclaration de variable :

```
$X = expression ;
```

On peut dire que la variable $X est du type de l'expression,

Mais ce type peut évoluer au fur et à mesure des affectations successives.

# Les variables

- PHP n'est pas un langage fortement structuré.
- A ce titre, il ne contient pas de partie déclaration clairement définie comme le cas de C ou PASCAL
- PHP n'impose de limites concernant la taille des noms des variables. Cependant, 20-30 caractères suffisant largement
- Les caractères tels que + - * et & ne sont pas autorisés
- En PHP, la casse (majuscule/minuscule) est prise en compte. (X et x sont deux variables différentes),
- Mais pour les fonctions la casse est ignorée.

# Les variables exemple: vars.php

```php
<?php
$auteur="toto1";
$AUTEUR= "TOTO2";
echo " cas minuscule $auteur";
echo "<br>";
echo " cas majuscule $AUTEUR";
echo "<br>";
echo " cas erreur $AUTeur";
?>
```

**Résultats**

cas minuscule toto1
cas majuscule TOTO2
cas erreur

# Types PHP

- booléen TRUE FALSE
- entiers
- flottants
- chaînes de caractères
- tableaux

# Les entiers

- Il est possible de spécifier les nombres entiers de la manière suivante :

```
$a = 1234;   # nombre entier en base 10
$a = -123;   # nombre entier négative
$a = 0123;   # nombre entier en base 8,
             octale (⇔ 83 en base 10)
$a = 0x12;   # nombre entier en base 16,
           hexadécimale (⇔  18 en base 10)
```

# Les nombre en virgule flottante

- Les nombres à virgule flottante ("réels") peuvent êtres spécifié en utilisant la syntaxe suivante:

```
$a = 1.234;
$a = 1.2e3;
```

# Les chaines de caractères ou string

```php
<!DOCTYPE html>
<html>
    <head>
        <title>Démarrer avec PhP</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>Un titre écrit en HTML</h1>
        <p>Un paragraphe écrit en HTML</p>

        <?php
            //$prez stocke une chaine de caractères
            $prez = "Je m'appelle Pierre";

            //$age stocke un nombre
            $age = 25;

            //$age2 stocke une chaine de caractères !
            $age2 = "25";
        ?>
    </body>
</html>
```

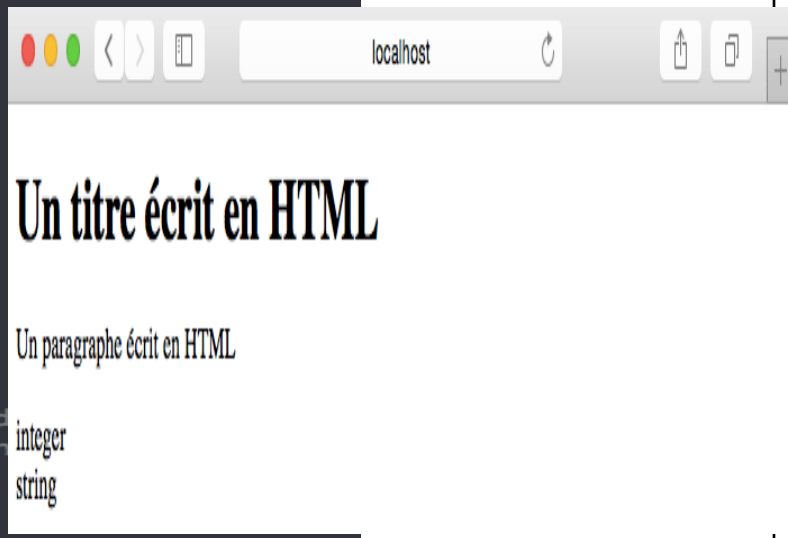
# Le type de valeurs booléen (boolean)

```
<!DOCTYPE html>
<html>
    <head>
        <title>Démarrer avec PhP</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>Un titre écrit en HTML</h1>
        <p>Un paragraphe écrit en HTML</p>

        <?php
            //$bool1 stocke la valeur de type booléen true
            $bool1 = true;

            //$bool2 stocke la valeur de type booléen false
            $bool2 = false;

            //$str1 stocke la chaine de caractères true !
            $str1 = 'true';
        ?>
    </body>
</html>
```

# La fonction gettype()

gettype() va renvoyer en résultat le type de la valeur stockée dans une variable. Nous allons ensuite utiliser une instruction echo pour afficher ce résultat.

```php
<!DOCTYPE html>
<html>
    <head>
        <title>Démarrer avec PhP</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>Un titre écrit en HTML</h1>
        <p>Un paragraphe écrit en HTML</p>

        <?php
            //$prez stocke une chaine de caractères
            $prez = "Je m'appelle Pierre";

            //$age stocke un nombre
            $age = 25;

            //$age2 stocke une chaine de caractères !
            $age2 = "25";

            /*On récupère le type de valeur contenue d
             *grâce à gettype(). On l'affiche avec ech
            echo gettype($age);
            echo "<br>";

            //On fait pareil pour $age2
            echo gettype($age2);
        ?>
    </body>
</html>
```



# La concaténation en PHP

La méthode pour faire cela va être différente selon qu'on utilise des **guillemets** ou des **apostrophes.**
•En utilisant des **guillemets,** il va être très simple par exemple d'afficher un texte et le contenu d'une variable au sein d'une même instruction echo. Voilà comment on va s'y prendre :

```
<!DOCTYPE html>
<html>
    <head>
        <title>Démarrer avec PhP</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>Un titre écrit en HTML</h1>
        <p>Un paragraphe écrit en HTML</p>

        <?php
            $prenom = "Pierre";
            $nom = 'Giraud';

            echo "Je m'appelle $prenom $nom";
        ?>
    </body>
</html>
```

# La concaténation en PHP

•Si l'on utilise des apostrophes, cependant, cela va paraître un peu plus complexe à priori puisque nous allons devoir utiliser l'opérateur de concaténation qui est le point en PHP pour séparer les différentes valeurs au sein de notre echo et plusieurs couples d'apostrophes.

```php
<!DOCTYPE html>
<html>
    <head>
        <title>Démarrer avec PhP</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>Un titre écrit en HTML</h1>
        <p>Un paragraphe écrit en HTML</p>

        <?php
            $prenom = "Pierre";
            $nom = 'Giraud';

            echo 'Mon prénom est ' .$prenom. '. Enchanté !';
        ?>
    </body>
</html>
```

```php
<!DOCTYPE html>
<html>
    <head>
        <title>Démarrer avec PhP</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>Un titre écrit en HTML</h1>
        <p>Un paragraphe écrit en HTML</p>

        <?php
            $prenom = "Pierre";
            $nom = 'Giraud';
            $prez = 'Je suis ' .$prenom. ' ' .$nom;

            echo $prez;
        ?>
    </body>
</html>
```

# Opérateurs arithmétiques

| Exemple | Résultat |
|---|---|
| `$a + $b` | Somme de $a et $b |
| `$a - $b` | Différence de $a et $b |
| `$a * $b` | Produit de $a et $b |
| `$a / $b` | Quotient de $a et $b |
| `$a % $b` | Reste de $a divisé par $b |

# Opérateurs de comparaison

| Exemple | Nom |
|---|---|
| `$a == $b` | Égal |
| `$a != $b` | Différent |
| `$a < $b` | Strictement inférieur |
| `$a <= $b` | Inférieur ou égal |
| `$a > $b` | Strictement supérieur |
| `$a >= $b` | Supérieur ou égal |

# Opérateurs booléens (logique)

| Exemple | Nom |
|---|---|
| `$a and $b, $a && $b` | ET |
| `$a or $b, $a \|\| $b` | OU |
| `$a xor $b` | OU Exclusif |
| `! $a` | NON |

# Opérateurs incrémentaux

| Exemple | Nom |
|---|---|
| `$a++` | Post-incrémente |
| `++$a` | Pré-incrémente |
| `$a--` | Post-décrémente |
| `--$a` | Pré-décrémente |

# Opérateurs d'assignation et autres

| Exemple | Fonction |
|---|---|
| `$a = $b ; $a = 4 ;`<br>`$a = "foo" ;` | Affectation de la valeur de l'expression de droite dans la variable de gauche. |
| `"bonjour "."Monde!";`<br>`$b . $c; $b . "ajout";` | Concaténation de chaînes de caractères. |
| `+=, -=, *=, /=, .=` | Opérateurs d'assignation combinés |

```
<!DOCTYPE html>
<html>
    <head>
        <title>Démarrer avec PhP</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>Un titre écrit en HTML</h1>
        <p>Un paragraphe écrit en HTML</p>

        <?php
            $x = 4;
            echo 'Valeur initiale contenue dans $x : ' .$x. '<br>';

            /*On ajoute 6 à la valeur de $x, puis on réaffecte la
             *nouvelle valeur dans $x*/
            $x += 6; //Equivalent à écrire $x = $x + 6
            echo 'Valeur intermédiaire contenue dans $x : ' .$x. '<br>';

            /*On soustrait 3 à la dernière valeur connue de $x,
             *puis on réaffecte la nouvelle valeur dans $x*/
            $x -= 3; //Equivalent à écrire $x = $x - 3
            echo 'Valeur finale contenue dans $x : ' .$x;
        ?>
    </body>
</html>
```

# Structures de contrôle’’’

# Structures de contrôle

- if/ switch
- while / do .. while
- for

# Syntaxe IF

```
If (Condition)
{
   # Instructions à exécuter si la condition est vraie …
}

else {
   # Instructions à exécuter si la condition est fausse …
   }
```

- Vérifier que les champs ne sont pas vide

```
$name=$_POST["name"];
if ($name=="")
{
echo "Vous devez introduire un nom.<br>";
}
else
{
echo "$name<br>";
}
```

# Choix multiple ( switch )

```
switch (expression)
 {
case exprt1:  {inst1 ;
               break;
              }
case exprt2:  {
             inst2 ;
              break ;
              }
case exprt3:  {
             inst3 ;
             break ;
              }
default : inst4 ;

}
```

# La boucle while

```
while (expr) {
                    instructions
             }
```

**Exemple :**

```
$i = 1;
 while ($i <= 10) {
     print $i++;
   /* La valeur affiché est $i avant l'incrémentation
 (post-incrémentation) */
 }
```

# la boucle do..while

```
$i = 10;
 do
{
 print $i;
    $i-- ;
 } while ($i>0);
```

# La boucle for

```
for (init; cond; iteration)  {
      instruction
   }
```

Exemple :

```
for ($i = 1; $i <= 10; $i++) {
     print $i;
 }
```

# Les Tableaux et les fonctions en PHP

# Les tableaux

- PHP supporte les tableaux scalaires (les indices sont des entiers) et les tableaux associatifs ( les indices sont des chaînes de caractères ).
- `$a[0] = "abc";` // cas scalaires
- `$b["Avril"]= 30;` // cas associatifs

# Les tableaux

```
$tab[0] = “Algiers”;
$tab[1] = ”England”;
$tab[2] = ”Irak";

$tab2[ ] = ‘’Algiers”;
$tab2[ ] = ” England”;
$tab2[ ] = ”Irak”;
$tab2[ ] = ”France”;  // idem que $tab2 [ 3] = ”France”;

$a["color"]   = "red";
$a["taste"]   = "sweet";
$a["shape"]   = "round";
$a["name"]    = "apple";
```

# Exemple : menu.php

```php
<?php
//Utiliser une table pour faire un menu de votre site
$menu[0]["title"]="Home";
$menu[0]["url"]="index.php";
$menu[1]["title"]="Links";
$menu[1]["url"]="link.php";
$menu[2]["title"]="Contact";
$menu[2]["url"]="contact.php";
$menu[3]["title"]="News";
$menu[3]["url"]="news.php";

for ($i=0;$i<4;$i++)
{
echo "<a href=\" ".$menu[$i]["url"]."\">".$menu[$i]
   ["title"]."</a><br/>";
}
```

# Fonctions prédéfinies nécessaires

| Description | Action |
|---|---|
| `int require ( string filename)` | Le fichier est nécessaire pour le script . |
| `int include(string filename)` | Insérer le fichier dans la page php. |
| `isset( mixed var)` | Indique si la variable var existe |

# Fonctions sur les chaînes

| Description | Action |
| --- | --- |
| `int strlen ( string $str)` | Retourne la longueur de la chaîne. |
| `array split ( string $pattern, string $string [, int $limit])` | retourne une tableau de chaînes : chacune d'entre elle est une sous-chaîne de string délimitée l'expression régulière pattern. |
| `Int strpos($chaine, $sous_chaine)` | Retourne la position de la $sous_chaine dans la $chaîne |
| http://www.php.net/manual/fr/function.trim.php | |

# Fonctions sur les tableaux

| Description | Action |
|---|---|
| `int sizeof ( array array)` | retourne le nombre d'élément d'un tableau. |
| `void sort ( array array)` | trie le tableau array dans l'ordre croissant. |
| `mixed array_pop ( array array)` | dépile et retourne le dernier élément du tableau array, le raccourcissant d'un élément. |
| `int array_push ( array array, mixed var [, mixed ...])` | empile les variables passées en paramètres à la fin de array. |
| http://www.php.net/manual/fr/ref.array.php | |

# Fonctions (1)

- Une fonction peut être définie en utilisant la syntaxe suivante :

```
<?php
function nom_fonction ([param_1, …, param_n]) {
        instruction_1;
        …
        instruction_m;
        [return $resultat;]
}
?>
```

# Les Fonctions en PHP (2)

- **function :** mot clé annonçant la définition d'une fonction,
- **Nom_fonction() :** nom de la fonction, servira à son appel,
- **return :** mot clé permettant d'associer un résultat à la fonction.

# Transmettre des données avec l'URL

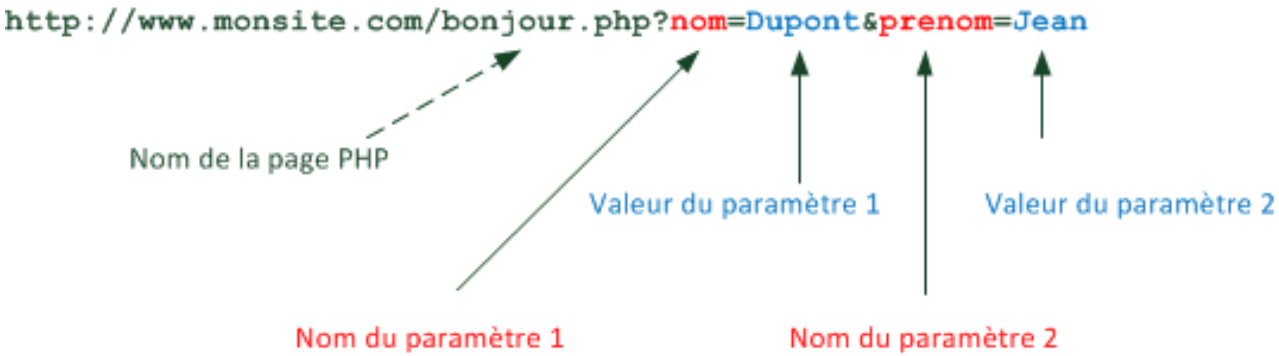
La méthode GET

# Former une URL pour envoyer des paramètres

•Pour accéder à la page **bonjour.php** , aller à l'URL suivante : http://www.monsite.com/bonjour.php

•Pour **envoyer** des informations à la page `bonjour.php`, on va ajouter des informations à la fin de l'URL, comme ceci :

**http://www.monsite.com/bonjour.php?nom=grine&prenom=ali**




•En théorie, Il suffit de les séparer par des **&** comme je l'ai fait. On peut donc voir une URL de la forme :

`page.php?param1=valeur1&param2=valeur2&param3=valeur3&param4=valeur4…`

•Nous voulons faire un lien de `index.php` qui mène à `bonjour.php` et qui lui transmet des informations dans l'URL, comme le schématise la figure suivante.


•Pour cela, ouvrez **index.php** (puisque c'est lui qui contiendra le lien) et insérez-y par exemple le code suivant :

html

```
1 <a href="bonjour.php?nom=Dupont&amp;prenom=Jean">Dis-moi bonjour !</a>
```

# Récupérer les paramètres en PHP

•Bonjour.php est la page qui réceptionne les paramètres. Celle-ci va automatiquement créer un array au nom **$_GET.** Il s'agit d'un atableau associatif dont les clés correspondent aux noms des paramètres envoyés en URL

•On peut donc récupérer ces informations, les traiter, les afficher, à travers le tableau **$_GET.** .

| Nom | Valeur |
| --- | --- |
| `$_GET['nom']` | Dupont |
| `$_GET['prenom']` | Jean |

•**Exemple** : essayons tout simplement de les afficher dans la page bonjour.php pour cela  placez-y le code suivant dans la page bonjour.php

```php
<p>Bonjour <?php echo $_GET['prenom'] . ' ' . $_GET['nom']; ?> !</p>
```

Dis-moi bonjour ! lien dans index.php

Bonjour Jean Dupont ! affichage de la page bonjour.php

# Problèmes ou risque de la méthode GET

Malgré que cette technique est très pratique pour transmettre des valeurs à une page, mais il faut prendre garde au fait que

- le visiteur peut les modifier ou les supprimer très facilement. Il ne faut donc pas faire aveuglément confiance à ces informations, et tester prudemment leur valeur avant de les utiliser.
- on ne peut pas envoyer beaucoup d'informations dans l'URL car il est préférable de ne pas dépasser 256 caractères

# Transmettre des données avec les formulaires

Dans un contexte Web, les données échangées avec le système se font à travers des formulaires

Les formulaires HTML sont la méthode la plus simple pour avoir des interactions avancées avec l'utilisateur

Ils permettent, par exemple, de :

- Créer un espace sécurisé
- Donner aux clients la possibilité de modifier eux-mêmes leurs sites
- Interagir avec le visiteur en lui demandant des informations complémentaires…

# Créer la base du formulaire

```
1 <form method="post" ou" "get"   action="cible.php">
2
3 <p>
4     On insèrera ici les éléments de notre formulaire.
5 </p>
6
7 </form>
```

# La méthode

Il existe plusieurs moyens d'envoyer les données du formulaire .

**GET** : les données transiteront par l'URL (
Après avoir cliqué soumettre, l'URL aura ceci ajouté à la fin )On pourra les récupérer grâce à l'array **$_GET**

**POST** : les données ne transiteront pas par l'URL, l'utilisateur ne les verra donc pas passer dans la barre d'adresse. Cette méthode permet d'envoyer autant de données que l'on veut, ce qui fait qu'on la privilégie le plus souvent.

PHP associe à cette variable  le tableau **$_POST** pour récupérer les données passées

## Création : méthode GET (transmission par URL)

- La différence avec la méthode POST est qu'elle passe les variables à la page web " cible.php" en les ajoutant à la fin de l'URL
- Après avoir cliqué soumettre, l'URL aura ceci ajouté à la fin :
  - "valider.php?objet=xxx&genre=xxx"
- Le point d'interrogation "?" dit au navigateur que les **objets suivants sont des variables**
- **On les récupérera en utilisant le tableau $_GET[]**
- **Dans ce mode de transmission, les variables sont apparentes**
- **Pour votre utilisation c'est égal**

- Création : méthode POST
  - Exemple

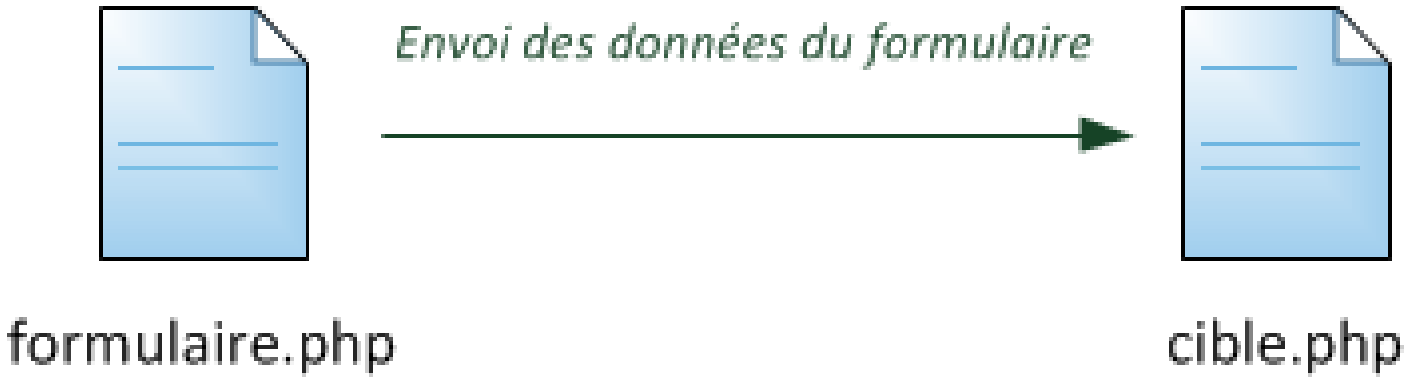
```
<form action=" cible.php" method="post">
  <select name="objet">
  ...
  <input name="genre" type="text" />
  ...
```

  - Ce code HTML spécifie que les données du formulaire seront soumises à la page web " cible.php" en utilisant la **méthode POST**
  - Prenez soin de noter les noms (après name) des données du formulaire, car ils représentent les "clés" dans le tableau associatif "$_POST"
  - Exemple : **$_POST['genre']** permettra de récupérer genre
  - **Ces variables seront cachées pendant l'envoi**

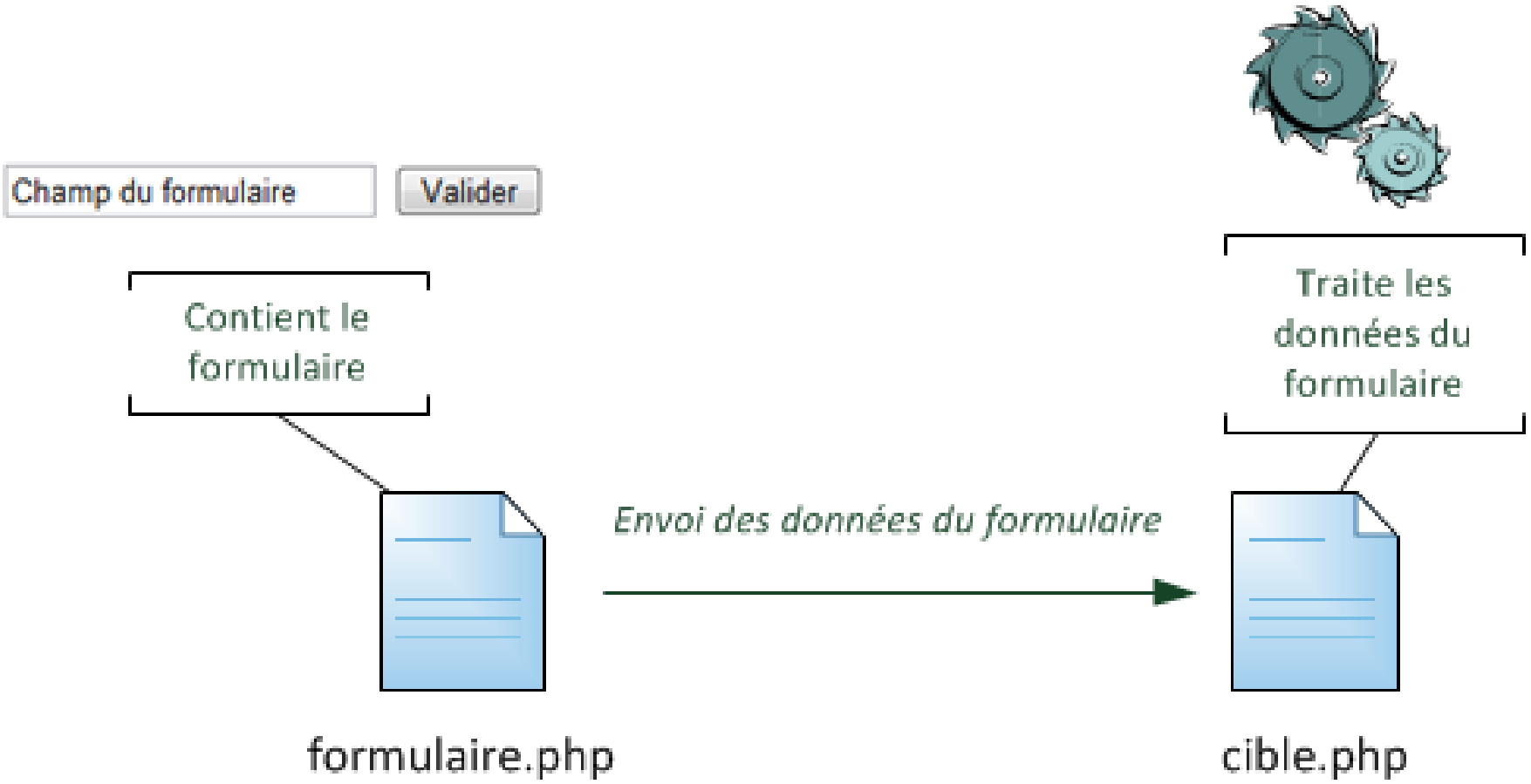
# L’attribut Action

L'attribut **action** sert à définir la page appelée par le formulaire. C'est cette page qui recevra les données du formulaire et qui sera chargée de les traiter.

Imaginons le schéma de la figure suivante.



# Fonctionnement



# Les éléments du formulaire

- Champ de texte

```
<input type="text" name="pseudo" value="M@teo21" />
```

  - 2 attributs
    - **name** : c'est le nom du champ de texte
      - Choisissez-le bien, car c'est lui qui va produire une variable
        - $_POST['pseudo']
    - **value** : permet d'affecter une valeur à la zone de texte
      - Par défaut, le champ de texte est vide

# Les éléments du formulaire

- Champ de texte : tester l'exemple : form-texte1.php

```
<?php
  if (isset($_POST['mon_champ'])) {
?>
  Votre champ contenait :
  <b> <?php echo $_POST['mon_champ']; ?></b>
  <br/><br/>
<?php
}
?>
<form method="POST">
  <input name="mon_champ" type="text"/>
  <input name="valider" type="submit" value="OK"/>
</form>
```

  - La fonction isset() permet de savoir si la variable *$_POST['mon_champ']* existe

# Les éléments du formulaire

- Zone de texte : textarea

```
<textarea name="message" rows="8" cols="45">
        Votre message ici.
</textarea>
```

  - Remarques
    - rows resp. cols détermine le nombre de lignes resp. le nombre de colonnes de la zone de texte
    - Il n'y a pas d'attribut value
    - En fait, le texte par défaut est ici écrit entre le <textarea> et le </textarea>

# Les éléments du formulaire

- Exemple 1 : form-textarea1.php
  - Afficher le contenu de la zone de texte si celle-ci n'est pas vide

```
<?php
$mon_champ = isset($_POST['mon_champ']) ?
   $_POST['mon_champ'] : '';

if ($mon_champ) {
?>
   Votre champ contenait :
   <b><?php echo $mon_champ; ?></b>
   <br/><br/>
<?php
}
?>

<form method="POST">
   <textarea name="mon_champ"></textarea>
   <input type="submit" value="OK"/>
</form>
```

# Les éléments du formulaire

- Les boutons d'options (radio)
  - Permettent de faire des choix

```
Aimez-vous les frites ?
<input type="radio" name="frites" value="oui"
   checked="checked" /> Oui
<input type="radio" name="frites" value="non" /> Non
```

  - On utilise le même nom pour la gestion de tous les boutons, ici : « frites »
  - C'est donc cette variable qui contiendra ce que l'utilisateur a choisi
  - Dans la page cible, une variable $_POST['frites'] sera créée
  - Elle aura la valeur du bouton d'option choisi par le visiteur
  - Si on aime les frites, alors on aura $_POST['frites'] = 'oui'
  - Il faut bien penser à remplir l'attribut "value" du bouton d'option car c'est lui qui va déterminer la valeur de la variable

# Les éléments du formulaire

- Exemple : form-bouton-option1.php

```
<?php
    $mon_champ = isset($_POST['mon_champ']) ? $_POST['mon_champ'] : '';
    if ($mon_champ) {
?>
   Vous avez choisi :
   <b><?php echo $mon_champ; ?></b>
   <br/><br/>
<?php
}
?>
//on utilise la même variable mon_champ
<form method="POST">
   <input type="radio" name="mon_champ" value="Option 1"/>Option 1<br/>
   <input type="radio" name="mon_champ" value="Option 2"/>Option 2<br/>
   <input type="radio" name="mon_champ" value="Option 3"/>Option 3<br/>
   <input type="submit" value="OK"/>
</form>
```

# Les éléments du formulaire

- Les cases à cocher
  - Ici, on fait appel à un tableau pour retenir toutes les cases cochées
  - Exemple : form_case-a-cocher1.php

```
<?php
if (isset($_POST['mon_champ'])) {
    echo "Vous avez choisi :";
    for ($i = 0, $c = count($_POST['mon_champ']); $i < $c; $i++) {
        echo "<br/><b>" . $_POST['mon_champ'][$i] . "</b>";
    }
}
?>
<form method="POST">
    <input type="checkbox" name="mon_champ[]" value="Option
    1"/>Option 1<br>
    <input type="checkbox" name="mon_champ[]" value="Option
    2"/>Option 2<br>
    <input type="checkbox" name="mon_champ[]" value="Option
    3"/>Option 3<br>
    <input type="submit" value="OK">
</form>
```

# Les éléments du formulaire

- Les listes déroulantes à sélection simple
  - Les listes déroulantes sont couramment utilisées pour que l'utilisateur ne puisse sélectionner qu'une valeur

```
<select name="mon_champ">
    <option>Option 1</option>
    <option>Option 2</option>
    <option>Option 3</option>
</select>
```

# Les éléments du formulaire

- Exemple : form-liste-deroulante1.php

```
<?php
$mon_champ = isset($_POST['mon_champ']) ? $_POST['mon_champ'] : '';
if ($mon_champ) {
?>
    Votre champ contenait :
    <b><?php echo $mon_champ; ?></b>
    <br/><br/>
<?php
}
?>
<form method="POST">
    <select name="mon_champ">
        <option>Option 1</option>
        <option>Option 2</option>
        <option>Option 3</option>
    </select>
    <input type="submit" value="OK"/>
</form>
```

# Stocker des informations dans une base de données

# Présentation BDD

- Une base de données est un outil qui stocke vos données de manière organisée et vous permet de les retrouver facilement par la suite.
- MySQL est une base de données implémentant le langage de requête SQL un langage relationnel très connu..
- phpMyAdmin qui permet l'administration aisée des bases de données MySQL avec php.
- Avec MySQL vous pouvez créer plusieurs bases de données sur un serveur. Une base est composée de tables contenant des enregistrements.

# Communiquer au SGBD en langage SQL

## Commande de manipulation des BDD

- Lister toutes les bases de données sur le serveur :

**SHOW DATABASES ;** cette commande va afficher toutes les base de données présentes dans le serveur**.**

- Créer une base de donnée **:**

**CREATE DATABASE nom-base-donnée ;**

- Sélectionner une base de donnée :

USE nom-base-donnée

- Supprimer une base de donnée :

DROP DATABASE nom-base-donnée ;

# Communiquer au SGBD en langage SQL

## Manipulation des table d'une BDD

- ***Création d'une table***

CREATE TABLE **nom-table (champ1 type1, champ2 type2,…) ;**

**Exemple :**

```
CREATE TABLE personne (
   id INT(11) NOT NULL AUTO_INCREMENT,
   nom VARCHAR (100) NOT NULL ,
   age SMALLINT  NOT NULL ,
 PRIMARY KEY  (id)
 ) ;
```

# Communiquer au SGBD en langage SQL

## Visualiser les tables

- Liste des tables contenues dans une base de données

- SHOW TABLES
  - affiche ensemble de toutes les tables contenues dans la base de données.
- DESCRIBE nom-table;
  - permet de lister des champs contenus dans une table.

# Communiquer au SGBD en langage SQL

## Supprimer une table

- **DROP TABLE matable;** Pour supprimer définitivement une table de la base de données
- Exemple

## Insertion d'un enregistrement dans une table

- INSERT INTO nom-table (champ1,champ2,…) VALUES ('valeur1','valeur2',…);
- Pour ajouter un enregistrement dans la table (définie précédemment), vous pouvez utiliser la requête suivante :

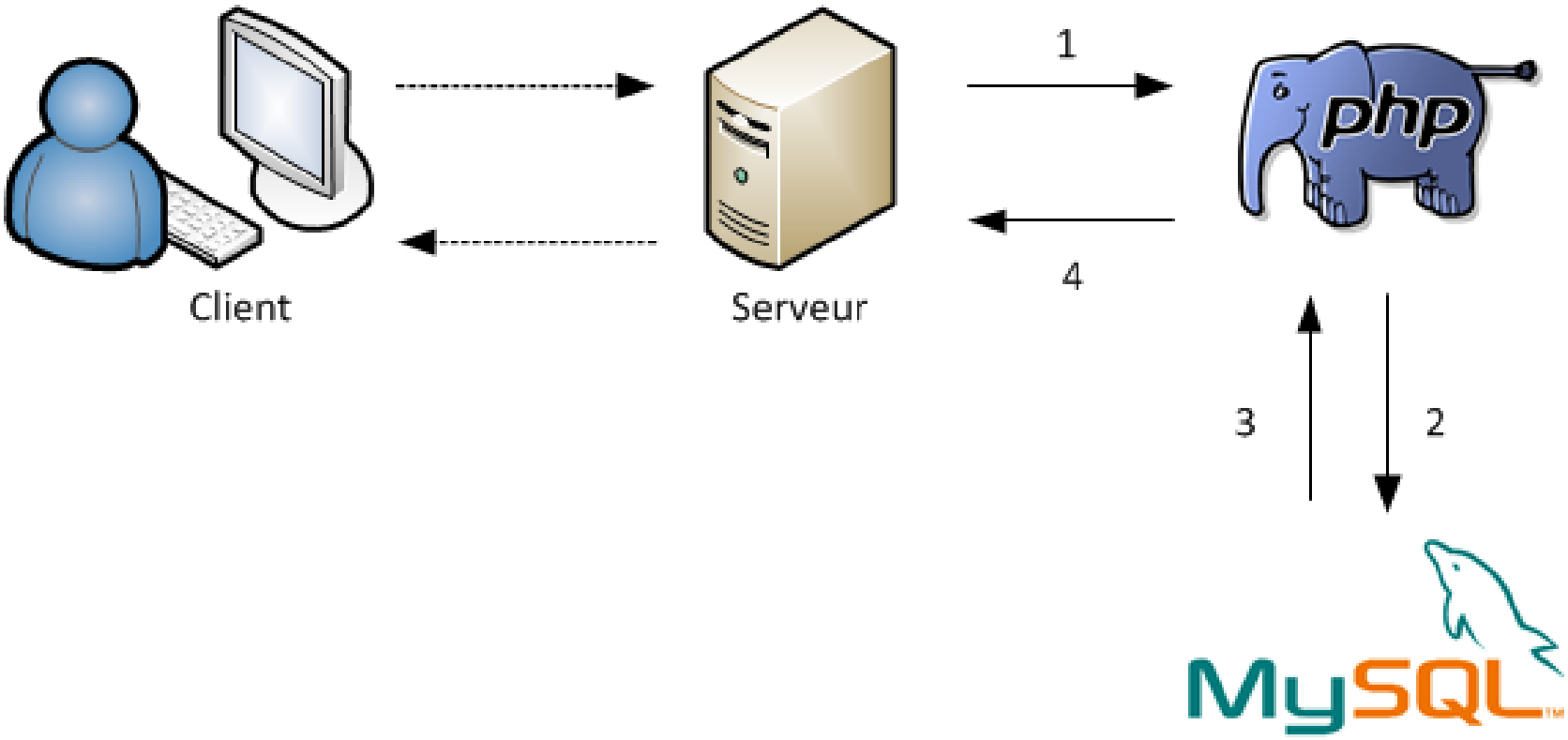
# Communiquer au SGBD en langage SQL

## Sélectionner des données

```
SELECT * FROM matable;
```

- **SELECT name , comment FROM guestbook ;**
- cette requête liste le contenu de la table guestbook, et affiche seulement les colonnes name et comment.

# Interface PHP/MySQL




**PHP fait la jonction entre vous et MySQL**

# Connexion

- **mysql_connect($server,$user,$password)** : permet de se connecter au serveur **$server** en tant qu'utilisateur **$user** avec le mot de passe **$password**, retourne l'identifiant de connexion si succès, FALSE sinon.
- **mysql_select_db($base[,$id])** : permet de choisir la base **$base**, peut prendre un identifiant **$id** de connexion ; retourne TRUE en cas de succès, sinon FALSE.
- **mysql_close([$id])** : permet de fermer la connexion à un serveur de bases de données, l'argument optionnel **$id** est l'identifiant de session retourné à l'ouverture de la connexion.

# Connexion en utilisant mysqli

Pour se connecter à MySQL, on va utiliser cette fonction :

html+php

```php
<?php
$bdd = mysqli_connect('serveur', 'utilisateur', 'mot_de_passe', 'base');
?>

```

- **$bdd** correspond à une variable où seront stockées les informations de la base de données. On s'en servira souvent.
- **serveur** correspond au serveur SQL.
- **utilisateur** correspond au nom d'utilisateur pour se connecter au serveur SQL.
- **mot_de_passe** correspond au mot de passe pour le serveur SQL !
- **base** correspond à votre base de données du serveur SQL.

# Connexion en utilisant mysqli

On peut tester si la connexion a réussi ou non :

html+php

```php
<?php
if($bdd = mysqli_connect('localhost', 'root', '', 'base'))
{
    // Si la connexion a réussi, rien ne se passe.
}
else // Mais si elle rate…
{
    echo 'Erreur'; // On affiche un message d'erreur.
}
?>
```

# Connexion

## Exemple de connexion : viewguest.php

```
<?php
require("config.php");
 $id_connect = mysql_connect($host,$user,$password);
if(!$id_connect)
        die('Echec de connexion au serveur de base de données.');
else {
        if(!mysql_select_db($db))
          die('Echec de connexion à la base.');
       else
       {
          echo 'Succès de connexion.';
          /* code du script … */
        }
      mysql_close($id_connect);
}
```

# Exécuter une requête

- **mysql_query($str):**
- qui prend comme paramètre une chaîne de caractères qui contient la requête écrite en SQL et retourne un identificateur de résultat ou FALSE si échec.

# Exécuter une requête

**Exemple 1 :**

```
$requete= "SELECT *  FROM guestbook  WHERE
   active='No' "  ;
$result = mysql_query($requete);
```

Cet exemple recherche touts les nouveaux visiteurs, c-à-d dont la valeur active='No';

L'identificateur de résultat **$result** permettra à d'autres fonctions d'extraire ligne par ligne les données retournées par le serveur.

# Récupérer le résultat d'une requête

- **mysql_fetch_row($result)** :
  - retourne une ligne de résultat (un tuple) sous la forme d'un tableau. Les éléments du tableau étant les valeurs des attributs de la ligne. Retourne FALSE s'il n'y a plus aucune ligne.

# Exemple :Viewguest.php

```
/* code du script … */
$requet = "SELECT * FROM  guestbook";
if($result = mysql_query($requet)) {

while($ligne = mysql_fetch_row($result))
 {
   $id = $ligne[0];
   $name = $ligne[1];
   $email = $ligne[2];
   $subject=$ligne[3];
   $comment=$ligne[4];
    $active=$ligne[5];
    echo "$id  -  Name : $name,  Email : $email<br>Subject : $subject<br>Active : $active <br> Message : $comment <br />";
    echo "<hr>";
  }
}
 else {
  echo "Erreur de requête de base de données.";
}
```

# Récupérer le résultat d'une requête

- **mysql_fetch_array($result)** :
- retournent un tableau associatif. Les clés étant les noms des attributs et leurs valeurs associées leurs valeurs respectives. Retourne FALSE s'il n'y a plus aucune ligne.

# Mysql_fetch_array : viewguest.php

```
$requet = "SELECT * FROM  guestbook";
if($result = mysql_query($requet)) {
while($ligne = mysql_fetch_array($result))
 {
    $id = $ligne["id"];
   $name = $ligne["name"];
   $email = $ligne["email"];
   $subject=$ligne["subject"];
    $comment=$ligne["comment"];
     $active=$ligne["active"];
    echo "$id  -  Name : $name,  Email : $email<br>Subject : $subject<br>Active : $active <br> Message : $comment <br />";
    echo "<hr>";
  }
} else {
  echo "Erreur de requête de base de données.";
}
```