Développement Web

HTML, CSS, JS, PHP & MySQL

Cours Magistral

Dr Edouard Ngor SARR
Enseignant-Chercheur
UFR SES
Université Assane SECK de Ziguinchor (UASZ)
Ziguinchor-Sénégal
Emai: edouard-ngor.sarr@univ-zig.sn
Site : www.edouardngorsarr.com
Avril 2025

Introduction

Internet & ses services

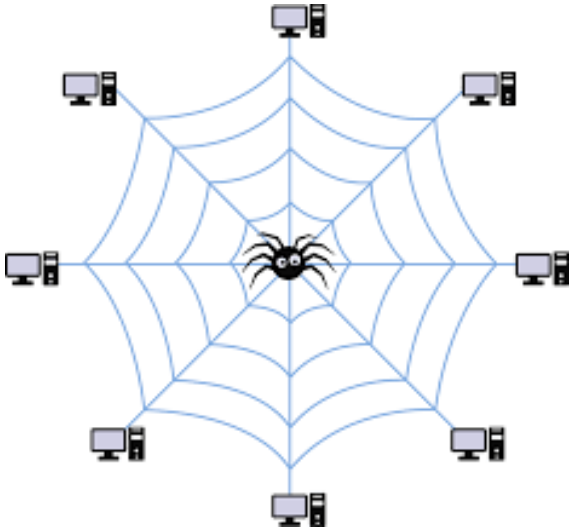
- **Réseau information**
  - Un ensemble d'équipements interconnectés (ordinateurs, serveurs, routeurs, commutateurs, etc.) qui échangent des données selon des règles communes (protocoles).
- **Type de Réseaux**

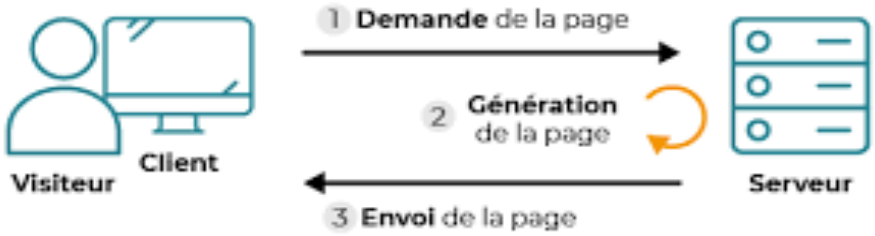
Type	Distance	Débit	Usage typique
LAN	≤ 1 km	100 Mbps - 10 Gbps	Maison, bureau
MAN	5 - 50 km	10 Mbps - 100 Gbps	Ville, campus étendu
WAN	Mondiale	1 Mbps - Tbps	Internet, multinationale

- **Le réseau Internet**
  - Un réseau informatique à la dimension Mondiale : Un WAN
  - Interconnexion des plusieurs réseaux : des LANs et des MANs
  - Finalité : Mettre en contact les utilisateurs et rendre disponible des ressources par le biais de serveurs et de terminaux (Ordinateur, Portable etc).
- **Les services d'Internet**
  - HTTP : web
  - NNTP : newsgroup
  - POP / SMTP : mail
  - FTP : transfert de fichiers
  - DNS : correspondance entre noms et adresses IP
  - SSH : connexion à distance sécurisée

Le service Web

- Un service parmi ceux du réseau Internet
- Service permettant de consulter à distance via un navigateur, des pages web (hypertextes).
- Est basé sur le **protocole HTTP** ou **HTTPs** pour la communication entre serveurs et clients
  - Les ordinateurs distants sur lesquels sont hébergés les ressources web sont des **serveurs**.
  - Les ordinateurs dotés du navigation qui sollicitent ces ressources sont les **clients**
- Un système client-serveur; Le client demande une ressource
  - Le serveur lui donne tel qu'il est stocké : **Processus statique**
  - Le serveur peut générer une réponse selon la demande: **Processus dynamique**





Les Sites web

- **Ensemble de pages web (fichiers)**
  - **Organisation :** Organisées autour d'une page centrale ou d'accueil (index)
  - **Relations :** Liées entre elles par des liens hypertextes (Url)
  - **Disponibilités :** Disponible sur un serveur web via HTTP
  - **Accés :** Accessible en ligne depuis n'importe où navigateurs.
- **Deux types de sites web**
  - **Sites web statiques**
    - Le contenu ne varie pas en fonction des caractéristiques de la demande des utilisateurs
    - Tous les internautes reçoivent le même contenu de la page
    - Un site figé sauf dans certains cas ou il est nécessaire de gérer certains contrôles via le JavaScript (JS)
    - Utilise pas un langage de programmation (PHP, C, …)
    - Souvent coder avec du l'HTML , du CSS et du Java Script
  - **Sites web dynamiques**
    - Le contenu peut varié selon l'utilisateur ou le profil
    - Utilise un langage de programmation (PHP le plus souvent)
    - Souvent coder avec du l'HTML , du CSS, du Java Script et du PHP
    - Utilise aussi souvent une base de données pour stocker les données

Outils necssaires

Editeur de code

Client FTP : Filezilla

FileZilla

- **Serveurs Web, Bases de données et PHP : WAMP SERVER ou XAMP**

WampServer

Apache, MariaDB, MySQL, PHP, PhpMyadmin, MySQL Workbench

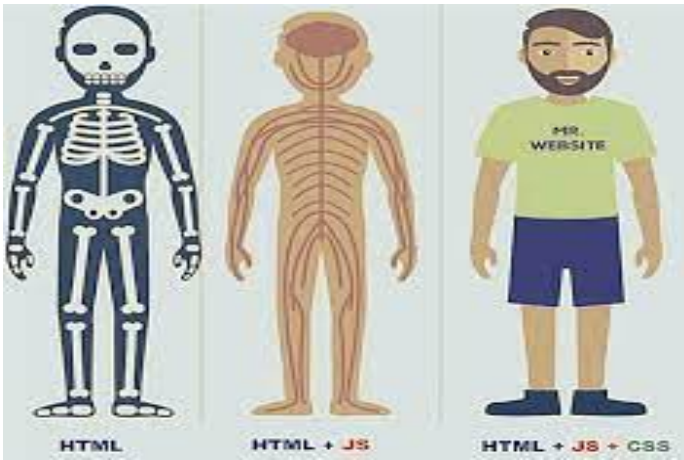
APACHE MariaDB MySQL php phpMyAdmin MySQL workbench

php MySQL

Partie 1
Sites web statiques
HTML, CSS et JS

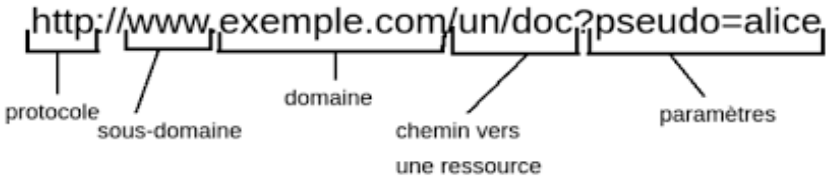
Sites Web Statiques : Définitions

- Un site web comme les autres
- Stocker dans un serveur et accessible par des clients
- Souvent conçu principalement avec du HTML, du CSS et du JS
  - **HTML** : HyperText Markup Language
    - Un langage à balisage / Langage de présentation : *N'est pas un langage de programmation*
    - Décrit la structure logique d'un document hypertexte (page web) : les Emplacement et la présentation des éléments qui composent la page web: Texte, Image, vidéos, Tableaux; Formulaires, Sections et Liens
    - Un fichier HTML (.html) est un format de fichier « texte » éditable dont les éléments ont du sens
  - **CSS**:
    - Rendre plus jolie les pages
    - Gere le Design : Les couleurs, le positionnement et les jeux d'animation
    - Vient compléter le HTML
  - **JavaScript**:
    - Pour rendre un peu dynamique la page mais coté client
    - Contrôle : Champ obligatoire, Activation des boutons …



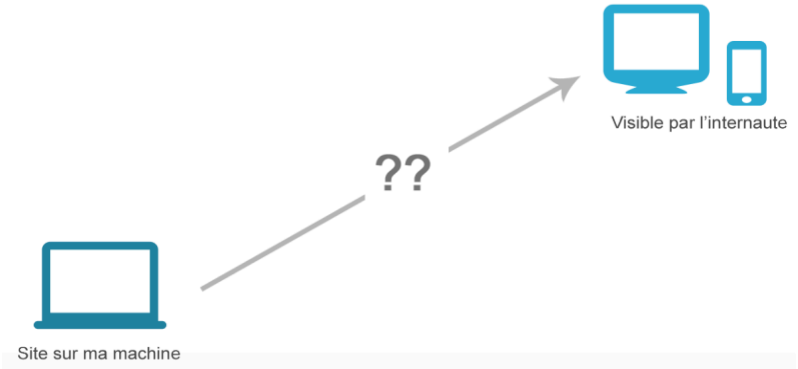
Sites Web Statiques : Autres éléments

- **URL :** Une chaîne de caractères utilisée comme adresse (dans le même site ou en dehors) des ressources dans le Web. Dans le même site
  - Permet de créer des documents interactifs grâce à des liens hypertextes, qui relient votre document à d'autres
  - Exemple:
    - http://www.sarrisgroupe.com
    - https://www.netflix.com/browse

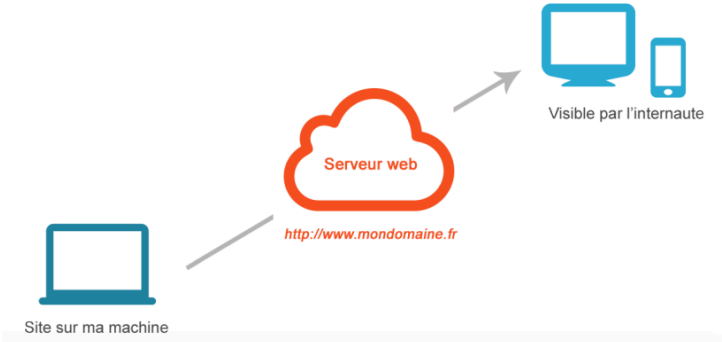


- **Navigateur Web**
  - Logiciel permettant de surfer sur le web et d'afficher sur l'écran les "pages" pointées via l'URL.
  - Comprend le langage du web : HTML, CSS et JavaScript
- **Serveur :** Ordinateur qui offre le service Web : APACHE

En local



En ligne



Le HTML : Les balises

- Les balises structurent le contenu de la page (texte, images, etc.)
- Chaque balise a un rôle et donne du sens au contenu présenté

- Les balises HTML ont la particularité de pouvoir être imbriquées de manière hiérarchique afin de permettre le cumul de leurs propriétés.
- Voici un exemple de texte formaté avec des balises imbriquées

Mon pays, est le Sénégal

***Mon pays**, est le Sénégal* **

- En contrepartie l'exemple ci-dessous n'est pas correct :

**Mon pays, est le Sénégal **

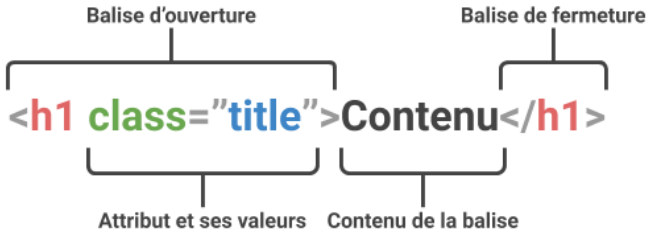
Le HTML : Les balises

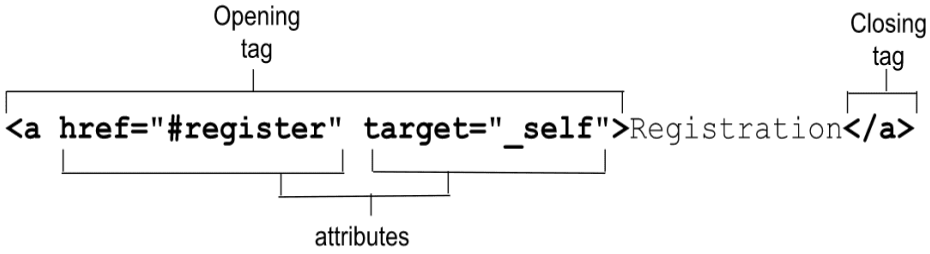
- <head> – Contient les métadonnées
- <title> – Titre de la page
- <body> – Contenu principal affiché
- <h1> à<h6> –Titres (plus grand plus petit)
- <p> – Paragraphe
- <a> – Lien hypertexte
- <img> – Image
- – Liste non ordonnée
- – Liste ordonnée
- – Élément de liste
- – Conteneur générique (structure)
- – Conteneur en ligne
-
 – Saut de ligne
- <hr> – Ligne horizontale
- <form> – Formulaire
- <input> – Champ de saisie
- <label> – Étiquette champ de formulaire
- <button> – Bouton
- <table> – Table de données
- <thead> – En-tête du tableau
- <tbody> – Corps du tableau
- <tfoot> – Pied du tableau
- <tr> – Ligne du tableau (table row)
- <th> – Cellule d'en-tête (table header)
- <td> – Cellule de données (table data)
- <caption> – Légende du tableau
- <nav> – Section de navigation
- <header> – En-tête d'une page ou section
- <footer> – Pied de page ou section
- – Met en valeur du texte (italique par défaut)
- – Texte important (gras)
- <mark> – Surligne du texte
- <script> – Intègre du JavaScript
- …

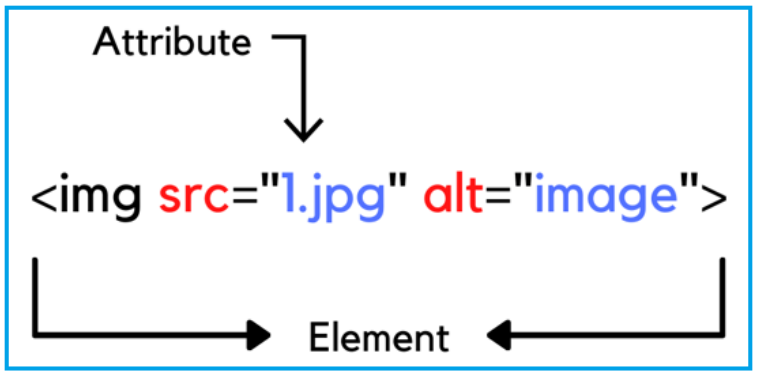
https://www.apprendre-en-ligne.net/javascript/HTML_balises.pdf

Le HTML: Attributs de balises

- Un attribut est un élément, présent au sein de la **balise ouvrante**, permettant de définir des propriétés supplémentaires.
- Les attributs se présentent la plupart du temps comme une paire **clé=valeur**, mais certains attributs ne sont parfois définis que par la clé.
- Chaque balise peut comporter un ou plusieurs attributs
- Chacun de ces attributs pouvant avoir (aucune,) une ou plusieurs valeurs.







Attribut	Valeur	Effet Visuel
ALIGN	LEFT	Texte aligné à gauche
	RIGHT	Texte aligné à droite
	CENTER	Texte centré
	JUSTIFY	Texte justifié

Le HTML: Structure d'un document

- Un document HTML commence par la balise **<HTML>** et finit par la balise **</HTML>**.
- Il contient également
  - **Un en-tête** décrivant le titre de la page qui est délimité par les balises **<HEAD>** et **</HEAD>**.
  - **Un corps** dans lequel se trouve le contenu de la page qui est délimité par les balises **<BODY>** et **</BODY>**.
- **DTD** (Document Type Définition) utilisée : sert à **indiquer au navigateur** quel type de document il doit interpréter.
- Ce n'est **pas une balise HTML**, mais une **instruction au navigateur**.
- En HTML 5 il suffit juste de mettre <!DOCTYPE html>

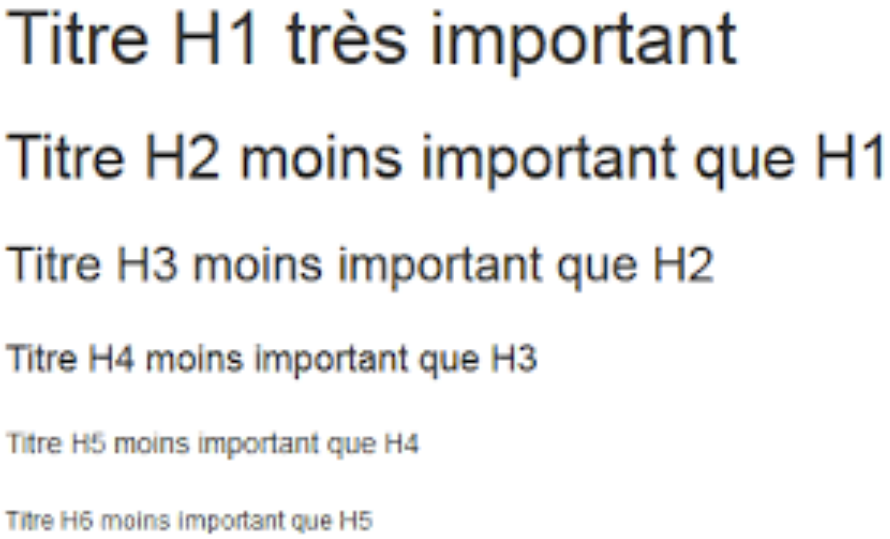
```
<html>
  <head>
    <title>Le titre de la page</title>
  </head>
  <body>
    <h1>Mon premier titre</h1>
    <p>Mon premier <b>paragraphe</b></p>
  </body>
</html>
```

Le HTML: Commentaires et titres

- **Commentaires**
  - Présentent dans une page web sans que ceux-ci soient affichés à l'écran donc Non prisent en compte par le Navigateur
  - Les commentaires doivent être précédés de la balise <!-- et être fermés par -->
  - Tout ce qui sera écrit entre ces balises Ne sera pas affiché à l'écran par le navigateur.

```
<!-- Voici un commentaire -->
```

- **Les titres Hx**
  - Permet de définir une structuration hiérarchique des paragraphes dans un texte
  - 6 niveaux de titre (en anglais heading) : H1, H2, H3, H4, H5 et H6



Le HTML: Les listes

- **Les listes**
  - Une liste est un paragraphe structuré composé d'une suite d'articles.
  - Le langage HTML définit deux types de listes :

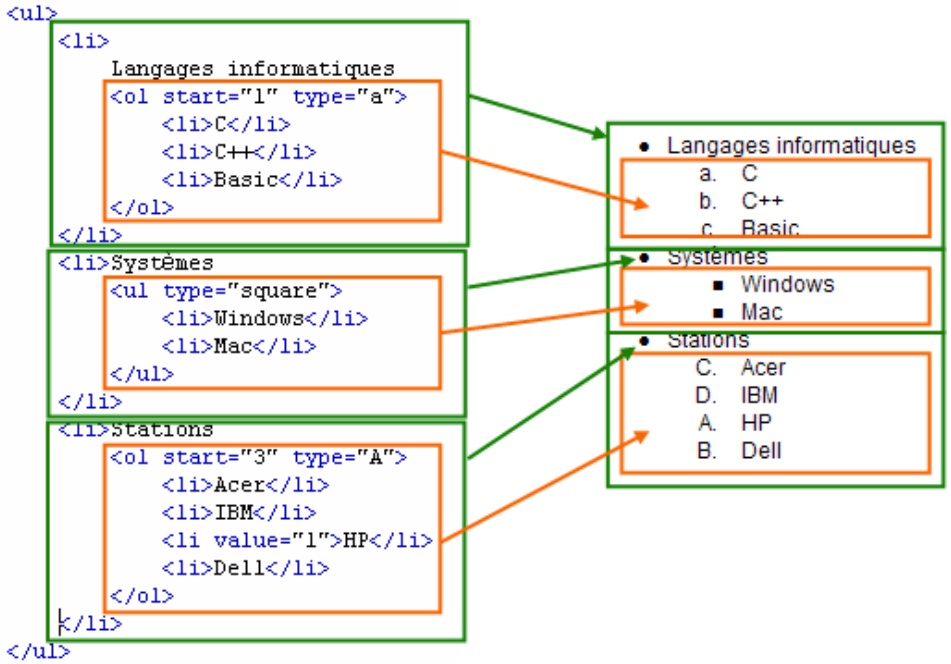
La liste ordonnée avec OL

```
<ol>
    <li> article 1 </li>
    <li> article 2 </li>
</ol>
```

La liste non ordonnée avec UL

```
<ul>
    <li> article 1 </li>
    <li> article 2 </li>
</ul>
```

Attribut	Valeur
COMPACT	resserre l'interligne
PLAIN	supprime les puces



Le HTML: Les Tableaux

- Les tableau
- Souvent utile de présenter des informations mieux structurées qu'avec des listes.
- Permettent de les afficher en lignes et en colonnes.
- Doit respecter les quelques règles suivantes :
  - Le titre du tableau est encadré par *<CAPTION></CAPTION>*
  - Chaque ligne est encadrée par *<TR></TR>*
    - Les cellules d'en-tête sont encadrées par *<TH></TH>*
    - Les cellules de valeur sont encadrées par *<TD></TD>*


```
<table>
<tr>
<td> A</td>
<td> B </td>
<td> C </td>
</tr>
<tr>
<td> D</td>
<td> E </td>
<td> F </td>
</tr>
</table>
```

A	B	C
D	E	F

Le HTML: Les liens hypertextes et les images

- **Les ancrages**
  - Permettent de lier des pages Web entre elles et de naviguer vers un autre endroit du document, vers un fichier HTML situé à un emplacement différent sur la machine qui héberge la page ou vers fichier se trouvant sur une autre machine
  - L'attribut principal des ancrages est **href**. Il s'écrit sous la forme suivante :

** CLIQUER ICI **

- **Les images**
  - La balise **IMG** du langage HTML permet d'insérer des images dans une page HTML.
    - L'image peut être située sur le même serveur que la page dans laquelle elle est insérée mais également sur un autre serveur en spécifiant son URL complète.
  - Les principaux attributs de la balise IMG sont les suivants :
    - SRC: Indique l'emplacement de l'image (il est obligatoire)
    - ALT: Permet d'afficher un texte alternatif lorsque l'image ne s'affiche pas.
    - TITLE: Permet d'afficher une infobulle lors du survol de l'image par le curseur.
    - WIDTH: Permet de spécifier la largeur de l'image.
    - HEIGHT: Permet de spécifier la hauteur de l'image.

<IMG SRC="img/1.jpg" ALT= "Image non trouvée" TITLE= "Ma Photo">

Le HTML: Les Meta TAGS

- Utilisées par les moteurs de recherche lors du référencement de la page web.
- Renseigner des informations relatives à la page HTML où au site
- On distingue deux types de méta tags :
  - Les métas **NAME**, permettant de décrire la page HTML
  - Les métas **HTTP-EQUIV**, permettant d'envoyer des informations supplémentaires au navigateur via le protocole HTTP :
- **Définir le jeu de caractères**

```
<meta charset="UTF-8">
```

- **Description de la page (SEO)**

```
<meta name="description" content="Ceci est le cours HTML de Dr Ngor SARR.">
```

- **Mots-clés (moins utilisé aujourd'hui)**

```
<meta name="keywords" content="HTML, meta, référencement, SEO, balises">
```

- **Auteur de la page**

```
<meta name="author" content="Edouard">
```

- **Contrôle du comportement du navigateur (ex : mise à jour automatique)**

```
<meta http-equiv="refresh" content="30">
```

- **Affichage sur mobile (Responsive Design)**

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Le HTML: FRAME (Les Cadres)

- Grâce à la technologie des frames (en français "cadres") il est désormais possible d'afficher plusieurs pages HTML dans différentes zones (ou cadres).
- A pour particularité d'avoir un conteneur <FRAMESET> à la place du jeu de balises <BODY>.
- Définit les cadres par leur dimension en pixels ou en pourcentage (%).
- Voyons ceci sur 3 exemples:

02 cadres verticaux

```
<FRAMESET COLS="20%,80%">
    <FRAME SRC="" NAME="gauche">
    <FRAME SRC="" NAME="droite">
</FRAMESET>
```

02 Cadres horizontaux

```
<FRAMESET ROWS="20%,80%">
        <FRAME SRC="" NAME="haut">
        <FRAME SRC="" NAME="bas">
</FRAMESET>
```

02 cadres horizontaux et un vertical

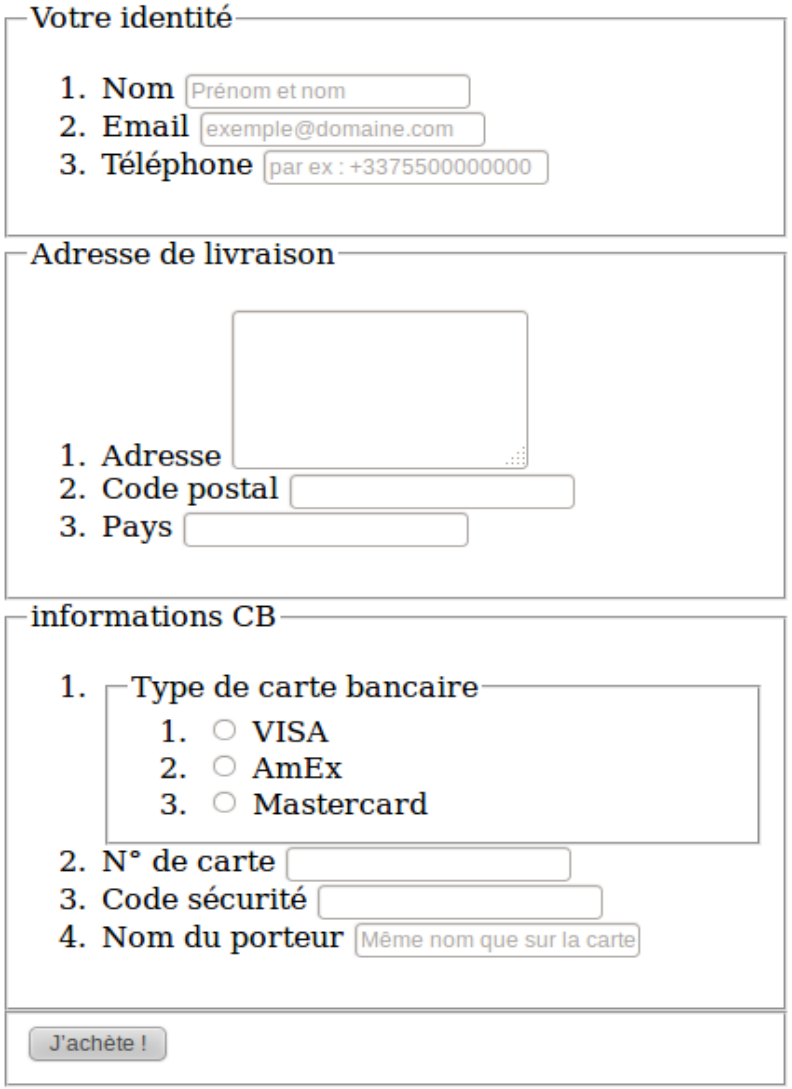
```
<FRAMESET COLS="20%,80%">
        <FRAME SRC="" NAME="gauche">
        <FRAMESET ROWS="50%, 50%">
            <FRAME SRC=""  NAME="droit_haut">
            <FRAME SRC=""  NAME="droit_bas">
</FRAMESET>
```

Le HTML: Les formulaires

- **FORM**
- Le formulaire sert à récupérer des données que l'utilisateur va entrer et les envoyer au serveur pour un traitement (en PHP par exemple)
- Sont utiles dès lors que l'on souhaite une

```
<form method="post" action="traitement.php">
```

- **METHOD** : 02 modes d'envoie des données
  - »GET" : limitée à 255 caractères, informations passées dans la barre d'adress
  - »POST" : envoie les données dans le corps de la requête sans passer par la barre d'adresse. C'est la méthode la plus utilisée
- **ACTION**: indique l'adresse du fichier ou programme qui va traiter les données



Le HTML: Les formulaires

- **INPUT**
  - **Permet de définir un champ de saisie**
  - Attributs :
    - **Id**: identifiant pour le Javascrpit
    - **Name:** le nom du champ
    - **Type:** le type des valeurs possible
      - **Number**: que des chiffre
      - **Text**: alphanumerique
      - **Password**: text masqué avec des ****
    - **value=" " :** donne une valeur par défaut au champ
    - **maxlength**: le nombre de caractères maximum acceptés
    - **placeholder**: donne une indication de ce que devrait contenir le champ
    - **Require**: Rendre le champ obligatoire
    - **ReadOnly**: permet de griser le champ
    - **disabled**="disabled » pour désactiver le champ

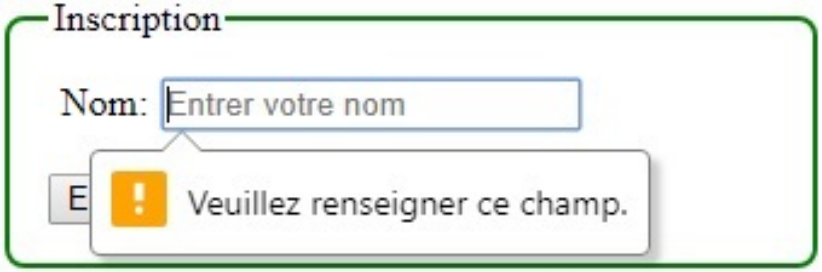
formulaire : exemple simple

form.html simplifié...

```
<form method="get" action="search.php">
<p>
 1 <input type="text" name="terms" />
 2 <input type="submit" value="rechercher" />
</p>
</form>
```

1 2

rechercher




```
<p>
   <label for="nom">Votre nom</label>
   <input value="Stéphanie" type="text" id="nom" name="nom"/>
</p>
<p>
   <label for="prenom">Votre Prénom</label>
   <input placeholder="ex : stéphanie" type="text" id="prenom" name="prenom" />
</p>
```

Votre nom stéphanie

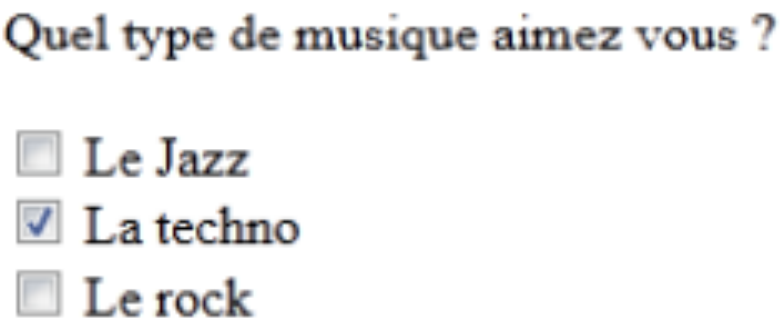
Votre Prénom ex: stéphanie

Le HTML: Les formulaires

- **TEXTAREA**
  - **<u>Permet de définir une zone de texte</u>**
  - Attributs
    - **Id:** identifiant pour le Javascrpit
    - **Name:** le nom du champ
    - **Require:** Rendre le champ obligatoire

Le HTML: Les formulaires

- **CHECKBOX**
  - **Permet de définir des elements à cocher**
  - **• Plusieurs cases, toutes peuvent être cochées**
  - Attributs
    - Id: identifiant pour le Javascrpit
    - Name: le nom du champ
    - Require: Rendre le champ obligatoire
    - Checked: choisir par defaut

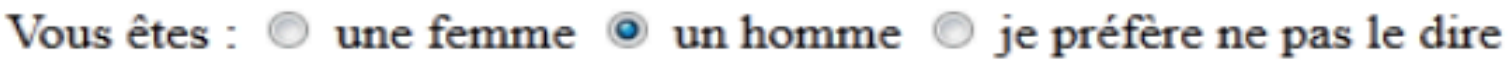



```
<p>
    <input type="checkbox" name="jazz" id="jazz">
    <label for="jazz">Le Jazz</label>
</p>
<p>
    <input type="checkbox" name="techno" id="techno" checked="checked">
    <label for="techno">La techno</label>
</p>
<p>
    <input type="checkbox" name="rock" id="rock">
    <label for="rock">Le rock</label>
</p>
```

Le HTML: Les formulaires

- **INPUT RADIO**
  - **Permet de définir des elements à cocher**
  - Un seul choix possible parmi tous les éléments
  - C'est INPUT de Type Radio

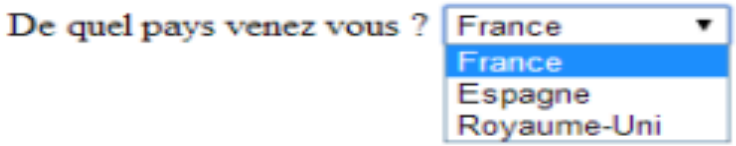
```
<p> Vous êtes :
    <input type="radio" name="sexe" value="femme" id="femme"/><
    label for="femme"> une femme </label>
    <input type="radio" name="sexe" value="homme" id="homme"
    checked="checked"/><label for="homme"> un homme </label>
    <input type="radio" name="sexe" value="nsp" id="nsp"/><label
    for="nsp"> je préfère ne pas le dire </label>
</p>
```



Le HTML: Les formulaires

- **SELECT**
  - **Permet de définir des éléments à choisir sur une liste**
  - Un seul choix possible parmi tous les éléments

```
<p>
    <label for="pays">De quel pays venez vous ?</label>
    <select name="pays" id="pays">
        <option value="france">France</option>
        <option value="espagne">Espagne</option>
        <option value="royaume-uni">Royaume-Uni</option>
    </select>
</p>
```



Le HTML: Les formulaires

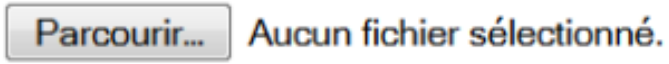
- **INPUT SUBMIT**
  - **Permet d'envoyer un formulaire**
  - L'attribut value=" " est obligatoire et permet de donner le « titre » du bouton
  - Le formulaire est envoyé pour traitement au fichier que l'on a défini dans action=" » du FORM
- **INPUT RESET**
  - **Permet d'annuler l'envoie d'un formulaire**
  - C'est un bouton
  - Pour remettre à zéro toutes les données du formulaire

```
<p>
    <button type="button">Cliquez moi</button>
    <button type="submit">Envoyer</button>
    <button type="reset">Reset</button>
</p>
```

Le HTML: Les formulaires

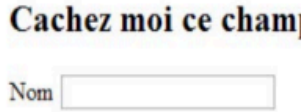
- **INPUT FILE**
  - **Permet de charger un fichier**
  - Pour télécharger vers le serveur un fichier on utilise input type="file"
  - Si on envoie un fichier, il faut permettre au formulaire d'envoyer des données avec l'attribut enctype="multipart/form-data" sur l'élément form.

```
<input name= "myfile" type="file" />
```

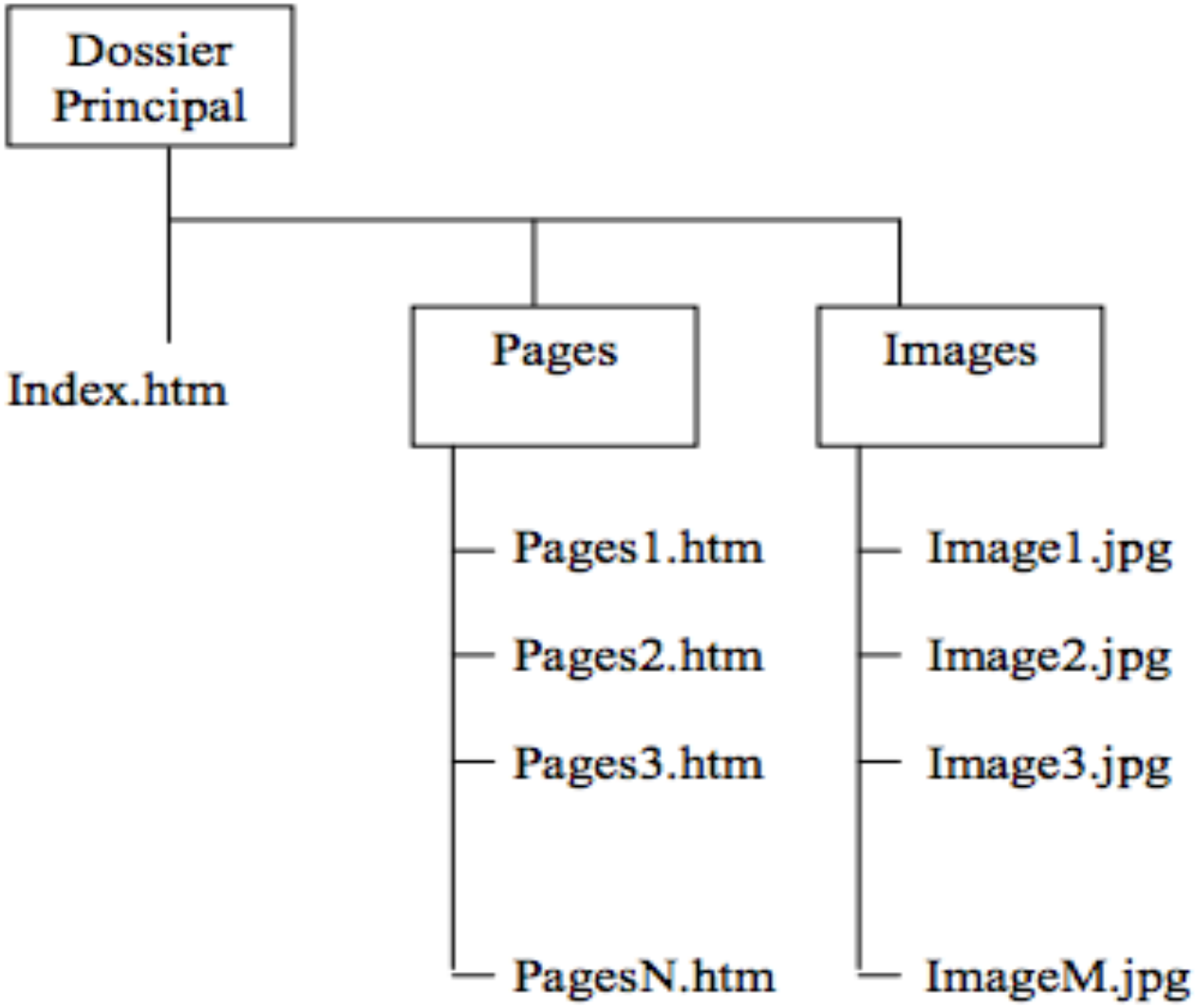


- **INPUT HIDDEN**
  - On peut cacher un champ avec input type="hidden"
  - Permet d'envoyer des données sans que l'utilisateur ne doive les remplir ou ne les voie.

```
<h2> Cachez moi ce champ ! </h2>
<input type="hidden" name="key" value="masupercleesecrète" />
<label for="nom">Nom</label><input type="text" name="nom" id="nom"/>
```



Le HTML : Organisation d'un site web Statique



Le HTML: Exercice d'application

Pique-nique Multi-associations

Inscriptions

Prénom :

Mot de passe :

Association : AssociationX

Disponibilités pour la semaine du 22 juin : Lundi Mardi Mercredi Jeudi Vendredi

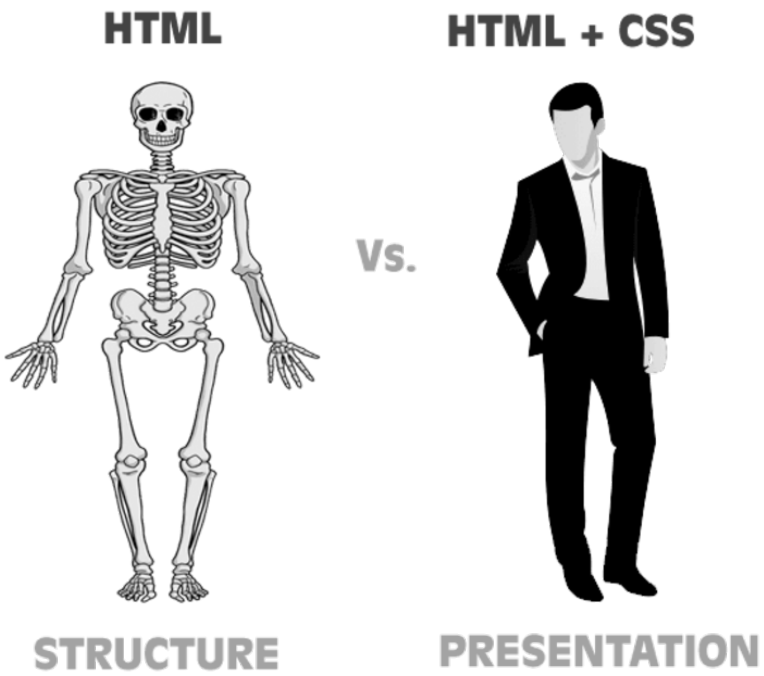
Contribution : Entrée Plat Dessert

Commentaires :

S'inscrire ! Annuler

CSS: Définitions

- CSS : euille de style en cascade
- Ensemble de règles stylistiques applicables à un, ou plusieurs documents HTML
- Facilite la mise à jour graphique, favorise l'accessibilité
- Permet de Gérer des différents médias possible (print, screen, mobile etc.)
- Permet de Gérer : Couleur de texte, image de fond, style de police, menu à gauche ou à droite
- **Permet de séparer la structure (HTML) de sa présentation (CSS)**



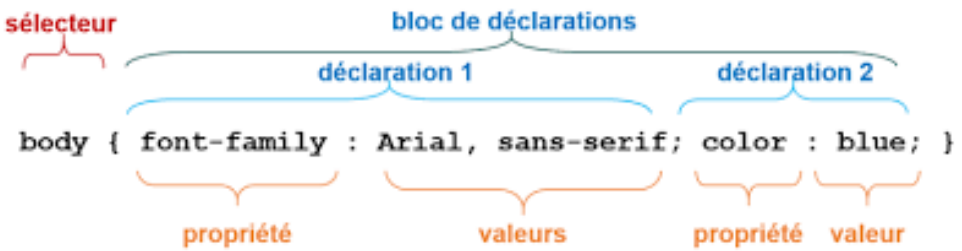
Le CSS "en ligne" dans la balise HTML

- Attribut style = …;
- « Je veux des titres roses » :

```
<h1 style="color:pink;" >
```

- Et des sous titres avec une couleur de fond violette

```
<h2 style="background-color:purple">
```



Problèmes de maintenabilité

Le CSS « interne» dans l'entête du HTML

- On place une balise <style> dans la balise <head> du document
- Changer la couleur de tous les titres H2 en blanc :

```
<style type="text/css">

h2{
  color: #fff;
}

</style>
```

Problèmes de maintenabilité

La/les feuille(s) de style externe(s)

index.html

```
<!DOCTYPE HTML>
<html lang="fr-FR">
<head>
    <meta charset="UTF-8">
    <title>Mon premier CSS</title>

    <link rel="stylesheet" type="text/css" href="styles.css" />
```

styles.css

```
1  p{
2      background:yellow;
3  }
```

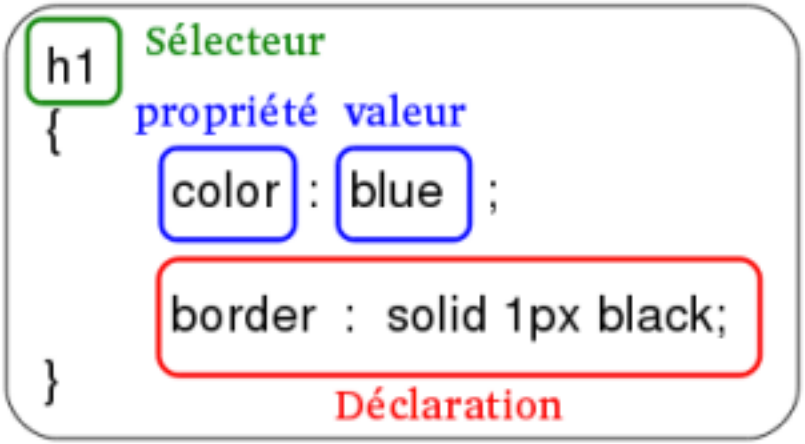
styles.css

Maintenance très aisée et rapide

CSS: Syntaxe

- La syntaxe est composée de 3 éléments :
  - Le **sélecteur** est l'élément sur lequel on applique les propriétés (balise HTML, id, classe, etc.)
  - La **propriété** est l'effet que l'on va vouloir donner (ex couleur de texte, positionnement, couleur de fond, etc.)
  - La **valeur** de la propriété CSS (rouge, 10px, etc.)

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



CSS: Les propriétés

- CSS : Feuille de style en cascade
- Ensemble de règles stylistiques applicables à un, ou plusieurs documents HTML
- Permet de séparer la structure (HTML) de sa présentation (CSS)
- Facilite la mise à jour graphique, favorise l'accessibilité
- Permet de Gérer des différents médias possible (print, screen, mobile etc.)
- Permet de Gérer : Couleur de texte, image de fond, style de police, menu à gauche ou à droite : c'est CSS qui va contrôler tout ça

- **color :** Définit la couleur du texte. Accepte des noms de couleurs, des codes hexadécimaux, RGB, ou HSL.
- **background-color:** Modifie la couleur d'arrière-plan d'un élément.
- **font-size:** Contrôle la taille de la police. Peut utiliser des unités comme px, em, rem, ou %.
- **Margin:** Gère l'espace extérieur autour d'un élément, séparant ce dernier des autres éléments.
- **Padding:** Détermine l'espace intérieur d'un élément, entre son contenu et sa bordure.
- un survol).

- **Border:** Ajoute une bordure autour d'un élément, avec contrôle sur son épaisseur, son style et sa couleur.
- **Display:** Change le mode d'affichage d'un élément (block, inline, flex, grid, etc.).
- **position :** Permet de positionner un élément de manière relative, absolue, fixe ou collante (sticky).
- **width / height :** Définit les dimensions d'un élément en largeur et hauteur.
- **Transition** Ajoute des effets d'animation fluides lors des changements d'état (comme un survol).

https://projet.eu.org/pedago/sin/1ere/4-css3_liste_proprietes.pdf

https://www.web-eau.net/blog/10-proprietes-css3-a-connaitre-et-a-maitriser

CSS: Syntaxe

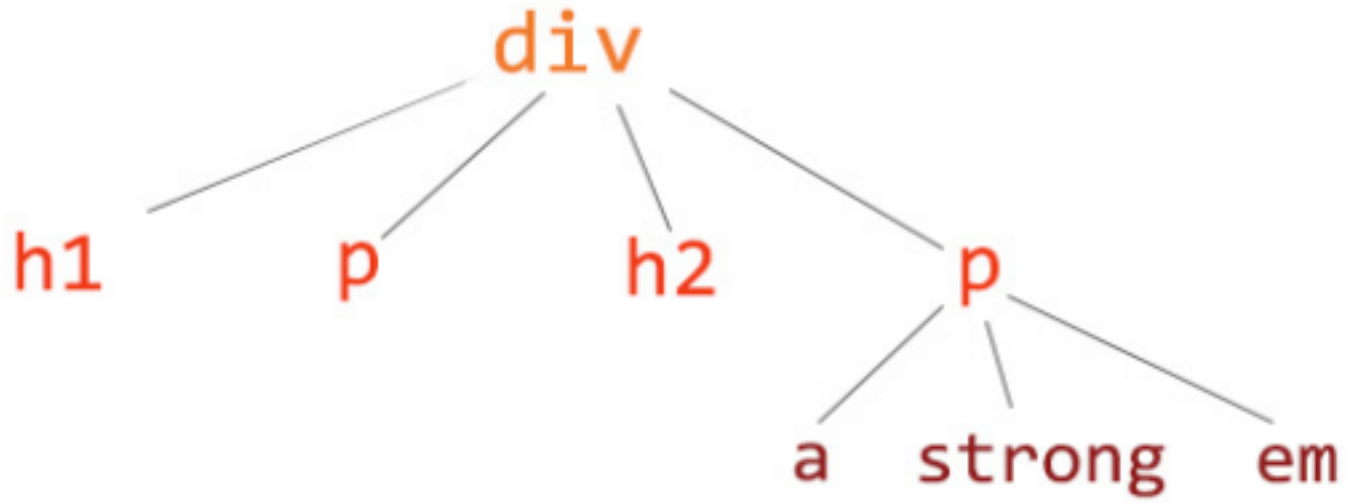
- La syntaxe est composée de 3 éléments :
  - Le **sélecteur** est l'élément sur lequel on applique les propriétés (balise HTML, id, classe, etc.)
  - La **propriété** est l'effet que l'on va vouloir donner (ex couleur de texte, positionnement, couleur de fond, etc.)
  - La **valeur** de la propriété CSS (rouge, 10px, etc.)

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

Pour créer un commentaire CSS, on utilise

/* commentaire */

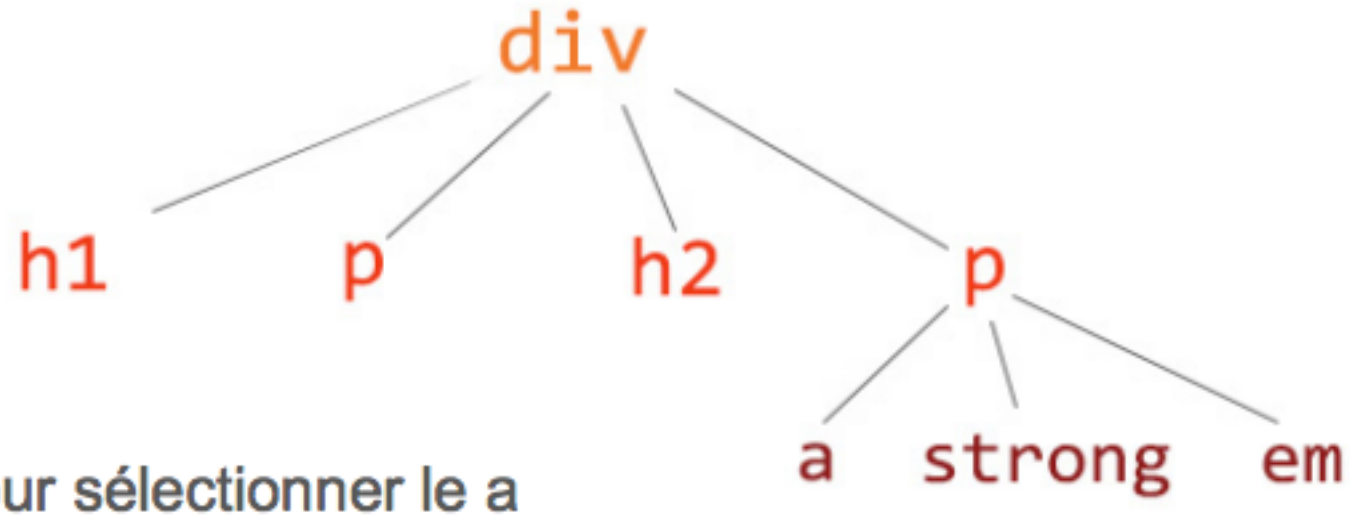
CSS: Généalogie



- h1, p, h2, p sont enfants de div
- a, strong et em sont enfant du p dans lequel ils sont contenus (mais pas de l'autre p)
- a, strong et em (et h1, p, h2 et p) sont descendants de div

- div est **parent** de h1 p, h2, p
- Le 2ème p est **parent** de a, strong et em.
- div est **ancêtre** de a, strong et em (et de h1, p h2 et p)

CSS: Sélecteur hiérarchique



- Pour sélectionner le a descendant de p, nous allons pouvoir écrire :

```
p a { ... }
```

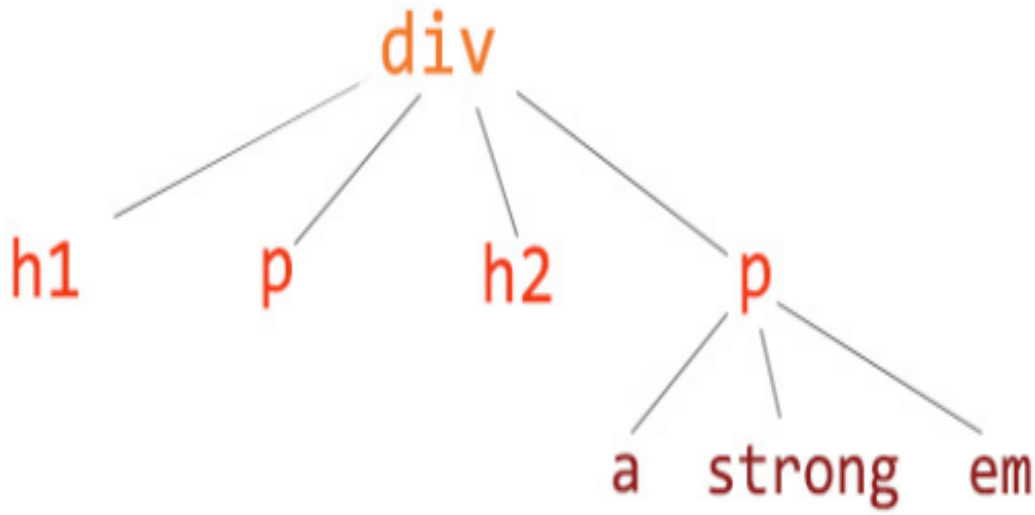
(notez l'espace entre le p et le a)

CSS: Sélecteurs de groupe et de classe

- Pour sélectionner plusieurs éléments et leur appliquer la même valeur, on les sépare par une virgule.

```
h1, h2 { color: red; }
```

=> Va donner la couleur rouge à tous les h1 ou h2



- class=" " est un attribut qui peut se mettre sur n'importe quelle balise

Il permet de cibler de manière plus « spécifique » certains éléments

```
<p class="important"> Un paragraphe spécifique qui est mis en avant </p>
```

CSS: Sélecteur de classes multiples

- Il est possible d'avoir plusieurs classes sur un élément, on les sépare d'un espace.

```
<p class="remarque important"> Une remarque mis en avant </p>
```

- Une même classe peut être utilisée sur plusieurs balises dans le même document

```
<p class="remarque"> une remarque paragraphe</p>

<blockquote class="remarque "> une remarque citation</blockquote>
```

CSS: Ciblage d'une balise HTML

- Pour cibler un élément doté d'une classe en CSS on procède de la manière suivante (notez qu'il n'y a pas d'espace)

  element.maclasse { }

- Par exemple : **p.toto { }** va cibler tous les éléments comme celui-ci :

  ** … **

- Si on veut cibler une classe, sans se préoccuper de la balise on utilise le « point ».

  .nomdeclasse { }

  - Par exemple **.remarque { }** va cibler

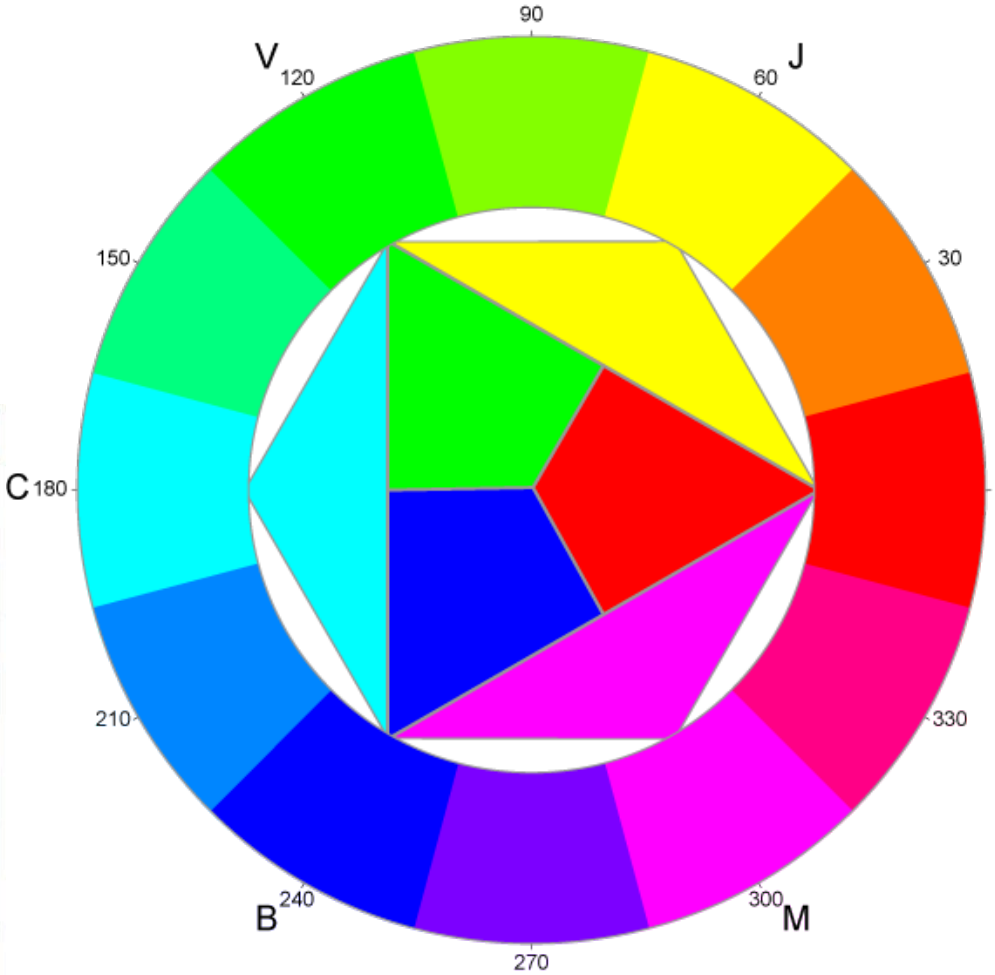
     …

    <blockquote class="remarque"> …. </blockquote>

CSS: Les Couleurs

2 digits pour le Rouge
2 digits pour le Vert
2 digits pour le Bleu

#A0342F



CSS: Résumé

CSS	HTML ciblé
p { }	....
.maclasse { }	<element class="maclasse"> ... </element>
#monid { }	<element id="monid"> ... </element>
p.maclasse { }	...
p .maclasse { }	<element class="maclasse"> .... </element>
p, div { }	... ...

JavaScript : Définitions

- Historiquement permet de programmer des interactions au sein des navigateurs
  - Interagir : savoir qu'un bouton a été cliqué○
  - Afficher : manipuler la page web pour rendre visible des nouvelles parties
  - Communiquer : envoyer ou recevoir des requêtes
- C'est le seul langage disponible côté navigateur
- Mais est aussi disponible côté serveur via NodeJS
- Il n'y a pas de "main method »
  - Le script est exécuté de haut en basIl n'y a pas de compilation par le développeur
- JavaScript est compilé et exécuté à la volée par le navigateur

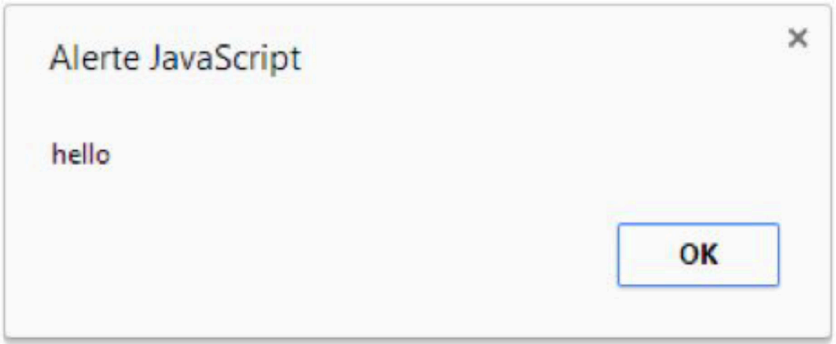
```
<script>
  instruction_1; // Ceci est ma première instruction
  instruction_2;
        /* La troisième instruction ci-dessous,
        avec un commentaire sur deux lignes */
  instruction_3;
</script>
```

JavaScript : HTML & JS

```
<!doctype html>
<html lang="fr">
  <head>
      <title>Example</title>
      <script>
        var msg = "hello";
        alert(msg);
      </script>
  </head>
  <body>
  </body>
</html>
```

JavaScript : HTML & JS

```html
<!doctype html>
<html lang="fr">
  <head>
    <title>Example</title>
    <script src="script.js"></script>
  </head>
  <body>
  </body>
</html>
```



script.js

```js
console.log('Hello, world!');
```

JavaScript : Variables, Constantes et opérateurs

Il y a plusieurs types primitifs:

- **Boolean** : true et false
- **Number** : tout est de type double (pas d'entiers)
- **String**: avec 'single' ou "double-quotes"
- **Null**: null une valeur qui signifie "ceci n'a pas de valeur"
- **Undefined**: la valeur d'une variable non affectée

Trois façons de déclarer des variables en JS

```
// Function scope variable
var x = 15;
// Block scope variable
let fruit = 'banana';
// Block scope constant; cannot be reassigned
const isHungry = true;
```

Le langage est dynamiquement typé

`==` : égal à

`!=` : différent de

`===` : contenu et type de variable égal à

`!==` : contenu ou type de variable différent de

`>` supérieur à

`>=` supérieur ou égal à

`<` : inférieur à

`<=` : inférieur ou égal à

`&&` qui signifie ET avec par exemple : `valeur1 && valeur2`

JavaScript : Récupération de valeur avec Prompt

Voici la base de cette fonction :

```
<script>
  var userName = prompt('Entrez votre prénom :');
  alert(userName); // Affiche le prénom entré par l'utilisateur
</script>
```

On peut demander le prénom et afficher un message avec concaténation :

```
<script>
  var start = 'Bonjour ', name, end = ' !', result;
  name = prompt('Quel est votre prénom ?');
  result = start + name + end;
  alert(result);
</script>
```

On peut aussi se servir de la fonction `prompt()` pour un calcul :

```
<script>
var first, second, result;
first = prompt('Entrez le premier chiffre :');
second = prompt('Entrez le second chiffre :');
result = parseInt(first) + parseInt(second); /* la fonction parseInt()
transforme la chaîne de caractères en nombre */
alert(result);
</script>
```

JavaScript : Conditions et itérations

```
<script>
  for (initialisation; condition; incrémentation) {
        instruction_1;
        instruction_2;
        instruction_3;
  } </script>
```

```
<script>
  var number = 1;
  while (number < 10) {
  number++; // Tant que le nombre est inférieur à 10, on l'incrémente de 1
  }
  alert(number); // Affiche : « 10 » </script>
```

```
<script>
var age = parseInt(prompt('Quel est votre âge ?'));
if (1 <= age && age <= 6) {
alert('Vous êtes un jeune enfant.');
} else if (7 <= age && age <= 11) {
alert ('Vous êtes un enfant qui a atteint l\'âge de raison.');
} else if (12 <= age && age <= 17) {
alert ('Vous êtes un adolescent.');
} else if (18 <= age && age <= 120) {
alert ('Vous êtes un adulte.');
} else {
alert ('Erreur !!');
}
</script>
```

```
<script>
  do {
        instruction_1; instruction_2; instruction_3;
  } while (condition);
</script>
```

JavaScript : Les fonctions

```
hello();
hello();

function hello() {
  console.log('Hello!');
  console.log('Welcome to JavaScript');
}
```

Cela fonctionne car les déclarations de fonctions sont ***hoisted*** : déplacées au sommet du scope dans lesquelles elles sont définies

il faut éviter de se reposer sur ce mécanisme

JavaScript : Les tableaux

Un tableau, ou plutôt un array en anglais, est une variable qui contient plusieurs valeurs, appelées items. Chaque item est accessible au moyen d'un indice (index en anglais) et dont la numérotation commence à partir de 0.

```
<script>
var myArray = ['Rafael', 'Mathilde', 'Ahmed', 'Jérôme', 'Guillaume'];
// Le contenu se définit entre crochets, avec une virgule entre chaque valeur.
// La chaîne 'Rafael' correspond à l'indice 0, 'Mathilde' à l'indice 1...
alert(myArray[1]); // Affiche : « Laurence »
</script>
```

On peut modifier une valeur :

```
<script>
var myArray = ['Rafael', 'Mathilde', 'Ahmed', 'Jérôme', 'Guillaume'];
myArray[1] = 'Paul';
alert(myArray[1]); // Affiche : « Paul »
</script>
```

JavaScript : Les tableaux

On peut ajouter des items avec la méthode `push()` :

```
<script>
var myArray = ['Rafael', 'Mathilde'];
myArray.push('Ahmed'); // Ajoute « Ahmed » à la fin du tableau
myArray.push('Jérôme', 'Guillaume'); // Ajoute « Jérôme » et « Guillaume » à la fin du tableau
</script>
```

La méthode `unshift()` fonctionne comme `push()`, excepté que les items sont ajoutés au début du tableau. Les méthodes `shift()` et `pop()` retirent respectivement le premier et le dernier élément du tableau.

```
<script>
var myArray = ['Rafael', 'Mathilde', 'Ahmed', 'Jérôme', 'Guillaume'];
myArray.shift(); // Retire « Rafael »
myArray.pop(); // Retire « Guillaume »
alert(myArray); // Affiche « Mathilde,Ahmed,Jérôme »
</script>
```

JavaScript : Les tableaux

On peut découper une chaîne de caractères en tableau avec `split()` :

```
<script>
var cousinsString = 'Jérôme Guillaume Paul',
cousinsArray = cousinsString.split(' '); // Avec les espaces, on a trois items
alert(cousinsString);
alert(cousinsArray);
</script>
```

On fait l'inverse avec `join()` :

```
<script>
var cousinsString_2 = cousinsArray.join('-');
alert(cousinsString_2); </script>
```

On peut parcourir un tableau avec `for` :

```
<script>
var myArray = ['Rafael', 'Mathilde', 'Ahmed', 'Jérôme', 'Guillaume']; // La
length est de 5
for (var i = 0, c = myArray.length; i < c; i++) { // On crée la variable c
pour que la condition ne soit pas trop lourde en caractères
alert(myArray[i]); // On affiche chaque item, l'un après l'autre, jusqu'à la
longueur totale du tableau
}
</script>
```

JavaScript : La DOM

Le DOM (*Document Object Model*) est une interface de programmation (ou API, *Application Programming Interface*) pour les documents XML et HTML. Via le Javascript, le DOM permet d'accéder au code du document ; on va alors pouvoir modifier des éléments du code HTML.

Contrairement à ce qui a été vu avant, `alert()` n'est pas vraiment une fonction, mais une méthode qui appartient à l'objet `window`, qui est implicite (il y a en fait très peu de variables globales). Les deux lignes suivantes signifient la même chose :

```
<script>
   alert('Hello world !');
   window.alert('Hello world !');
</script>
```

L'objet document est un sous-objet de window. L'objet document possède trois méthodes principales : `getElementById()`, `getElementsByTagName()` et `getElementsByName()`.

Avec `getElementById()` :

```
<div id="myDiv"><p>Un peu de texte <a>et un lien</a></p></div>
<script>
var div = document.getElementById('myDiv');
   alert(div); </script>
```

On nous dit alors que `div` est un objet de type `HTMLDivElement`. Cela fonctionne.

JavaScript : La DOM

On peut jouer sur les attributs d'une balise HTML avec l'objet `Element` et `getAttribute()` et `setAttribute()`, permettant par exemple de modifier un lien :

```
    <a id="myLink" href="http://www.un_lien_quelconque.com">Un lien modifié
dynamiquement</a>
    <script>
       var link = document.getElementById('myLink');
       var href = link.getAttribute('href'); // On récupère l'attribut « href »
       alert(href);
       link.setAttribute('href', 'http://blog.crdp-versailles.fr/rimbaud/'); // on édite
    </script>
```

Cela fonctionne aussi avec :

```
    <a id="myLink" href="http://www.un_lien_quelconque.com">Un lien modifié
dynamiquement</a>
    <script>
       var link = document.getElementById('myLink');
       var href = link.href;
       alert(href);
       link.href = 'http://www.clg-rimbaud-aubergenville.ac-versailles.fr/';
    </script>
```

Par contre on ne peut pas utiliser `class`, à remplacer par `className` en Javascript. Comme `for`, à remplacer par `htmlFor` (le `for` du Javascript servant aux boucles utiles aux fonctions).

JavaScript : Les évènements

Plusieurs exemples d'événements :

`click`	Cliquer (appuyer puis relâcher) sur l'élément
`dblclick`	Double-cliquer sur l'élément
`mouseover`	Faire entrer le curseur sur l'élément
`mouseout`	Faire sortir le curseur de l'élément
`mousedown`	Appuyer (sans relâcher) sur le bouton gauche de la souris sur l'élément
`mouseup`	Relâcher le bouton gauche de la souris sur l'élément
`mousemove`	Faire déplacer le curseur sur l'élément
`keydown`	Appuyer (sans relâcher) sur une touche de clavier sur l'élément
`keyup`	Relâcher une touche de clavier sur l'élément
`keypress`	Frapper (appuyer puis relâcher) une touche de clavier sur l'élément
`focus`	« Cibler » l'élément
`blur`	Annuler le « ciblage » de l'élément
`change`	Changer la valeur d'un élément spécifique aux formulaires (`input`, `checkbox`, etc.)
`select`	Sélectionner le contenu d'un champ de texte (`input`, `textarea`, etc.)

Deux événements spécifiques à <form> :

`submit`	Envoyer le formulaire
`reset`	Réinitialiser le formulaire

JavaScript : Les évènements

Dans `<input>`, on utilise la propriété `value`.

```
<input id="text" type="text" size="60" value="Vous n'avez pas le
focus !" />
<script>
  var text = document.getElementById('text');
  text.addEventListener('focus', function(e) {
        e.target.value = "Vous avez le focus !";
  }, true);
  text.addEventListener('blur', function(e) {
        e.target.value = "Vous n'avez pas le focus !";
  }, true);
</script>
```

Pour désactiver un champ de texte :

```
<input id="text" type="text" />
<script>
  var text = document.getElementById('text');
  text.disabled = true;
</script>
```

JavaScript : Les évènements

On peut utiliser `checked` pour des boutons radio :

```
<label><input type="radio" name="check" value="1" />Case n°1</label><br />
<label><input type="radio" name="check" value="2" />Case n°2</label><br />
<label><input type="radio" name="check" value="3" />Case n°3</label><br />
<label><input type="radio" name="check" value="4" />Case n°4</label>
<br /><br />
<input type="button" value="Afficher la case cochée" onclick="check();" />
<script>
  function check() {
  var inputs = document.getElementsByTagName('input'),
  inputsLength = inputs.length;
  for (var i = 0 ; i < inputsLength ; i++) {
        if (inputs[i].type == 'radio' && inputs[i].checked) {
        alert('La case cochée est la n°'+ inputs[i].value);
        }
  }
  }
</script>
```

JavaScript : Les évènements

Ou `selectedIndex` pour des listes déroulantes :

```
<select id="list">
  <option>Sélectionnez votre sexe</option>
  <option>Homme</option>
  <option>Femme</option>
</select>
<script>
  var list = document.getElementById('list');
  list.addEventListener('change', function() {
  // On affiche le contenu de l'élément <option> ciblé par la propriété selectedIndex
        alert(list.options[list.selectedIndex].innerHTML);
  }, true);
</script>
```

JavaScript : Les évènements

L'élément `<form>` possède deux méthodes intéressantes : `submit()` pour effectuer l'envoi du formulaire, `reset()` pour réinitialiser le formulaire :

```
<form id="myForm">
  <input type="text" value="Entrez un texte" /><br /><br />
  <input type="submit" value="Submit !" />
  <input type="reset" value="Reset !" />
</form>
<script>
  var myForm = document.getElementById('myForm');
  myForm.addEventListener('submit', function(e) {
        alert('Vous avez envoyé le formulaire !\n\nMais celui-ci a été bloqué pour que vous ne changiez pas de page.');
        e.preventDefault();
  }, true);
  myForm.addEventListener('reset', function(e) {
        alert('Vous avez réinitialisé le formulaire !');
  }, true);
</script>
```

JavaScript : Les évènements

Les méthodes `focus()` et `blur()` pour donner ou retirer le focus sur un événement.

```
<input id="text" type="text" value="Entrez un texte" /><br /><br />
<input type="button" value="Donner le focus"
  onclick="document.getElementById('text').focus();" /><br />
<input type="button" value="Retirer le focus"
  onclick="document.getElementById('text').blur();" />
```

Avec `select()`, on sélectionne le texte :

```
<input id="text" type="text" value="Entrez un texte" /><br /><br />
<input type="button" value="Sélectionner le texte"
  onclick="document.getElementById('text').select();" />
```

Partie 2: Sites web dynamiques

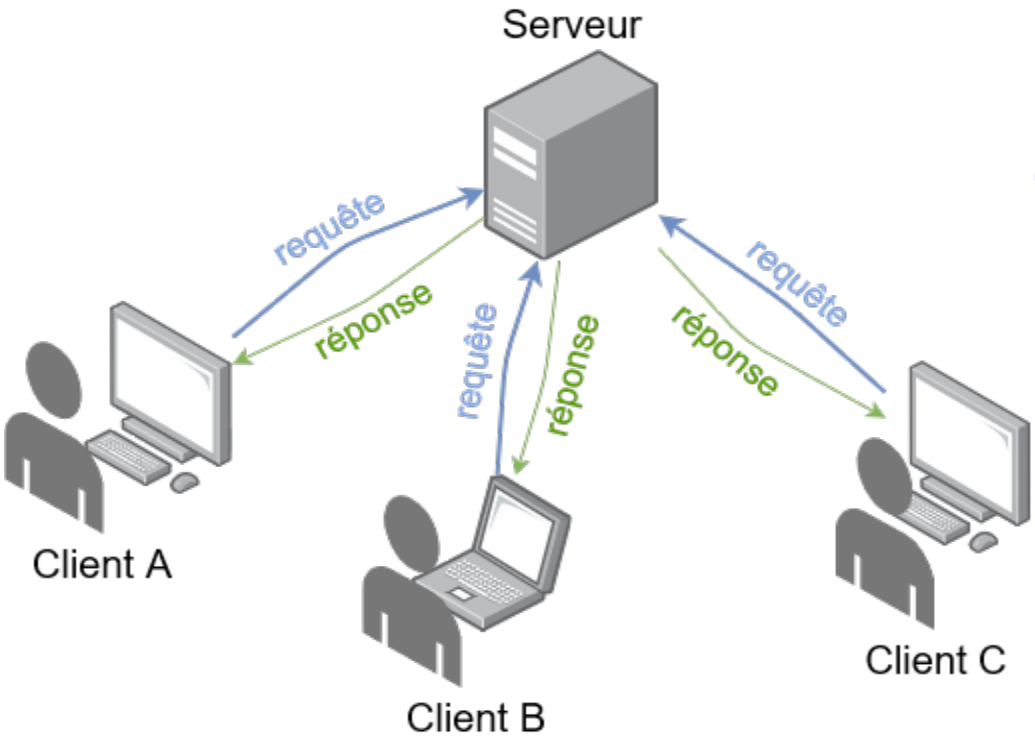
PHP & MySQL

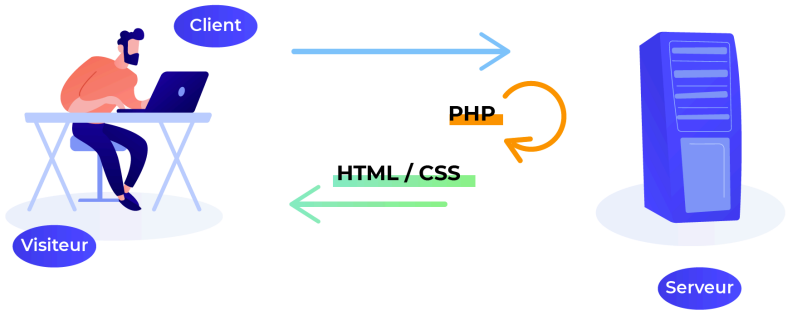
Sites web Dynamiques

- Site web Static (HTML+CSS) + Un ou plusieurs langages de programmation (PHP et JS)
- **Nécessite :** Serveur Web (APACHE), Serveur de développement (PHP) et Serveur de BD (MySQL)
- Rôle du Le langage de programmation :
  - Gérer les fonctionnalités du site : Calcul & traitements
  - Assurée l'Interactivité
    - Avec les Utilisateurs (Réseaux sociaux)
    - Avec une base de données (MySQL, Oracle etc)
    - D'autres Application via des API (Web Services)
  - Est souvent utilisé coté serveur sauf pour le JavaScript (JS) qui lui est coté client

PHP : Définitions

- Un langage de programmation interprété et orienté Objets
  - Respecte les concepts de la Programmation Orientée Objet (POO): CLASSES & OBJETS, ENCAPSULATION, HERITAGE et POLYMORPHISME
  - Syntaxiquement, c'est un mélange de C et de Perl.
- **C**omporte plus de 500 fonctions implicites.
- **Adopte l'architecture Client-Serveur**





PHP : Syntaxe de base

- Un fichier PHP à pour extension .php (Exemple: ajouter.php)
- Débute par un <?php et Se termine par **?>**
  - **Une instruction PHP se termine toujours par un point virgule ;**
  - PHP ne respect pas la casse
    - Pas de différence entre les majuscules et les minuscules
    - Sauf par rapport à l'utilisation des fonctions
- En PHP on peut intégrer des commentaires qui seront ignorés lors de l'exécution du script par le serveur. Les commentaires en PHP peuvent avoir deux formes:
  - **Commentaire de fin de ligne**:
    - S'étend jusqu'à la fin de la ligne à partir du symbole double slash (**//**).
  - **Commentaire sur plusieurs lignes**:
    - Peut contenir plusieurs lignes comprises entre les symboles **/*** et ***/**.

```
<?php
    // Commentaire de fin de ligne
    /*
        Bloc entier
        vu comme un
        commentaire
    */
?>
```

PHP : Affichage avec echo et print

- **Pour afficher en PHP on utilise echo ou print**
  - echoet printsont plus ou moins les mêmes. Ils sont tous deux utilisés pour sortir des données à l'écran.
- **La fonction echo :**
  - Syntaxte: echo(*strings*)
  - Les parenthèses ne sont pas obligatoires
    - Exemples:
      - echo "Chaine de caracteres";
      - echo (1+2)*87;
- **La fonction print**
  - print(expression);
    - print("Chaine de caracteres");
    - print ((1+2)*87);

```php
<?php
echo "Hello world!";
?>
```

```php
<?php
$str = "Hello world!";
echo $str;
echo "<br>What a nice day!";
?>
```

```php
<?php
$str1="Hello world!";
$str2="What a nice day!";
echo $str1 . " " . $str2;
?>
```

PHP : Les variables

- Une variable est une structure de données de ***type :***
  - ***Primitif*** (entier, réel, caractère, chaîne de caractères, booléen ou null)
  - ***Structuré*** (tableau ou objet) qui permet de stocker une ou plusieurs valeurs.
- Les variables n'ont pas à être explicitement déclarées
- Les noms de variables sont préfixées par le symbole dollar (**$**)
- On peut affecter des valeurs de types différents à une variable tout sans recourir au CAST.

```
<?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)
?>
```

- On peut tester l'existence d'une variable avec isset()

```
<?php
    // Déclaration de la variable
    $prenom = 'Ngor';
    echo isset($prenom); // Retourne TRUE -> affiche 1
?>
```

Fonctions de verification de Type

- empty() : est null
- gettype() : Recuperer le type
- is_array(): est un tableau
- is_bool(): est boolen
- is_double(): est double
- is_float(): est float
- is_int(): est entier
- is_integer: est entier
- is_long(): est entire long
- is_object(): est un objet
- is_real(): est un reel
- is_numeric(): est numerique
- is_string(): est une chaine

PHP : Les constantes

- Les **constantes** servent à stocker des valeurs dans un programme, mais à l'inverse des variables, leurs valeurs ne changent pas durant l'exécution du programme.

- **Fonction define()**
  - Pour définir un constante on utilise la fonction **define(cte,val)**.
  - **cte** représente l'identifiant de la constante à définir et **val** sa valeur.
  - Pour lire la valeur d'une constante, il suffit de l'appeller par son nom sans le $

```php
<?php
    define("defined", "juste une constante définie",true);
    //true le rend non sensible à la casse
    echo DEFINED ."<br>";
    echo constant("defined")."<br>";

    // en utilisant const
    const tryconst = "Il s'est passé beaucoup de choses";
    echo tryconst."<br>";
    echo constant("tryconst");
?>
```

```php
<?php
    define(cte,"Bonjour");
    echo cte; // Affiche Bonjour
    define(cte,"Bonsoir");
    echo cte; // Affiche Bonjour
?>
```

PHP : Les operateurs

- **Les operateurs arithmétiques**
  - Ce sont des opérateurs qui effectuent des calculs mathématiques classiques

Opérateur	Opération	Exemple	Résultat
-	Négation	-$a	Opposé de $a
+	Addition	$a + $b	Somme de $a et $b
*	Multiplication	$a * $b	Produit de $a et $b
-	Soustraction	$a - $b	Différence de $a et $b
/	Division	$a / $b	Quotient de $a et $b
%	Modulo	$a % $b	Reste de $a / $b

```
$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
```

- **Les operateurs incrémentation**
  - Ce sont des opérateurs qui permettent de modifier la valeurs d'une variable en l'augmentant ou la diminuant de 1. Deux opérateurs sont utilisés:
    - Incrémentation (**++**)
    - Décrémentation (**--**).

Opérateur	Opération	Exemple	Résultat
++	Pré-Incrémentation	++$a	Incrémente $a, puis retourne $a
++	Post-Incrémentation	$a++	Retourne $a, puis incrémente $a
--	Pré-Décrémentation	--$a	Décrémente $a, puis retourne $a
--	Post-Décrémentation	$a--	Retourne $a, puis décrémente $a

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

PHP : Les operateurs

- **Les operateurs Logiques**
  - Ce sont des opérateurs qui regroupent logiquement plusieurs conditions.
  - Les trois opérateurs utilisés sont: ET logique (**&&**), OU logique (**||**) et NON logique (**!**).

Opérateur	Opération	Exemple	Résultat
&&	ET	$a && $b	TRUE si $a ET $b sont vrais
AND	ET	$a AND $b	Alias de &&
||	OU	$a || $b	TRUE si $a OU $b est vrai
OR	OU	$a OR $b	Alias de ||
XOR	OU exclusif	$a XOR $b	TRUE si $a OU $b est vrai mais pas les deux
!	NON	!$a	TRUE si $a est faux

- **Les operateurs d'assignement**
  - Permettent d'affecter une valeur à une variable directement ou à travers une opération
  - Permettent de simplifier des opérations telles l'*ajout d'une valeur dans une variable et stocker le résultat dans la même variable*.
  - *$x=$x+2; donne* Avec les opérateurs d'assignement *$x+=2*

Opérateur	Effet
+=	addition deux valeurs et stocke le résultat dans la variable (à gauche)
-=	soustrait deux valeurs et stocke le résultat dans la variable
*=	multiplie deux valeurs et stocke le résultat dans la variable
/=	divise deux valeurs et stocke le résultat dans la variable
%=	donne le reste de la division deux valeurs et stocke le résultat dans la variable
|=	Effectue un OU logique entre deux valeurs et stocke le résultat dans la variable
^=	Effectue un OU exclusif entre deux valeurs et stocke le résultat dans la variable
&=	Effectue un Et logique entre deux valeurs et stocke le résultat dans la variable
.=	Concatène deux chaînes et stocke le résultat dans la variable

PHP : Les operateurs

- **Les operateurs de comparaison**
  - Ce sont les opérateurs qui testent une condition si elle est vraie ou fausse.

Opérateur	Opération	Exemple	Résultat
==	Egalité en valeur	$a == $b	Vérifie que les valeurs de $a et $b sont identiques
===	Egalité en valeur et type	$a === $b	Vérifie que les valeurs et types de $a et $b sont identiques
!=	Différence en valeur	$a != $b	Vérifie que les valeurs de $a et $b sont différentes
!==	Différence en valeur et type	$a !== $b	Vérifie que les valeurs et types de $a et $b sont différents
<>	Différence en valeur	$a <> $b	Alias de !=
<	Infériorité stricte	$a < $b	Vérifie que $a est strictement inférieur $b
<=	Infériorité ou égalité	$a <= $b	Vérifie que $a est strictement inférieur ou égal à $b
>	Supériorité stricte	$a > $b	Vérifie que $a est strictement supérieur $b
>=	Supériorité ou égalité	$a >= $b	Vérifie que $a est strictement supérieur ou égal à $b

PHP : La concaténation

- La concaténation n'est ni plus ni moins que l'opération permettant d'assembler deux ou plusieurs informations dans une variable.
- Cette opération se réalise au moyen de l'opérateur de concaténation qui est le point (.).

```
<?php
    $prenom = 'Edouard';
    $nom = 'SARR';
    $identite = '';
    $identite = $prenom .' '. $nom;
    Echo $identite;
?>
```

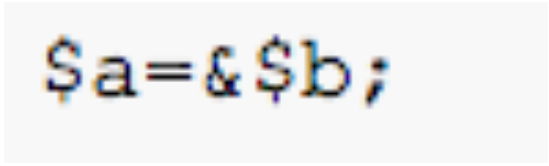
```php
<?php
    Define ('Taux',0.18);
    $MHT= 1200000;
    $MTVA= $MHT*Taux;
    $NetPayer= $MHT + $MTVA;
    Echo ('la Tva sur : '.$MHT . 'est : '.$MTVA );
    Echo ('le net a payer est : '.$NetPayer)
?>
```

```php
<html>
<body>
    <?php
        $prenom = 'edouard';
        $nom= 'sarr';
        $salaire= 100000;
        echo ('vous etes : '.$prenom.' '.$nom);
        echo ('votre salaire est : '.$salaire);
        $new_salaire= $salaire + (($salaire *10)/100);
        echo  ('votre nouveau salaire est: '.$new_salaire);
    ?>
</body>
</html>
```

PHP : Assignation par référence

- L'assignation par référence permet à deux variables de pointer sur le même contenu.
- Cette assignation est assurée par une simple affectation avec le paramètre **&**.
- Dans ce cas, les variables $a et $b pointent sur le même contenu.
- Autrement dit, si, après cette assignation, on change la valeur de $a alors $b changera aussi et aura la même valeur que $a et inversement.

```
$a=&$b;
$a=10;
echo ($a); //affiche 10
echo ($b); //donne 10
```




```php
<?php
   $var1= 100;
   $var2=200;

   echo "var1 =".$var1.'</br>';
   echo "var2 =".$var2.'</br>';
   echo "**************************************".'</br>';
   $res1= $var1;
   $res2= &$var2;
   $res1=110;
   $res2= 500;
   echo "**************************************".'</br>';
   echo "res 1 =".$res1.'</br>';
   echo "res 2 =".$res2.'</br>';
   echo "var1 =".$var1.'</br>';
   echo "var2 =".$var2.'</br>';

?>
```

```
var1 =100
var2 =200
**************************************
**************************************
res 1 =110
res 2 =500
var1 =100
var2 =500
```

PHP : Le transtypage

- **Transtypage explicité**
  - Le nom du type désiré est écrit entre parenthèses
  - Les conversions autorisées sont :
    - (int) , (integer) - type entier
    - (bool) , (boolean) - booléen
    - (double) , (float) , (real) - type double
    - (string) - type chaîne de caractère
    - (array) - type tableau
    - (object) - type objet
- **Transtypage implicite**
  - Changement automatique de type

```php
<?php
$foo = 10;   // $foo est un entier
$bar = (double) $foo;   // $bar est un double
?>
```

```php
<?php
$foo = 10;            // $foo est un entier
$str = "$foo";        // $str est une chaîne
$fst = (string) $foo; // $fst est aussi une chaîne

// Ceci affiche : "Identique"
if ($fst === $str) {
    echo 'Identique';
}
?>
```

```php
<?php
$foo = "1";                  // $foo est une chaîne de caractères (ASCII 49)
$foo *= 2;                   // $foo est maintenant un entier (2)
$foo = $foo * 1.3;           // $foo est maintenant un nombre à virgule flottante (2.6)
$foo = 5 * "10 Little Piggies"; // $foo est un entier (50)
$foo = 5 * "10 Small Pigs";     // $foo est un entier (50)
?>
```

PHP : Instruction conditionnelle IF

- **Permet de gérer les conditions**
  - Plus connue de toutes car on la retrouve dans tous les langages de programmation.
  - Permet d'exécuter un bloc d'instructions uniquement si l'expression est vraie.
- **IF imbriqués**
  - Introduire un If dans un autre IF
  - Nous pouvons alors avoir

    IF (--)

    IF(--)

    ELSE (--)

    IF(--)
  - Impossible d'avoir

    IF(--)

    ELSE(--)

    ELSE(--)

    ELSE (--)
  - Attention à la différence entre
    - ELSEIF
    - ELSE
      - IF

```php
<?php
if ($a > $b) {
    echo "a est plus grand que b";
} elseif ($a == $b) {
    echo "a est égal à b";
} else {
    echo "a est plus petit que b";
}
?>
```

```php
<?php
  if($var == 'Whatever') {

  } else {
      if($var == 'Something Else') {

      }
  }
?>
```

PHP : Instruction conditionnelle SWITCH

- **S*witch()* (traduire par *au cas où*).**
  - Permet de tester toutes les valeurs possibles que peut prendre une variable.
  - ***case*** définit la valeur à tester.
  - ***break*** permet de sortir du bloc switch() si l'on a exécuté un bloc d'instructions.
    - Il est facultatif mais vivement conseillé dans la majorité des cas d'utilisation.
  - Si la variable a une valeur non définie dans le switch(), alors un bloc ***default*** est lancé.
    - Recommandé de toujours prévoir un bloc *default*

```
switch (n) {
  case label1:
    code to be executed if n=label1;
    break;
  case label2:
    code to be executed if n=label2;
    break;
  case label3:
    code to be executed if n=label3;
    break;
    ...
  default:
    code to be executed if n is different from all labels;
}
```

```
<?php
switch ($i):
    case 0:
        echo "i égal 0";
        break;
    case 1:
        echo "i égal 1";
        break;
    case 2:
        echo "i égal 2";
        break;
    default:
        echo "i n'est ni égal à 2, ni à 1, ni à 0";
endswitch;
?>
```

PHP : IF vs Switch

- **Tout ce que peut fait un IF, un switch peut aussi le faire**
- Le bloc Switch est plus rapide que IF
- Quand utiliser If à la place de Switch
  - Si des conditions (des cas) ont des traitement similaires
    - Permet de regrouper des cas
    - Edition plus simple et courte
    - Mains lent car toutes les conditions sont à tester
- Quand utiliser Switch à la place de IF
  - Si chaque conditions à un traitement particulier
    - Permet de tester tous les cas possibles
    - Plus long à editer
    - Plus rapide grace au BREAK

```php
if ($i == 0) {
    echo "i égal 0";
} elseif ($i == 1) {
    echo "i égal 1";
} elseif ($i == 2) {
    echo "i égal 2";
}
```

```php
switch ($i) {
    case 0:
        echo "i égal 0";
        break;
    case 1:
        echo "i égal 1";
        break;
    case 2:
        echo "i égal 2";
        break;
}
```

PHP : Instruction itérative FOR

- **FOR = POUR en ALGORITHME**
- Une boucle: permet de répéter un ou plusieurs instructions plusieurs fois (n)
- Nous avons trois boucle en PHP : FOR, WHILE et DO WHILE
- Toutes les boucles font la même chose : Répéter des instructions
  - La différence réside dans l'exécution de la condition d'arrêt.
- **Particularité de la boucle FOR**
  - La boucle for est utilisée lorsque vous **savez** à l'avance le nombre d'itérations (tours)
    - Nb Itération = MAX-MIN
- **Syntaxe**

```
for (init counter; test counter; increment counter) {
  code to be executed for each iteration;
}
```

- **Exemple**

```
<?php
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
?>
```

PHP : Instruction itérative WHILE

- **WHILE : TANT QUE de l'Algorithme**
- **Particularité**:
  - La condition est vérifier dés le départ
  - WHILE exécute un bloc de code tant que la condition spécifiée est vraie.
  - La boucle WHILE boucle est utilisée lorsque
    - vous **ne savez pas** à l'avance le nombre d'itérations
    - Et que le minimum d'itérations soit égale à 0
      - Peut y arriver qu'aucun tour ne soit effectué
- **Syntaxe**

```
while (condition is true) {
  code to be executed;
}
```

- **Exemple**

```php
<?php
$x = 1;

while($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
}
?>
```

PHP : Instruction itérative DO WHILE

- **DO WHILE = REPETER (FAIRE) TANT QUE en Algorithme**
- **Particularité**:
  - La condition est vérifier dés la fin du premier tour
  - Do WHILE exécute un bloc de code une première fois puis répète le bloc tant que la condition spécifiée est vraie.
  - La boucle DO WHILE boucle est utilisée lorsque
    - Vous **ne savez pas** à l'avance combien de fois le script doit s'exécuter
    - Et que le minimum soit égale à 1
      - Au mois un tour est effectué
- **Syntaxe**

```
do {
  code to be executed;
} while (condition is true);
```

- **Exemple**

```
<?php
$x = 1;

do {
  echo "The number is: $x <br>";
  $x++;
} while ($x <= 5);
?>
```

PHP : Instructions de branchement

- Une instruction de branchement permet au programme d'arrêter ou de sauter des instructions dans un bloc particulièrement dans une boucle.
- **Break**
  - Break = Stop
  - Peut également être utilisée pour sortir d'une boucle.

```php
<?php
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
    break;
  }
  echo "The number is: $x <br>";
}
?>
```

```php
<?php
$x = 0;

while($x < 10) {
  if ($x == 4) {
    break;
  }
  echo "The number is: $x <br>";
  $x++;
}
?>
```

- **Continue**
  - Permet interrompre une itération (dans la boucle), si une condition spécifiée se produit, et continue avec l'itération suivante dans la boucle.

```php
<?php
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
    continue;
  }
  echo "The number is: $x <br>";
}
?>
```

PHP : Les tableaux

- **Un tableau array() : Peut contenir des valeurs de types differents**
  - Une variable spéciale, qui peut contenir plusieurs valeurs à la fois.
  - On peut accéder à valeur grace son indice (numéro de sa cellule).
- Il existe trois types de tableaux :

```
array();
```

  - Tableaux **indexés** - Tableaux avec un index numérique
  - Tableaux **associatifs** - Tableaux avec clés nommées
  - Tableaux **multidimensionnels** - Tableaux contenant un ou plusieurs tableaux
- **Les fonctions PHP sur les tableaux**
  - array_count_values — Compte le nombre de valeurs d'un tableau
  - array_key_first — Récupère la première clé d'un tableau
  - array_key_last — Récupère la dernière clé d'un tableau
  - array_merge — Fusionne plusieurs tableaux en un seul
  - array_rand — Prend une ou plusieurs clés, au hasard dans un tableau
  - array_reverse — Inverse l'ordre des éléments d'un tableau
  - array_sum — Calcule la somme des valeurs du tableau
  - array_unique — Dédoublonne un tableau
  - sort — Trie un tableau en ordre croissant
  - rsort — Trie un tableau en ordre décroissant
  - array_shift() et array_pop() - récupérer le premier (oudernier) élément d'un tableau
  - array_key_exists('nom', $tableau) - Test si un tableau contient une clé donnée
  - count($cars): Compte le nombre d'éléments dans un tableau

PHP : Les tableaux

- **Parcourir un tableau associatif avec Foreach**
  - Permet parcourir un tableau ayant des indices de type chaîne de caractères
  - **foreach(TAB as $value)**
    - Passe en revue le tableau TAB.
    - A chaque itération, la valeur de l'élément courant est assignée à $value et le pointeur interne de tableau est avancé d'un élément

```
foreach ($array as $value) {
  code to be executed;
}
```

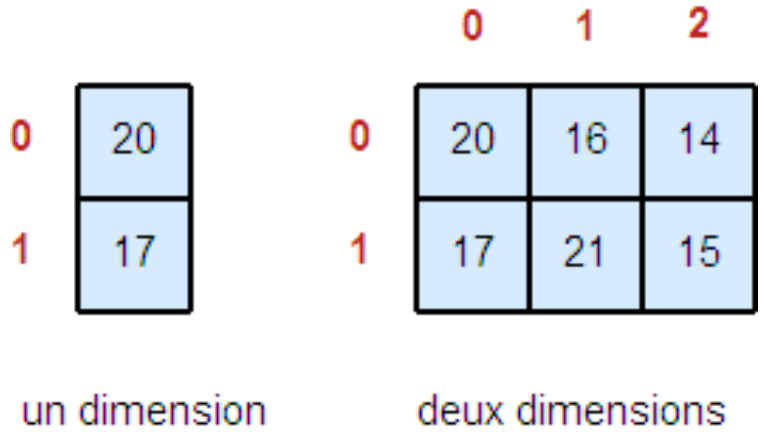
  - **foreach(an_array as $key => $value)**
    - Fait exactement la même chose
    - En plus de la valeur, la clé de l'élément courant est assigné à la variable $key, ce qui permet de récupérer à la fois la clé et sa valeur.

```
<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
?>
```

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $val) {
  echo "$x = $val<br>";
}
?>
```

PHP : Les tableaux multidimensionnels

- Un tableau multidimensionnel est un tableau qui va lui-même contenir d'autres tableaux en valeurs.
- **Un Tableau à deux dimensions est un tableau qui contient un ou plusieurs tableaux à une dimensions**
- **Tableau** à trois dimensions un **tableau** qui contient un ou plusieurs **tableaux** en valeurs qui contiennent eux-mêmes d'autres **tableaux** en valeurs et etc.
- Les « sous » tableaux vont pouvoir être des tableaux numérotés ou des tableaux associatifs ou un mélange des deux.




```php
$suite = [
    [1, 2, 4, 8, 16],
    [1, 3, 9, 27, 81]
];
```

```php
//$sous_suite = [1, 2, 4, 8, 16]
$sous_suite = $suite[0];
echo $sous_suite[0]. '<br>'.$sous_suite[2]. '<br>';
```

PHP : Fonctions sur les chaines de caractères

- AddCSlashes —Ajoute des slashes dans une chaîne
- AddSlashes — Ajoute un slash devant tous les caractères spéciaux.
- strrev — Inverse l'ordre des caractères d'une chaîne.
- Chop — Enlève les espaces de fin de chaîne.
- crypt — Encrypte une chaîne avec un DES.
- explode — Scinde une chaîne en morceau, grâce à un délimiteur.
- implode — Regroupe tous les éléments d'un tableau dans une chaîne, avec une chaîne de jointure.
- join — Regroupe tous les éléments d'un tableau dans une chaîne, avec une chaîne de jointure
- ltrim — Enlève les espaces de début de chaîne.
- strstr() -- insensible à la casse.
- strlen — Retourne la longueur de la chaîne.
- strpos — Recherche la dernière occurrence d'un caractère dans une chaîne.
- strrchr — Recherche la dernière occurrence d'un caractère dans une chaîne
- strrpos — Recherche la dernière occurrence d'un caractère dans une chaîne.
- strstr — Trouve la première occurence d'une chaîne.
- strtolower — Met les caractères en minuscule.
- strtoupper — Met les caractères en majuscule.
- str_replace — Remplace toutes les occurrences d'une chaîne par une autre.
- strtr — Remplace toutes les occurrences d'un caractère par un autre.
- substr — Retourne une partie de la chaîne.
- substr_replace — Remplace dans une sous partie de chaîne
- trim — Enlève les espaces de fin dans une chaîne.
- ucfirst — Premier caractère en majuscule.
- ucwords — Force le premier caractère de chaque mot d'une chaîne en majuscul

PHP : Gestion des fichiers

- En PHP il est possible de manipuler des fichier (LIRE et ECRIRE)
- Etape 1: Ouvrir le fichier avec fopen()
- Etape 2: Eécrire avec fwrite () ou lire fread() ou fgets()
- Etape 3: Fermer le fichier fcclose ()
- NB : Lors de l'ouverture du fichier il y a plusieurs mode *fopen(fichier, mode);*

r	Ouvre le fichier en lecture seule. Cela signifie que vous pourrez seulement lire le fichier.
r+	Ouvre le fichier en lecture et écriture. Vous pourrez non seulement lire le fichier, mais aussi y écrire (on l'utilisera assez souvent en pratique).
a	Ouvre le fichier en écriture seule. Mais il y a un avantage : si le fichier n'existe pas, il est automatiquement créé.
a+	Ouvre le fichier en lecture et écriture. Si le fichier n'existe pas, il est créé automatiquement. Attention : le répertoire doit avoir un CHMOD à 777 dans ce cas ! À noter que si le fichier existe déjà, le texte sera ajouté à la fin.

fopen('doc/fic.ttxt' 'a');

– Le premier argument de la fonction open peut être le nom du fichier lui-même ou un chemin d'accès vers le fichier lui-même

PHP : Gestion des fichiers

- **Ecrire dans un fichier avec fwrite**
  - Elle posséde 2 arguments :

  int fwrite (le fichier, La chaine) ;
  - Retourne un int (0 ou 1).
- **Lire un fichier jusqu'à une limite**
  - Lire tout le contenu du fichier : fread()
  - Lire le fichier ligne par ligne : fgets()

```php
<?php
    $fp=fopen ("fichier.txt", "w");
    //ouverture du fichier en mode écriture, création du fichier s'il n'existe pas.
    fwrite($fp,"Un texte dans votre fichier");
    // insert le texte: Un texte dans votre fichier.
?>
```

Lire le fichier en entier :

```php
<?php
$file = fopen("test.txt","r");
fread($file,filesize("test.txt"));
fclose($file);
?>
```

```php
<?php
    $chemin="F:/fic.csv";
    $monfichier = fopen ($chemin, 'r') ;
    $ligne=fgets($monfichier);
    while ($ligne != null) {
        echo ($ligne."</br>") ;
    $ligne=fgets($monfichier) ;
    }
    fclose($monfichier) ;
?>
```

PHP : INCLUDE & REQUIRE

- **Include (ou require)**
  - Prend tout le texte/code/balisage existant dans le fichier spécifié et le copie dans le fichier.
  - L'inclusion de fichiers est très utile lorsque vous souhaitez inclure le même PHP, HTML ou texte sur plusieurs pages d'un site Web.
  - Les instructions include et require sont font la même chose sauf que Lorsqu'un fichier est inclus avec include et que PHP ne le trouve pas, le script continuera à s'exécuter contrairement au require qui arrete l'execution au niveau du require.
- Si nous incluons le fichier "footer.php", les variables peuvent être utilisées dans le fichier appelant :

```php
<?php include 'footer.php';?>

<?php require 'noFileExists.php';
```

```
include 'filename';

or

require 'filename';
```

PHP : Les variables d'environnement

- Variables superglobales prédéfinies accessibles dans tous les scripts via $_SERVER

$_SERVER['HTTP_ACCEPT_LANGUAGE'] Langage accepté par le navigateur

$_SERVER['HTTP_HOST'] Nom de domaine du serveur

$_SERVER['HTTP_USER_AGENT'] Type de navigateur

$_SERVER['REMOTE_ADDR'] Adresse IP du client

$_SERVER['SERVER_ADDR'] Adresse IP du serveur

$_SERVER['PHP_SELF'] Nom du script en cours d'exécution

…

PHP : Les formulaires

- L'un des points forts de PHP est sa capacité à gérer les formulaires.
- Les formulaires sont l'élément essentiel qui permet l'interactivité entre un site et ses visiteurs.
- Ils constituent pour cette raison la base de la création de sites dynamiques.
- Tout échange entre visiteur et serveur passe par un formulaire, dans lequel l'utilisateur peut saisir textes ou mots de passe, opérer des choix grâce à des boutons radio, des cases à cocher ou des listes de sélection, voire envoyer ses propres fichiers depuis le poste client.
- Il est donc important d'en maîtriser la création à la fois avec HTML, pour obtenir des formulaires présentables, et avec PHP, pour gérer les informations fournies par le formulaire au script côté serveur.

```
<form action="action.php" method="post">
   <label>Votre nom :</label>
   <input name="nom" id="nom" type="text" />

   <label>Votre âge :</label>
   <input name="age" id="age" type="number" /></p>

   <button type="submit">Valider</button>
</form>
```

PHP : Les formulaires

<form method="post" action="submit1.php">

- **<FORM METHOD=… ACTION=…> ... </FORM>**, une balise qui permet de regrouper plusieurs éléments de formulaire (boutons,champs de saisie,...) et dont les attributs les plus courants sont :

  - **METHOD** indique la méthode de transmission HTTP des données saisies dans le formulaire :
    - « **POST** » est la valeur qui correspond à un envoi de données stockées dans le corps de la requête (les données sont cachées)
    - « **GET** » correspond à un envoi des données **visibles dans l'URL** (séparées de l'adresse du script par un point d'interrogation) : **à éviter pour des raisons évidentes de sécurité.**
    - **si cet attribut est omis, il vaut "get" par défaut.**

  - **ACTION** permet d'indiquer l'URL qui va recevoir les informations entrées dans le formulaire, lorsque l'on cliquera sur le bouton de validation. Plus précisément, l'URL est l'adresse d'un programme (un script) qui va récupérer les données et les traiter. Si le champ ACTION est absent, l'URL sera celle du document courant.

PHP : Les formulaires

<input type="submit" name="envoi" value="Soumettre">

- La balise **input** définit des champs d'information. L'attribut **name** désigne l'élément, l'attribut **value** désigne éventuellement une valeur initiale, et l'attribut **type** définit des objets:
  - **text** : petit champ pour que l'utilisateur saisisse un nombre ou une phrase
  - **button** : pour cliquer dessus
  - **checkbox** : cases à cocher (information on/off)
  - **radio** des "boutons radio" (information on/off mais un seul peut être coché à la fois)
    - les checkbox et radios n'envoient rien (même pas l'attribut name) s'ils ne sont pas cochés
  - **hidden** : champ caché pour transmettre des informations vers le serveur, sans que l'utilisateur ne le sache : très utile pour gérer un caddie ...
  - **password** : champ texte mais masquant le texte tapé
  - **reset** pour remettre le formulaire dans son état initial
  - **submit** : action indispensable lorsque le formulaire est **utilisé pour envoyer de l'information depuis l'utilisateur vers le serveur** : "envoie" les données du formulaire vers le serveur
  - **image** : même principe que submit mais c'est une image au lieu d'un bouton.
  - **file** : Affiche un bouton permettant de sélectionner un fichier de l'ordinateur ; la plupart des navigateurs l'accompagnent d'une case de texte contenant le chemin d'accès au fichier

PHP : Les formulaires

Récupération d'un champ

```
<?php
$bouton = $_POST['send'];
if(!empty($bouton))
{
  $nom = trim($_POST['nom']);
  $prenom = trim($_POST['prenom']);
  if(!empty($nom) && !empty($prenom))
  {
    echo 'Bonjour, '.$prenom.' '.$nom;
  }
else
  {
    echo 'vous n\'avez pas rempli tous les champs';
  }
}
?>
```

dans le tableau associatif $_POST, une entrée à été créée avec le nom de chaque variable du formulaire, ici deux champs nom et prénom.

Le formulaire contient :

```
<input type="submit"
value="envoyer"
name="send">
```

Remarque : !empty permet de vérifier que :

- la variable n'est pas vide
- la variable ne contient ni 0, NULL, "0" ou FALSE

PHP : Les formulaires

Récupération d'un Fichier

$_FILES['variable']['name'] Le nom original du fichier provenant de la machine de l'utilisateur

$_FILES['variable']['type'] Le type mime du fichier

$_FILES['variable']['size'] Le taille du fichier en bytes

$_FILES['variable']['tmp_name'] Le nom temporaire du fichier stocké sur le serveur

$_FILES['variable']['error'] Le code erreur associé à l'upload (attention cette option à été ajoutée en PHP 4.2.0)

Formulaire :

```
<form enctype="multipart/form-data" action="exemple2.php" method="post">
<input name="fichier" type="file">
<input type="submit" value="send" name="bouton">
</form>
```

PHP : Les formulaires

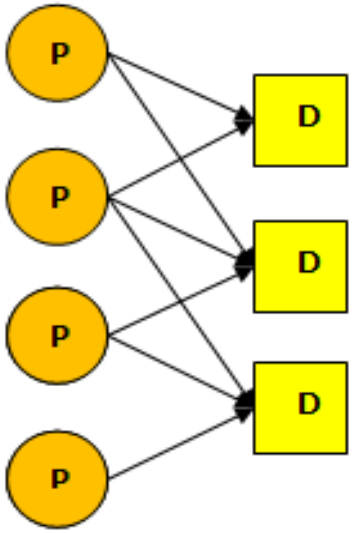
Récupération d'un Fichier

```
<?php
$bouton = $_POST['bouton'];
if(!empty($bouton)) {
  $fichier = $_FILES['fichier']['name'];       $size = $_FILES['fichier']['size'];
  $tmp = $_FILES['fichier']['tmp_name'];   $type = $_FILES['fichier']['type'];
  $error = $_FILES['fichier']['error'];          $max = 10000;
}
if(isset($fichier)) {
  if($size <= $max) {
    echo 'Nom d\'origine =>'.$fichier.'<br />';  echo 'Taille =>'.$size.'<br />';
    echo 'Nom sur le serveur =>'.$tmp.'<br />'; echo 'Type de fichier =>'.$type.'<br />';
    echo 'Code erreur =>'.$error.'<br />';
  }
  else {
     echo 'Le fichier est trop volumineux';
  }
}
else {
   echo 'aucun fichier envoyé';
}
}
?>
```

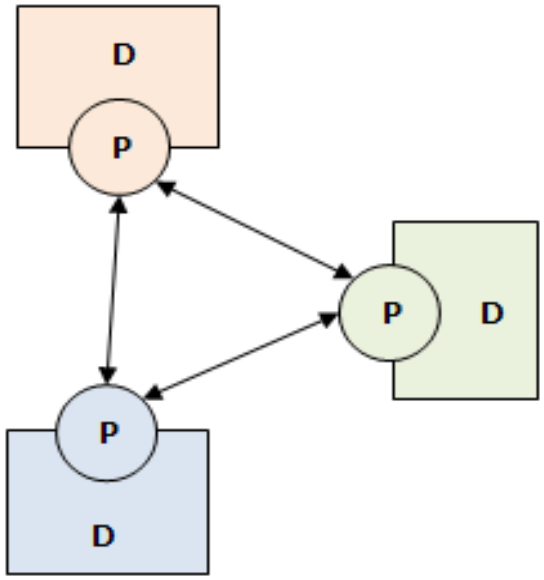
PHP Objet : La POO

- **Dans l'approche procédurale**
  - Les données et les procédure sont traitées indépendamment les unes des autres
  - Basée sur le principe d'appel de fonction.
- **Dans L'approche objet**
  - **D**onnées et traitements sont regroupés dans une classe pour améliorer le contrôle et la réutilisabilité.
  - Des instances de cette classe (Objets) sont créées (instancier) pour être utiliser

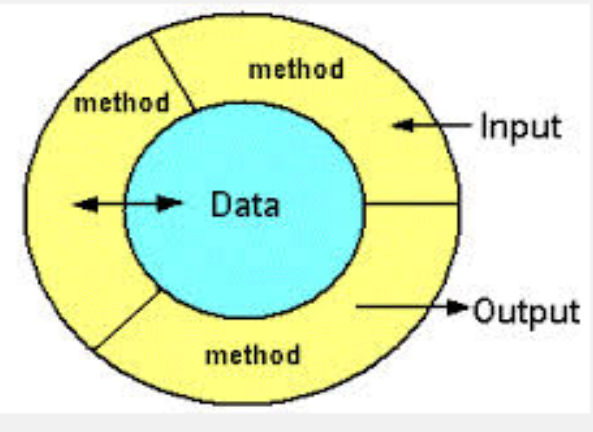
NON ORIENTÉE OBJET



ORIENTÉE OBJET



Encapsulation



PHP Objet : Classe & Objet

- Dans le monde réel, nous comprenons les objets en les associant à des catégories : Awa est un étudiant, Mr SARR un professeur, UASZ une université et Sénégal un Pays.
- Tous les membres d'une catégorie possèdent des caractéristiques et des comportements communs peu importe leurs états individuels.
- Un étudiant particulier Moussa est simplement une **instance** de la **classe (Type) Etudiants** donc possède les même propriétés (attributs) et comportements (méthodes) que toutes autres instances de la classe Etudiants.
  - Par exemple. Si l'objet Moussa et l'objet Awa sont des étudiants, alors si Awa à une propriété Prénom, alors moussa devra aussi en disposé et inversement.
  - **Par contre les valeurs des attributs** (caractéristiques propres) sont susceptibles d'être différentes c-a-d l'Age de l'objet Awa peut être égale à 23 alors que celui de Moussa est de 25.
- Un objet est une entité logicielle :
  - Ayant une identité c'est à dire un nom.
  - Capable de sauvegarder un état
  - Répondant à des messages précis en déclenchant des activations internes appropriés
- Exemple:
  - Moussa est un Objet de la classe Etudiant
  - Range_Rover qui est de type Voiture
  - Compte00545 qui est de type compte

PHP Objet : Les fonctions (Méthodes)

- Une fonction est une méthodes (un comportement de la classe) qui retourne une valeur à la fin de l'opération.
- Une fonction peut avoir trois types de visibilités
  - private : signifie que la fonction est accessible **seulement** à l'intérieur de la classe.
  - public function : permet de définir des fonction publiques, donc accessibles à l'extérieur.
  - $this : référence à la fonction actuelle

```
<?php
    class Calculatrice {
        public function addition($a, $b) {
            return $a + $b;
        }
        public  function aoustraction($a, $b) {
            return $a - $b;
        }
    }
    // Utilisation de la classe
    $calc = new Calculatrice();
    $resultat = $calc->addition(5, 3);
    echo "Le résultat de l'addition est : " . $resultat;
?>
```

PHP Objet : Getter et Setter

- Un getter est une méthode qui retourne la valeur d'un attribut privé ou protégé d'une classe.
- Un setter est une méthode qui permet de modifier la valeur d'un attribut, souvent avec des vérifications.

```php
<?php
class Fruit {
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
}
?>
```

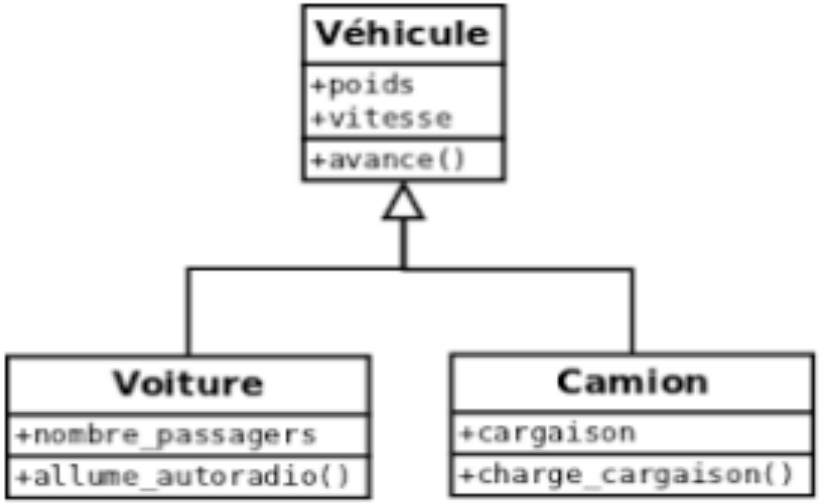
```php
$apple = new Fruit();
$banana = new Fruit();
$apple->set_name('Apple');
$banana->set_name('Banana');

echo $apple->get_name();
echo "<br>";
echo $banana->get_name();
```

PHP Objet : Héritage

- Les classes peuvent être organisées en hiérarchies, chaque classe héritant de sa classe mère ou **super-classe**.
- **Permettant à une classe fille d'heriter des propriétés et méthodes de sa classe mère sauf celles declarées private.**
- *On dit qu'une classe B hérite d'une classe A (B étant une sous-classe de A) lorsque toutes les méthodes et propriétés public de la classe A sont accessible aux objets de la classe B.*

La Classe fille peut alors :

1. *Utiliser les membres de la classe mère*
2. *Ajouter ses propres propriétés et méthodes*
3. *Redéfinir certains membres de la classe mère.*

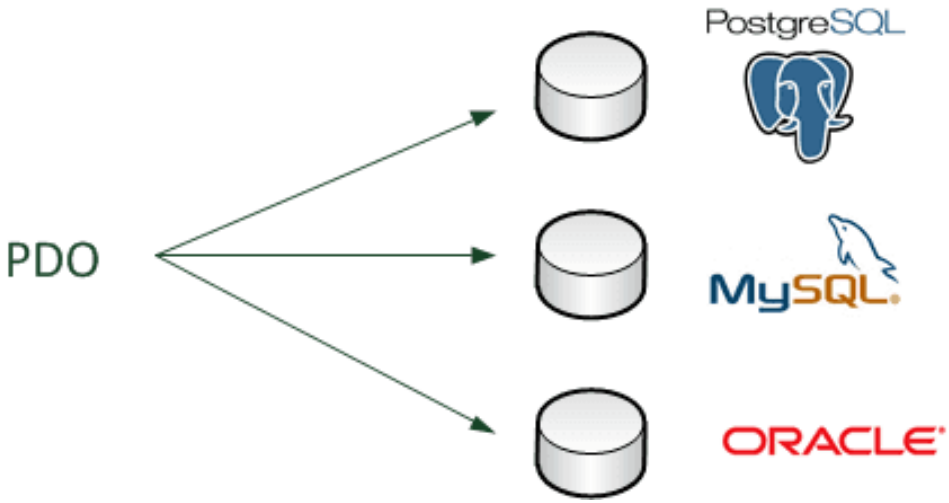



```php
<?php
class MyClass {
  public function hello() {
    echo "Hello World!";
  }
}

class AnotherClass extends MyClass {
}

$obj = new AnotherClass();
$obj->hello();
?>
```

PDO (PHP Data Objects) : Connexion

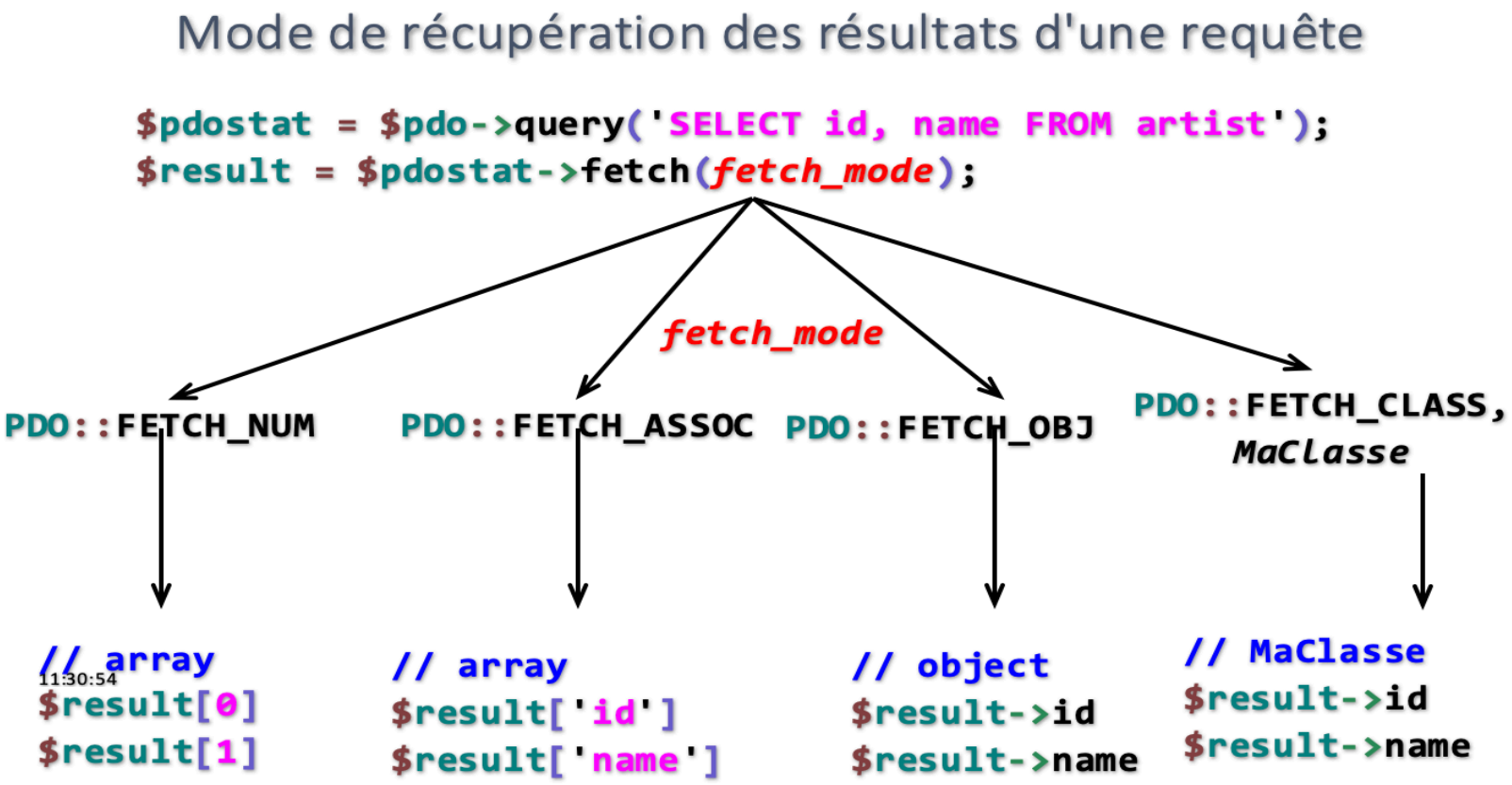
- **Connexion à la base de données**
  - Connecteur utiliser pour la connexion entre PHP et les Bases de données
  - Pour se connecter il faut instancier un objet de la classe PDO en passant au constructeur

```php
<?php
try {
    $dbh = new PDO('mysql:host=dbs;dbname=music', $user, $pass);
}
catch (PDOException $e) {
    echo '<p>Erreur de base de données';
    die();
}
```



PDO (PHP Data Objects) : Requête non préparée

- **Exécution et Exploitation des résultats**
  - Récupération des données ligne à ligne
  - Récupération des données d'une colonne



PDO (PHP Data Objects) : Requête non préparée

- **Exécution et Exploitation des résultats**

- **`PDO::FETCH_ASSOC`**
  - Retourner chaque ligne dans un tableau indexé par les noms des colonnes comme elles sont retournées dans le jeu de résultats correspondant.
- **`PDO::FETCH_NUM`**
  - retourner chaque ligne dans un tableau indexé par le numéro des colonnes comme elles sont retournées dans le jeu de résultats correspondant, en commençant à 0.
- **`PDO::FETCH_CLASS`**
  - retourner une nouvelle instance de la classe demandée, liant les colonnes aux propriétés nommées dans la classe.
- **`PDO::FETCH_BOTH`** (*par défaut*)
  - retourner chaque ligne dans un tableau indexé par les noms des colonnes ainsi que leurs numéros en commençant à 0.
- **`PDO::FETCH_OBJ`**
  - retourner chaque ligne dans un objet avec les noms de propriétés correspondant aux noms des colonnes comme elles sont retournées dans le jeu de résultats.

PDO (PHP Data Objects) : Requête non préparée

- **Exécution et Exploitation des résultats**
  - Récupération des données d'une colonne : **FETCH_ASSOC**

```
try {
    $pdo = new PDO('mysql:host=dbs;dbname=UASZ');
    $pdo->setAttribute(PDO::ATTR_ERRMODE,
                       PDO::ERRMODE_EXCEPTION);
    $pdostat = $pdo->query('SELECT Mat, prenom FROM Etudiants ');
    $pdostat->setFetchMode(PDO::FETCH_ASSOC);
    foreach ($pdostat as $ligne) {
        echo '<p>' . $ligne['prenom'] . "\n";
    }
}
catch (Exception $e) {
    echo '<p>Erreur de base de données';
}
```

PDO (PHP Data Objects) : Requête non préparée

- **Exécution et Exploitation des résultats**
  - Récupération des données d'une colonne : **FETCH_NUM**

```
try {
    $pdo = new PDO('mysql:host=dbs;dbname=UASZ');
    $pdo->setAttribute(PDO::ATTR_ERRMODE,
                       PDO::ERRMODE_EXCEPTION);
    $pdostat = $pdo->query('SELECT Mat, prenom FROM Etudiants ');
    $pdostat->setFetchMode(PDO::FETCH_NUM);
    foreach ($pdostat as $ligne) {
        echo '<p>' . $ligne[1] . "\n";
    }
}
catch (Exception $e) {
    echo '<p>Erreur de base de données';
}
```

PDO (PHP Data Objects) : Requête non préparée

- **Exécution et Exploitation des résultats**
  - Récupération des données ligne à ligne
  - Récupération des données d'une colonne : **FETCH_CLASS**

```php
$stmt = $pdo->query(<<<SQL
    SELECT id, name
    FROM artist
    WHERE id = 12
SQL
);
$stmt->setFetchMode(PDO::FETCH_CLASS, Artist::class);
if (($object = $stmt->fetch()) !== false) {
    return $object;
}
```

Instancie un objet de la classe `Artist` dont les attributs sont supposés être `id` et `name`, exactement le nom des champs de la table

PDO (PHP Data Objects) : Requête non préparée

- **Exécution et Exploitation des résultats**
  - Récupération des données ligne à ligne
  - Récupération des données d'une colonne : **FETCH_OBJECT**

```
$stmt = $pdo->query(<<<SQL
    SELECT id, name
    FROM artist
    WHERE id = 12
SQL
);
```

Instancie un objet de la classe `Artist` dont les attributs sont supposés être `id` et `name`, exactement le nom des champs de la table

```
if (($object = $stmt->fetchObject(Artist::class)) !== false) {
    return $object;
}
```

PDO (PHP Data Objects) : Requête préparée

Avec PDO, une requête doit toujours être préparée !

- **Préparation et exécution d'une requête**
- **NB : Intérêt des requêtes préparées**
  - Amélioration des performances en cas d'exécutions répétées
  - Émulation faite par **PDO** si le driver ne les supporte pas nativement
  - Protection automatique des valeurs des paramètres pour interdire les attaques par injection de code SQL

PDO::prepare(*string statement* [,*array driver_options*]): PDOStatement

- ***statement*** : la requête à préparer.
  Peut contenir des paramètres anonymes (**?**) ou nommés (**:nom**)
- ***driver_options*** : tableau d'options du driver
- retourne un objet **PDOStatement** qui effectuera l'association des paramètres et exécutera la requête

```
$pdo = new PDO('mysql:host=localhost;dbname=music');
$pdostat = $pdo->prepare('SELECT id, name FROM artist WHERE id = ?');
```

PDO (PHP Data Objects) : Requête préparée

- **Préparation et exécution d'une requête**

```php
$pdo = new PDO('mysql:host=dbs;dbname=app');
$pdo->setAttribute(PDO::ATTR_ERRMODE,
                   PDO::ERRMODE_EXCEPTION);
$pdostat = $pdo->prepare('SELECT * FROM user
                          WHERE login = ?');
$pdostat->bindValue(1, 'root');
$pdostat->execute();
// Utilisation du résultat

$pdostat->bindValue(1, 'cutrona');
$pdostat->execute();
// Utilisation du résultat
```

paramètre anonyme

PDO (PHP Data Objects) : Requête préparée

- **Préparation et exécution d'une requête**

```php
$pdo = new PDO('mysql:host=dbs;dbname=app');
$pdo->setAttribute(PDO::ATTR_ERRMODE,
                   PDO::ERRMODE_EXCEPTION);
$pdostat = $pdo->prepare('SELECT * FROM user
                          WHERE login = ?');
$pdostat->execute( ['root'] );
// Utilisation du résultat
$pdostat->setFetchMode(PDO::FETCH_NUM);
    foreach ($pdostat as $ligne) {
        echo '<p>' . $ligne[1] . "\n";
    }
$pdostat->execute( ['cutrona'] );
// Utilisation du résultat
```

paramètre anonyme

PDO (PHP Data Objects) : Requête préparée

- **Préparation et exécution d'une requête**

```php
$pdo = new PDO('mysql:host=dbs;dbname=app');
$pdo->setAttribute(PDO::ATTR_ERRMODE,
                   PDO::ERRMODE_EXCEPTION);
$pdostat = $pdo->prepare('SELECT * FROM user
                          WHERE login = :utilisateur');
$pdostat->bindValue(':utilisateur', 'root');
$pdostat->execute();
// Utilisation du résultat
$pdostat->setFetchMode(PDO::FETCH_ASSOC);
    foreach ($pdostat as $ligne) {
        echo '<p>' . $ligne['prenom'] . "\n";
    }
$pdostat->bindValue(':utilisateur', 'cutrona');
$pdostat->execute();
// Utilisation du résultat
```

paramètre nommé

PDO (PHP Data Objects) : Requête préparée

- **Préparation et exécution d'une requête**

```php
$pdo = new PDO('mysql:host=dbs;dbname=app');
$pdo->setAttribute(PDO::ATTR_ERRMODE,
                   PDO::ERRMODE_EXCEPTION);
$pdostat = $pdo->prepare('SELECT * FROM user
                          WHERE login = :utilisateur');
$pdostat->execute( [':utilisateur' => 'root'] );
// Utilisation du résultat
$pdostat->setFetchMode(PDO::FETCH_ASSOC);
    foreach ($pdostat as $ligne) {
        echo '<p>' . $ligne[’prenom'] . "\n;
    }
$pdostat->execute( [':utilisateur' => 'cutrona'] );
// Utilisation du résultat
```

paramètre nommé

PDO (PHP Data Objects) : Attaque par injection

- Exemple : validation d'un login/password sur un site
- Requête consistant à trouver un enregistrement correspondant au couple login/password fourni par l'utilisateur
- Requête naïve :

- ```
  SELECT *
  FROM utilisateurs
  WHERE login='{$_POST['login']}'
    AND Mot_passe=SHA1('{$_POST['passwd']}')
  ```

- Et si on essayait de fournir un mot de passe un peu particulier…
  ```

PDO (PHP Data Objects) : Attaque par injection

EXEMPLE D'UNE ATTAQUE PAR INJECTION

```
$pdo = new PDO('mysql:host=dbs;dbname=app');
$pdostat = $pdo->query($req = <<<SQL
    SELECT *
    FROM utilisateurs
    WHERE login      = '{$_POST['login']}'
      AND mot_passe= SHA1('{$_POST[’Mot_passe']}')
SQL
    );
echo "Requête:\n$req\n";
if (($utilisateur = $pdostat->fetch())) !== false)
    { echo "Bienvenue {$utilisateur['nom']}\n"; }
else { echo "Désolé...\n"; }
```

PDO (PHP Data Objects) : Attaque par injection

Saisie de l'utilisateur par formulaire :

- login : `whatever`
- pass : `who_cares?`

```
Requête:
    SELECT *
    FROM utilisateurs
    WHERE login='whatever'
      AND mot_passe =SHA1('who_cares?')
 Désolé...
```

PDO (PHP Data Objects) : Attaque par injection

Saisie de l'utilisateur :

- login : whatever
- pass : who_cares?') OR (true!='

Donne toutes les lignes de la table des utilisateurs de votre plateforme

```
Requête:
    SELECT *
    FROM utilisateurs
    WHERE login='whatever'
   AND mot_passe =SHA1('who_cares?')                    ue!='')
Bienvenue John
```

PDO (PHP Data Objects) : Protection aux injections

```php
$pdo = new PDO('mysql:host=dbs;dbname=app');
$login  = $pdo->quote($_POST['login']);
$passwd = $pdo->quote($_POST['passwd']);
$pdostat = $pdo->query($req = <<<SQL
    SELECT *
    FROM membre
    WHERE login       = $login
      AND mot_passe = SHA1($passwd)
SQL
 );
echo "Requête:\n$req\n";
if (($utilisateur = $pdostat->fetch()) !== false)
     { echo "Bienvenue {$utilisateur['nom']}\n"; }
else { echo "Désolé...\n; }
```

PDO (PHP Data Objects) : Protection aux injections

```
$pdo = new PDO('mysql:host=dbs;dbname=app');
$login = $pdo->quote($_POST['login']);
$passwd = $pdo->quote($_POST['passwd']);
$req = <<<SQL
SELECT *
FROM utilisateurs
WHERE login=$login
AND mot_passe =SHA1($passwd)
SQL
);
echo "Requête:\n$req\n";
if (($utilisateur = $pdost
    { echo "Bienvenue {$utilis
else { echo "Désolé...\n"; }
```

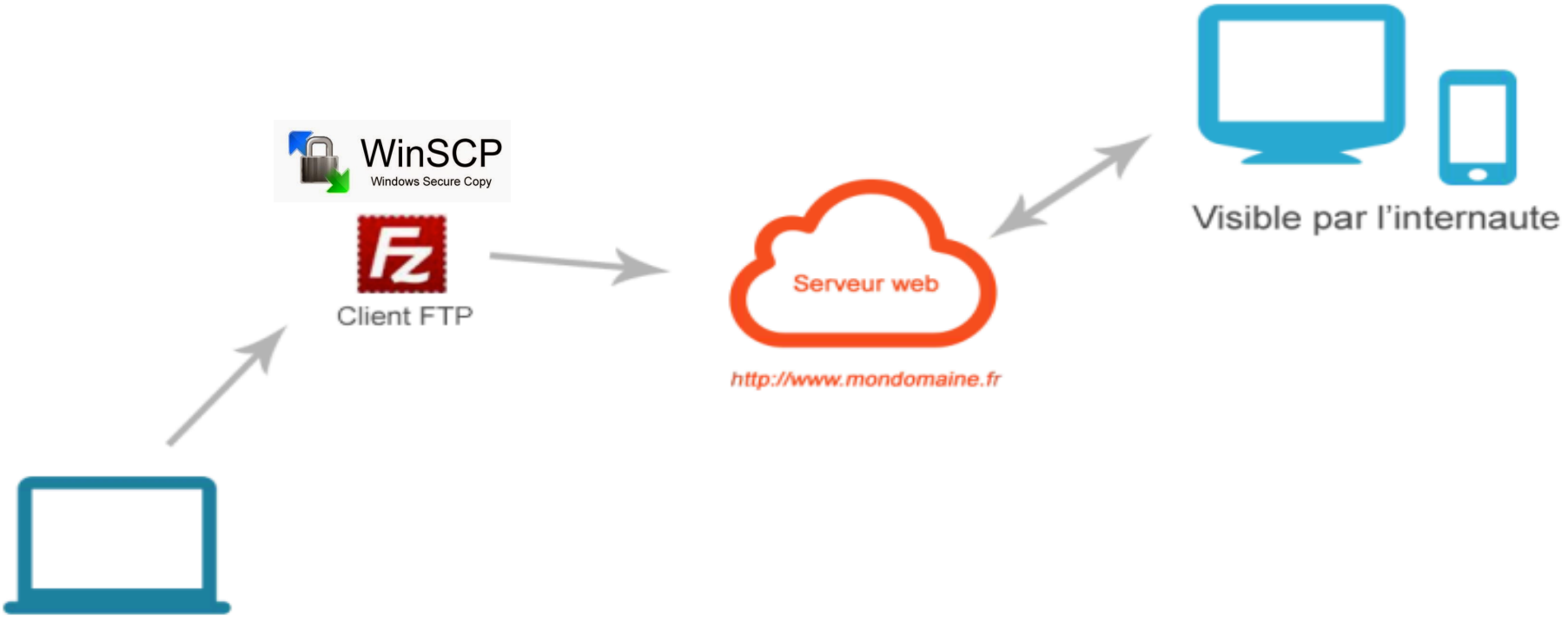
NON utilisé en pratique
ICI, uniquement pour montrer ce qu'il se passe

```
Requête:
SELECT *
FROM utilisateurs
WHERE login='whatever'
AND mot_passe =SHA1('who_cares?|') OR true!=|'')
```

Partie 3 :
Mise en ligne

La mise en ligne

- Consiste à Uploader (Charger) les fichiers de sa plateforme dans un hébergeur (Serveur) web **souvent à l'aide**
  - Client FTP : Filezilla ou WinSCP
  - Protocole de transfert de fichier FTP (**File Transfer Protocol)**
  - Port 21



La mise en ligne