Université Paris Nanterre

Université Paris Nanterre

Programmation web et Introduction à PHP

L2 MIASHS

M. NAFI

m.nafi@parisnanterre.fr

Année: 2022/2023

Objectifs du cours

- Création de **sites web statiques, interactifs** et **dynamiques** à l'aide de:
  - Langage **html**
  - Langage **CSS**
  - Langage **JavaScript**
  - Langage **PHP**

Plan du cours

1. Introduction
2. Langage Html
3. Langage CSS
4. Langage Javascript
5. Langage PHP

Plan du cours

1. **Introduction**
2. Langage HTML
3. Langage CSS
4. Langage Javascript
5. Langage PHP

Introduction

❑ Internet vs Web

- 02 termes utilisés de façon **interchangeable**
- Souvent **confondus** mais **ne désignent pas** la même chose
- Internet et le World Wide Web (www ou web) sont deux concepts **liés** mais **distincts**.

❑ **Internet** (1969 (ARPANET) DARPA)

*(**D**efense **A**dvanced **R**esearch **P**rojects **A**gency)*

- Internet= **Inter**connected + **net**works
- Internet est un **ensemble de réseaux** informatiques reliés entre eux (réseau des réseaux)
- C'est le réseau **mondial** d'ordinateurs interconnectés qui communiquent en utilisant des protocoles.

Internet

- **Infrastructure** physique de réseaux
- Internet offre de nombreux **services** :
  - Courrier électronique (POP, IMAP, SMTP)
  - Messagerie instantanée (SIP)
  - Transfert de fichiers (FTP)
  - **Web** (HTTP)

❑ **Web** (World Wide Web ou www) (1989 CERN)

*(**C**onseil **E**uropéen pour la **R**echerche **N**ucléaire)*

- Le **web** est une des applications ou services d'Internet.
- Une collection de documents ou pages web et ressources liés par des hyperliens ou liens hypertextes
- Le **web** et d'autres applications telles que la messagerie instantanée, le courrier électronique, le transfert de fichiers, reposent sur Internet.

Internet , Intranet, Extranet

- Internet. Réseau **public** qui connecte des dispositifs à travers le monde
- Nombre illimité d'utilisateurs (n'importe qui peut y accéder)

Intranet

- Intranet. Réseau **privé** qui interconnecte des ordinateurs dans une organisation ou une entreprise
- Utilise les **firewall** (pare-feu) pour empêcher les utilisateurs externes d'accéder aux ressources du réseau
- Utilisé pour partager des données sensibles qui ne doivent pas être accessibles à l'extérieur de l'entreprise
- Nombre limité d'utilisateurs (seul les employés y accèdent)

Extranet

- Intranet mais accessible depuis n'importe quel appareil connecté à Internet par **authentification** ou **identification** (mot de passe)
- Partage d'information avec les acteurs externes à l'entreprise tels que les clients, les fournisseurs, etc

=> permet d'ouvrir le système d'informations d'une entreprise à des partenaires extérieurs.

Sites web

- Un site web est un ensemble de pages web et de ressources reliées par des hyperliens
- Les sites web sont hébergés sur des serveurs, accessibles via une adresse URL (*Uniform Resource Locator*)
- Un ensemble de fichiers, hébergés sur un **serveur**
- Une page web est un document dans lequel on trouve du texte, des images, listes, menus, etc.
- Les sites web peuvent être statiques ou dynamiques

❑ Site statique

- C'est un site qui affiche le même contenu pour **tous** les utilisateurs. Le contenu des pages ne changent pas sauf par l'administrateur.
- Pages sont codées en HTML, CSS et Javascript pas de code exécuté coté serveur.
- **Ex.** Sites vitrines utilisés pour présenter des produits ou des services.

❑ Site dynamique

- C'est un site qui affiche des informations qui **changent** en fonction de l'utilisateur. Le contenu est généré par le serveur en utilisant des technologies telles que PHP, Python, etc.

- **Ex.** e-commerce (vente de services ou produits en ligne), réseaux sociaux, site nécessitant des mise à jour fréquentes (bourse), etc.

Modèle Client/serveur (1/2)

- Mode de communication informatique largement utilisé dans les sites dynamiques

  - **Client** (Navigateur)

    demande des pages web (requêtes HTTP)

    comprend les langages **html**, **css** et **Javascript**

Modèle Client/serveur (2/2)

- **Serveur** Tout le temps en écoute

- Traite les requêtes reçues et génère des pages web (codes html et css)
- renvoie des réponses HTTP (pages html)
- Sait interpréter le langage **PHP**, **Python**, etc

Ex. Appache, IIS (**I**nternet **I**nformation **S**ervices)

Navigateur

- Un logiciel qui permet de naviguer sur Internet
- Le **moteur** ou l'**agent utilisateur** est chargé de l'interprétation du code HTML, CSS et JavaScript.
- Plusieurs navigateurs peuvent se partager le même moteur
- **Ex.** Chrome, Safari et Android ont le même moteur **WebKit**
- Ex. **Propriétaires** (Internet explorer ou Microsoft Edge, Opéra)

  Open source (Firefox, Google Chrome, Android)

Protocole HTTP (1/4)

- Le **dialogue** entre le client et le serveur se fait avec le protocole de communication HTTP(HyperText Transfer Protocol)
- Il est utilisé pour la transmission de données sur Internet.
- Il définit un ensemble de règles pour la transmission de données
- HTTP est un protocole de communication conçu pour formuler des demandes (requêtes) et transférer des réponses et contenu des pages web.

Protocole HTTP (2/4)

- Requêtes: les pages sont demandées avec les méthodes **GET** ou **POST**
- Chaque requête HTTP débute par une ligne contenant: une **commande**, **emplacement** de la ressource, **HTTP/ numéro de version**
- Ex. GET /HTTP/1.1

  Host: www.google.com

  …

Protocole HTTP (3/4)

- Le serveur renvoie une réponse HTTP contenant:

Le nom du **protocole**, sa **version**, le **code d'état**

Ex. HTTP/1.1 200ok

Date: …

Serveur: appache/2

…

Protocole HTTP (4/4)

- HTTP définit un ensemble de codes d'état:

  1xx : Information

  2xx : Succès

  3xx : Redirection

  4xx : Erreur du client HTTP

  5xx : Erreur du serveur

- Ex. **200 ok :** tout s'est bien passé

  404: Page non trouvée

URL (Uniform resource locator)

- Une adresse qui permet de **localiser** un document ou une ressource sur Internet
- Souvent affichée dans la barre d'adresse des navigateurs web
- Chaque page ou ressource dans le web dispose d'une adresse appelé URL
- Une URL est composée de plusieurs éléments:
  - Protocole. http ou https (HyperText Transfer Protocol Secure)
  - Nom de domaine. www.google.fr
  - Chemin de la ressource. /path/index.html
  - Paramètres de requêtes ?param1=value1

Communication client/serveur

- Demander une page web en tapant une URL ou en cliquant sur un lien
- Le navigateur envoie au serveur une requête HTTP
- Le serveur recherche la page et renvoie une réponse HTTP
- Le navigateur parse (parcourt) le document HTML. Si la page contient d'autres fichiers (image, script, style), il recontacte le serveur
- La page est enfin réassemblée et affichée dans le navigateur

Conclusion

- Internet vs web
- Internet, Intranet, Extranet
- Sites web et leur types
- Client/Serveur
- Navigateur web
- Protocole HTTP
- Adresse web ou Url
- Communication Client/Serveur

Université Paris Nanterre

Université Paris Nanterre

Programmation web et Introduction à PHP

L2 MIASHS

M. NAFI

m.nafi@parisnanterre.fr

Année: 2022/2023

Objectifs du cours

- Création de **sites web statiques, interactifs** et **dynamiques** à l'aide du:
  - Langage **HTML**
  - Langage **CSS**
  - Langage **JavaScript**
  - Langage **PHP**

Plan du cours

1. Introduction
2. Langage Html
3. Langage CSS
4. Langage Javascript
5. Langage PHP

Plan du cours

1. Introduction
2. **Langage HTML**
3. Langage CSS
4. Langage Javascript
5. Langage PHP

Langage HTML

Historique

- HTML a évolué à partir de SGML (**S**tandard **G**eneralized **M**arkup **L**anguage). Un méta langage de la norme ISO

- À la fin des années **80**, le britanique Tim Berners-Lee, employé au CERN (suisse), developpa un langage de balisage qui donna naissance à HTML.

Définition

- HTML (**H**yper**T**ext **M**arkup **L**anguage) (langage de **balisage,** liens **hypertextes** ou hyperliens)

- HTML est un langage de **description** qui fait appel aux balises pour structurer le document

- Il permet de définir et de structurer les éléments d'un document textuel à l'aide de titres, paragraphes, listes, tableaux, etc
- Permet d'ajouter du **contenu** aux pages web

- Le code HTML est interprété par des agents utilisateurs au niveau des navigateurs
- Une page web est un fichier texte ayant l'extension **.html**
- Le code source des pages web est **gratuit** et **accessible**

Structure de base d'un document HTML

```
<!DOCTYPE html>          // Type et la version du document
<html>                   // l'élément racine
    <head>
        <meta charset="utf-8">          //En-tête
        <title> Titre de la page </title>
        …
    </head>
    <body>
        <!– le contenu de la page-->
        <p>ceci est un paragraphe</p>     // Corps
        <a href='www.exemple.com' > accéder au site</a>
        …
    </body>
</html>
```

DOCTYPE

- Un document HTML débute toujours par une ligne nommée DOCTYPE, avant même la première balise **<html>**
- Cette déclaration renseigne le navigateur sur le type du document et la version du HTML utilisée
- **Ex.** <!doctype html> document HTML5
- La version 5 est simplifiée par rapport aux précédentes

  Ex. HTML 4.01

  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML4.01//EN"http://www.w3.org/TR/html4/strict.dtd">

Elément, balise et attribut

❑ Elément <*NomElement*> Contenu </*NomElement*>

- Un élément peut ne pas avoir de balise fermante (Ex. img, link, base)
- 02 types d'éléments:
  - **Bloc:** retour automatique à la ligne (<p>, , …)
  - **Ligne:** pas de retour à la ligne (, , …)
- Un élément en bloc peut contenir d'autres éléments en bloc ou en ligne
- Cependant, un élément en ligne ne peut contenir que des éléments en ligne

Élément

- Les éléments peuvent se **succéder**

Ex. <h1> …</h1> …

- ou **s'imbriquer**

Ex. ………

- doit posséder un sous élément <li> (élément enfant) **Ex**.

```
<ul>
    <li> </li>
    <li></li>
</ul>
```

❑ Balise

- Forme concrète d'un élément et permet de structurer le document
- Une balise débute par le chevron ou le signe < et finit par le signe >
- **Deux** types de balises:
  - Balise **ouvrante**: <*NomElement*>
  - Balise **fermante**: </ *NomElement* > le nom de l'élément est précédé du caractère **slash (/)**

Ex. <html> … </html>

❑ Attribut

- **Propriété** d'un élément (nom= "valeur")
- Modifie les propriétés des balises
- Un élément peut comporter **zéro** ou **plusieurs** attributs
- Plusieurs attributs sont séparés par des espaces
- Les attributs font partie de la balise ouvrante
- Certains attributs sont obligatoires comme **src** dans la balise <img src="url">, d'autres sont facultatifs comme alt.

Ex. Type="Text"
href="www.parisnanterre.fr"

Elément racine

- **<html>** est l'élément **racine** de tout document HTML
- Cet élément possède deux enfants directs:

<head> et <body>

- L'attribut global applicable à cet élément est lang qui spécifie la *langue* (si le champs est vide ie langue est inconnue)
- **Ex.** <html lang="fr">

En-tête (head) (1/2)

- Fournit de multiples informations sur le document (Titre, métadonnées (auteur, mots clés, description, etc), liens, style, script, base)
- Ces informations ne sont pas affichées dans le navigateur excepté le **titre**
- À ne pas confondre avec les éléments *heading (titres)* <h1> à <h6>

En-tête (2/2)

- Dans un en-tête, on trouve:
  - Titre du document: <title>..</title>
  - Métadonnées: <meta … />
  - Liens: <link>
  - Style: <style> </style>
  - Script: <script></script>
  - Base: <base>

Titre

- Le titre du document est affiché par le navigateur dans la **barre de titre** ou **l'onglet actif**
- Le titre est utilisé comme intitulé des liens par les moteurs de recherche
- Il est introduit par les balises **<title>** Titre du document **</title>**
- Le titre est le seul élément **obligatoire** dans l'élément <head> d'un document

Métadonnées

- La balise **meta** est utilisée pour fournir davantage d'informations sur le document telles que le nom de l'auteur, la description du contenu, les mots clés, etc
- Cette balise ne possède pas de balise **fermante**
- Ces informations ne seront pas affichées dans la page
- On distingue plusieurs applications:
  - Attribut name et content
  - Attribut http-equiv et content
  - Attribut charset

meta name

- Mots clés: <meta name='**keywords**' content='html, css ,Js'>
- Description: <meta name='**description**' content='Le cours de HTML, CSS et Js'>
- Auteur: <meta name="**author**" content="M. NAFI">
- Dimension de la page: <meta name="**viewport**" content="width=300, initial-scale=1">

meta http-equiv

- **Rafraichissement** de la page

<meta http-equiv="refresh" content= "30" />
(actualisation de la page toute les 30 secondes)

- **Redirection** vers une URL après un délai exprimé en secondes:

<meta http-equiv="refresh" content= "30; url= www.google.fr" />

meta charset

- **Jeu de caractère** spécifie l'encodage de caractères utilisé
- Les valeurs les plus utilisées sont: UTF-8, ISO-8859-1, ASCII

Ex. <meta **charset**= "UTF-8">

Style

- La balise <style> est utilisée pour ajouter à la page du code **CSS interne**
- Elle possède les attributs **type** et **media**
- **Ex.** <style type='text/css'>

```
p {
    color : red;
}
</style>
```

<style **media**="screen"> … </style>

<style **media**="print"> … </style>

Script (Javascript)

- Le code JavaScript est introduit avec l'élément <script>…</script>
- Code interne
- **Ex.** <script>

```
document.getElementById("ID").innerHTML = "Bonjour!";
</script>
```

- Code externe

  Ex.

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

Liens (1/2)

- Insérer un lien vers d'autre pages ou ressources externes comme un fichier (css, js,…) à l'aide de la balise **<link>**
- Cet élément ne possède pas de balise fermante
- Il possède les attributs :
  - **rel: relation**
  - **type: type du fichier**
  - **href:** Adresse absolue ou relative de la cible

Liens (2/2)

- Adresse relative => 03 cas peuvent se présenter:
  - Fichier se trouve dans **le <u>même</u>** dossier que la page (le répertoire courant)

Ex. <link rel="stylesheet" type="text/css " href="**style.css** " >

  - Fichier se trouve dans un **<u>sous</u>** <u>dossier</u>(niveau **inférieur**)

Ex.<link rel= "stylesheet " type="text/css"
href="**dossier/**style.css " > 1 seul niveau

  - Fichier se trouve dans un dossier **parent** (niveau **<u>supérieur)</u>**

Ex.<link rel="stylesheet" type="text/css" href="**../**style.css">
1 seul niveau

Base

- L'élément <base> définit l'URL de base pour tous les liens relatifs dans une page web
- Il ne peut y avoir qu'un seul élément base dans un document
- Si plusieurs balises <base> sont utilisées, seule la première sera prise en compte par les navigateurs web.
- Ex.

```
<base href="https://www.google.com/">
```

Corps du document html

- Le corps d'un document html est défini par la balise <body>…</body>
- Il peut être structuré comme suit:
  - En-tête: <header >..</header >
  - Menu de navigation: < nav>..</ nav>
  - Contenu principal: < main>…</main >
  - Section: <section >..</ section>
  - Article: < article>..</article >
  - Pied de page: < footer>..</ footer>
- Chacun de ces éléments peut avoir des titres, paragraphes, images, liens, tableaux, formulaires, etc.

Titres

- Définir des titres et sous titres (*headings*)
- Il existe 6 niveaux: h1 à h6
- <h1> le plus **important**: les moteurs de recherche leur affectent un poids plus **élevé**
- <h6> le moins important
- Par défaut, les titres sont affichés en **gras**
- **Ex.**

  <h1> Titre de niveau 1 <h1>

  <h3> Titre de niveau 3 <h3>

Paragraphe

- **Paragraphe** … permet d'ajouter du texte à la page
  - Par défaut, une ligne **avant** et **après** le paragraphe seront affichées par les navigateurs
  - Un paragraphe peut contenir du **texte**, des **images** et les autres éléments en **ligne** (, <b>, etc).

- **Séparateurs** :
  -
 : retour à la ligne
  - <hr> : ligne horizontale

Formatage du texte

- **Gras** ou texte important: … , …
- **Italic**: …,
- Mise en **évidence**: <mark>…</mark>
- Mise en **indice** : _…
- Mise en **exposant** : […]
- **Citation**: <cite>…</cite>
- etc

Liens (ancres) (1/3)

- Un lien hypertexte vers un endroit précis d'une page ou vers une autre page se fait à l'aide de l'élément anchor <a>…
- L'attribut **href** ((**h**ypertext **ref**erence) est utilisé pour définir l'adresse ou l'URL de la page cible

Ex.

 lien vers un endroit précis d'une page (section)

 lien vers une autre page

Liens (2/3)

Deux types d'URLs: absolue et relative

- **Absolue**: adresse complète:

  protocole, **nom du domaine**, **chemin** si nécessaire

- Utilisée pour faire référence à une ressource externe (serveur ou site différent) ie lien externe
- **Ex.** Le site de l'université https => aller sur le web

Liens (3/3)

- **Relative**: chemin vers un fichier ie lien **interne**
  - Sans le http, le navigateur regarde sur le serveur **local**
  - Utilisée pour faire référence à une ressource interne (*même serveur* ou site)
    - Lien à l'intérieur d'un dossier
      - Ex. Page d'acceuil
    - Lien vers un **sous** dossier (1 niveau)
      - Ex. Aide

Liens (3/3)

- Lien vers un dossier **parent** (1 niveau)
- Ex. Page d'acceuil
- Lien à partir de la racine, on ajoute un "/"

- Nb: on peut mettre n'importe quel élément html entre <a> et pour le transformer en un lien
- Ex. Une image comme lien:
- <img src="logo.gif" alt=" logo">

Images

- L'insertion d'images se fait à l'aide de la balise <img>
- Cet élément ne possède pas de balise fermante
- La balise <img> possède les attributs suivants:
  - **src** : la **source** de l'image ou le chemin vers l'image (obligatoire)
  - **height**: **hauteur** de l'image (px, em, %, etc)
  - **width**: spécifie la **largeur** de l'image (px, em, %, etc)
  - **alt**: texte **alternatif** en cas d'échec de l'affichage ou chargement de l'image
- **Ex.** <img src="Image.jpg" height="100px" width="100px" alt="Image">

Liste

- 03 types de listes :
- Non ordonnée. l'ordre des items n'est pas important.
- Ordonnée. l'ordre des items est important.
- Définition ou description. Elle consiste en paires (nom, valeur)

Liste non ordonnée

- Liste non ordonnée est définie avec la balise
- Cette balise possède des items **<li>**
- Les items ** ne** sont **pas** ordonnés (précédés par des puces par défaut) => l'ordre n'est pas important
- Seul des <li> sont permis entre <ul> et
- **Ex.**

```
<ul>
        <li> HTML</li>
        <li> CSS </li>
        <li> JavaScript </li>
</ul>
```

Liste ordonnée

- Liste ordonnée est introduite avec la balise
- Liste où l'ordre est **important** (Ex. recette de cuisine, liste des étudiants admis selon l'ordre de mérite, etc)
- À la place des puces, le navigateur insère des nombres (par défaut) ou des lettres
- Les items **<li>** sont ordonnés avec des chiffres ou lettres alphabétiques
- **Ex.** <ol type= "a" >
   HTML => Liste avec des lettres
   CSS

Remarque

- Si l'on souhaite que la liste commence à partir d'un nombre qui est diffèrent de "1", on utilise l'attribut **start** comme suit:
- Ex. <ol start= "5 "> //commence à partir de 5

```
        <li> HTML </li>
        <li> CSS </li>
        <li> JavaScript </li>
</ol>
```

Liste de description (1/2)

- Liste de *description* est définie avec la balise <dl> </dl>
- Un élément <dl> contient un certain nombre de d' éléments <dt> et leur <dd> respectifs
- Le **nom** ou le **terme** est introduit avec <dt></dt>
- La **valeur** ou la définition est introduite avec <dd> </dd>

Liste de description (2/2)

- **Ex.** <dl>
  <dt> HTML </dt>
  <dd> Langage de balisage </dd>
  <dt> PHP </dt>
  <dd> Langage de script </dd>
  </dl>
- NB: Pour un même terme <dt>, on peut avoir plusieurs définitions <dd>

Tableau (1/3)

- Pour créer un tableau, on utilise la balise <table>
- Cet élément possède une balise ouvrante <table> et fermante </table>
- En HTML, un tableau consiste en un ensemble de **lignes**
- Chaque ligne est introduite à l'aide de la balise **<tr>**
- <th>: permet d'insérer l'entête du tableau
- <td>: permet d'insérer les données du tableau

Tableau (2/3)

- Ex.

```
<table>
    <tr>
        <th> Nom </th>
        <th> Age </th>
    </tr>
    <tr>
        <td> Jean </td>
        <td> 30 </td>
    </tr>
</table>
```

Tableau à 2 lignes et 2 colonnes

Nom	Age
Jean	30

Tableau (3/3)

- Attributs:
  - **colspan**: fusion de cellules d'une même ligne en définissant la valeur de l'attribut colspan d'un élément <td> ou <th> avec un entier
- Cet entier indique le nombre de cellules à fusionner en partant de la gauche
- Ex. <td colspan="N">Contenu de la cellule</td> **N** cellules
  - **rowspan**: fusion de cellules situées dans les lignes ***adjacentes***
- Ex. <td rowspan="N"> Contenu de la cellule</td>

Formulaire (1/9)

- Un formulaire est introduit avec l'élément <form>…</form> et peut contenir des champs de saisie, des boutons, textes, etc
- Il permet de **collecter** les informations des utilisateurs
- Les informations collectées sont traitées par un **script** ou une application au niveau du serveur
- Cet élément peut contenir d'autre éléments en bloc mais **ne** peut pas contenir d'autres élément <form>

Formulaire (2/9)

- L'élément <form> possède les attributs: action, method
  - **action**=" URL " l'adresse du script ou de l'application qui traitera les données saisies
  - **action**="": Si le code PHP se trouve sur la page elle-même
- Ex. <form action="/script.php" >…</form>

Formulaire (3/9)

- method: spécifie comment l'information devrait être envoyée au serveur. Il existe deux méthodes: GET et POST
- Cet attribut est **optionnel**. S'il n'est pas spécifié, la méthode GET est choisie par défaut

Formulaire (4/9)

- GET vs POST
  - GET: données saisies sont insérées dans l'URL après « ? »
    - Nombre de caractères à envoyer est limité
    - Données sont visibles à tous
  - POST: données insérées dans une requête HTTP
    - Nombre de caractères est illimité
    - Sécurisé: seul le serveur est capable de lire les données

Formulaire (5/9)

- Types de **contrôles**: la plupart des éléments sont introduits avec l'attribut « type ». **L'apparence** de l'élément change en fonction de la valeur de « type »
- Plusieurs types de contrôles en HTML5:
  - Champs de texte
    - Monoligne: <input type=" text " name= " nom " value= "Jean" placeholder='Text' maxlength= "25" >
      - **name**: nom de l'élément (obligatoire)
      - **value**: valeur par défaut lors du chargement de la page
      - **placeholder**: son contenu n'est pas envoyé
      - **maxlength** et **minlength**: nombre de caractères

Formulaire (6/9)

- Multilignes: l'élément <textarea> est utilisé lorsque l'utilisateur souhaite saisir plus d'une ligne
- Ex. <textarea name= "adresse " placeholder=" pas plus de 50 caractères " rows= "50 " cols= "5">
- *placeholder*: Son contenu ne sera pas émis au serveur
- *content*: Son contenu sera transmis au serveur

Formulaire (7/9)

- **Champs de texte particuliers:**

- Mot de passe: <input type= " password " …>

  Texte **invisible** mais ne veut pas dire chiffré
- Recherche: <input type= "search " …>
- Email: <input type= " email " …>
- Téléphone : <input type= " tel " …>
- Adresse mail: <input type= "url " …>

Formulaire (8/9)

- Bouton: <input type= " button " …>
  <input type= " radio " …>
  <input type= " checkbox " …>
- Menu: <datalist id="lang">
  <option value="HTML">
  <option value="CSS">
  </datalist>
  <select name="nom">
  <option>HTML</option>
  <option> CSS</option>
  </select>

Formulaire (9/9)

- Fichier: <input type="file" …>
- Date et heure: <input type="date" …>
  <input type="time" …>
- Couleur: <input type="color" …>
- etc

Division (1/2)

- L'élément est utilisé pour créer une division ou section dans un document html
- Cet élément **n'a pas** de signification propre en HTML mais utilisé conjointement avec le CSS
- C'est un élément par **bloc** ie débute et finit par un retour à la ligne
- **Ex.** Voici une liste d'articles:

```
<ol>
    <li> Ordinateur </li>
    <li> Imprimante </li>
</ol>
</div>
```

Division (2/2)

- L'élément span est similaire à div sauf que celui-ci est un élément en **ligne** ie il ne débute pas et ne finit pas par un retour à la ligne
- **Ex.** Le premier est un élément par bloc, le second est un élément en ligne .
- **Résultat**

  Le premier est un

  élément par bloc

  , le second est un élément en ligne.

Section (1/2)

- Elle sert à diviser le corps du document HTML
- **04** éléments sont utilisés pour définir les différentes sections:
  - <header>
  - <nav>
  - <main>
  - <footer>
- Ces éléments font partie de l'élément **body**

Section (2/2)

- <header>…</header>: section en **haut** de la page. Elle peut contenir le logo, etc
- Ne pas confondre avec l'élément <head> dont le contenu ne sera pas affiché dans la page (sauf le titre)
- <nav>…</nav> : liens de navigation (menu)
- <main>…</main>: section principale de la page
- <footer>…</footer>: section en **bas** de page (liens additionnels, copyright, contacts)

Commentaires

- Un commentaire est introduit par les symboles <!– et -->

- Permet de rendre le document HTML facile à lire et à comprendre

- Les commentaires ne seront pas affichés par les navigateurs

Validation du HTML

- Le site pour vérifier le code html en ligne est https://validator.w3.org/
- 03 techniques:
  - Validation par URI
  - Validation par fichier contenant le code HTML
  - Validation par la saisie directe du code HTML

Université Paris Nanterre

Université Paris Nanterre

Programmation web et Introduction à PHP

L2 MIASHS

M. NAFI

m.nafi@parisnanterre.fr

Année: 2022/2023

Objectifs du cours

- Création de **sites web statiques, interactifs** et **dynamiques** à l'aide de:
  - Langage **html**
  - Langage **CSS**
  - Langage **JavaScript**
  - Langage **PHP**

Prérequis

- Avoir des connaissances sur le langage HTML

Plan du cours

1. Introduction
2. Langage Html
3. Langage CSS
4. Langage Javascript
5. Langage PHP

Plan du cours

1. Introduction
2. Langage HTML
3. **Langage CSS**
4. Langage Javascript
5. Langage PHP

Langage CSS

Introduction

- CSS(Cascading StyleSheet): **Feuilles de style en cascades** est un standard de W3C pour définir la présentation des pages web
- Il s'occupe du style ou de l'apparence des éléments d'un document HTML
- Le langage CSS possède sa propre syntaxe
- La dernière version est CSS3

Syntaxe (1/2)

- Une **règle** CSS décrit comment un élément ou un ensemble d'éléments devraient être affichés par le navigateur
- Elle a la forme suivante:

```
Selecteur {
        propriété 1: valeur 1;      /* déclaration 1
        propriété 2: valeur 2;      /* déclaration 2
        …
        propriété N: valeur N;      /* déclaration N
}
```

Syntaxe (2/2)

- Une règle CSS est constituée de:
  - **Sélecteur** qui identifie l'élément ou les éléments qui vont être affectés par le style
  - Elle indique au navigateur que le code CSS qui suit s'applique à cet élément ou ensemble d'éléments
  - **Déclaration** qui fournit des instructions d'affichage
    - Composée d'une **propriété** et d'une **valeur** séparée par deux points **':'**.
    - Les déclarations sont séparées entre elles par des **';'**
- **Ex.**

```
h1{color: blue;}
body{background-color:orange;}
```

Types de sélecteur

- Il existe plusieurs types de sélecteurs:
  - Sélecteur universel
  - Sélecteur d'élément
  - Sélecteur d'identifiant
  - Sélecteur de classe
  - Sélecteur d'attribut
  - Sélecteur de pseudo-classe
  - Sélecteur de pseudo-élément
  - etc

Sélecteur universel

- Le sélecteur universel noté '*', est utilisé pour appliquer un style à tous les éléments du document HTML
- Ex. *{

```
color:blue;
}
```

=> Tous le **texte** de la page web sera affiché en **bleu**.

Sélecteur d'élément

- Le code CSS s'applique à **tous** les éléments de **même type**
- Il est constitué du **nom** de l'élément (p, div, table, etc)
- Il permet de cibler un élément HTML spécifique

Ex.

```
h1{
    color:green;
}
```

=> Tous les titres de niveaux 1 seront affichés en **vert**

Remarque

- On peut appliquer le même style à plusieurs éléments différents en les énumérant et séparant par une **virgule** (,) => Ce regroupement est très utile
- Ex.

h1 {color:blue;} **et** **p** {color:blue;}

⇔

h1, p {color:blue;}

Sélecteur d'élément enfant

- Les éléments **parent** et **enfant** sont séparés par un espace
- Ex. <div p{/* code css */}

  => 'p' est l'élément enfant de 'div'('div' est son parent)

  - Le code css s'appliquera seulement aux paragraphes qui se trouvent à l'intérieur de l'élément 'div'
- Ex.

   Paragraphe 1

  Paragraphe 2

=> Le style CSS ne s'appliquera qu'au 'Paragraphe1'

Sélecteur d'élément enfant

- first-child: premier enfant
- nth-child(n): **n** ème enfant
- last-child: dernier enfant
- Ex.
  premier enfant
  second enfant
  troisième enfant

- cibler le second => p:nth-child(2){color:red;}

Question

Lorsqu'on souhaite cibler un **seul** élément ou un élément **spécifique**.

Quel est le type de sélecteur le plus approprié?

Réponse

On peut utiliser le sélecteur d'**identifiant**

Sélecteur d'ID

- Ajouter l'attribut '**id**' à un élément du document HTML
- Un identifiant doit être **unique** dans une page web
- Le nom du sélecteur commence toujours par le caractère '#' suivi par la valeur de l'id

  Ex. <h1 id='titre1'> /* HTML

  #titre1{ color:red; } /* code CSS
- Ce style ne s'appliquera qu'à un **seul** élément (l'élément dont la valeur de l'id est 'titre1').

Question

Lorsqu'on souhaite cibler simultanément **plusieurs** éléments de différents types.

Quel est le type de sélecteur le mieux adapté?

Réponse

On peut utiliser le sélecteur de **classe**

Sélecteur de classe

- Pas besoin d'être unique dans une page
- Un élément peut avoir plusieurs classes séparées par un **espace**
- Ex. <p class='classe1 classe2'>
- Ajouter l'attribut **class** aux éléments en question
- Ex. <h1 class='titre'> <h2 class='titre'>
- Le nom du sélecteur commence toujours par le caractère '**.**' suivi par le nom de la classe
- Ex. **.titre** {color:red;}

Sélecteur d'attribut

- Cibler des éléments selon le nom ou la valeur d'un attribut spécifique
- Ex. a[href]{color:blue;}

a[href='www.parisnanerre.fr']{color:blue;}

=> Sélectionner tous les liens vers l'adresse www.parisnanerre.fr

Sélecteur de pseudo-classes

- Pseudo-classe : **état** d'un élément
- On utilise la notation: 'élément:état'
- Ex. l'élément **a** possède **4** états**:**
  - **link** (lien non visité)
  - **visited** (lien visité)
  - **hover** (lien survolé avec la souris)
  - **active** (lien cliqué)

Ex. a:hover{ color:white;}

a:visited{ color:blue;}

Sélecteur de pseudo élément

- On utilise les '::'

❑ Première lettre:

 Paragraphe

p::first-letter { /* code css */ }

❑ Première ligne d'un texte

p::first-line {/* code css */ }

❑ Insertion d'un contenu

Avant : before

Après: after

Ex. p::after {

content: ".";

} /* ajoute un point à la fin de chaque paragraphe

Application du code CSS

03 méthodes existent:

- Lien vers un fichier CSS externe
- En utilisant l'élément <style> </style>
- CSS en ligne (l'attribut style)

❑ Lien vers un fichier externe

Le code CSS est écrit dans un fichier texte **séparé** et sauvegarder sous l'extension **.css**

- Relier le fichier CSS au fichier HTML à l'aide de la balise <link> dans l'élément <head>

Ex. <link rel="stylesheet" type="text/css" href="style.css" >

- Les attributs **rel** et **type** indiquent au navigateur qu'il s'agit d'un fichier CSS

- L'attribut **href** spécifie le chemin vers le fichier CSS

❑ En utilisant l' élément <style> </style>

Le code CSS est incorporé dans l'élément <head> du document HTML

Ex.

```
<head>
        <style>
            /* code CSS
        </style>
</head>
```

❑ CSS en ligne (l'attribut style)

Le code CSS est inséré dans la balise ouvrante d'un élément donné en utilisant l'attribut **style**

Ex. <p **style= " color:'red'; "** > Paragraphe</p>

Remarque

- La première méthode est la plus recommandée car elle permet de séparer le code CSS du code HTML
- Très utile lorsqu'on souhaite appliquer le même code CSS à **plusieurs pages web**
- La seconde est communément utilisée quand on veut appliquer le même code CSS à **une seule page web**
- La dernière méthode est utilisée quand on souhaite appliquer des règles CSS à **un seul élément**

Sensibilité à la casse

- Nom des éléments n'est pas sensible

  Ex. P{} et p{} sont identiques => pas d' erreur

- Le nom des **classes** et **identifiants** sont sensibles

  Ex. Classe1 et classe1 sont différentes => erreur

Ordre de précédence

- Plus le sélecteur est spécifique, plus est la précédence

  Element#id > id > classe > élément

- Ex. PARAGRAPHE

  Style1 :

```
p {color:"green";}
#p1{color:"orange";}
.c1{color:"red";}
```

Résultat: la **seconde** règle sera appliquée ie le paragraphe sera donc affiché en couleur 'orange'

Style 2: p{color:"green";}

#p1{color:"orange";}

p#p1{color:“red";}

.c1{color:"white ";}

Résultat: la 3ème règle sera appliquée ie le paragraphe sera donc affiché en couleur ‘rouge’

- Un élément **hérite** le style de son parent si **aucun** style n'est spécifié

- Ex. Texte en

 Gras

Code CSS:

p { color: "blue "; }

=> La couleur bleue sera appliquée à la fois à 'texte en' et 'gras' si la règle strong {color:'autre couleur';} n'est pas définie

- Si des règles ont la **même** précédence, la **dernière** sera appliquée

- Ex. p { font-size:10px; }
  p { font-size:12px; }

=> Le paragraphe sera affiché avec une police de 12 pixels. ie la dernière règle sera appliquée.

Notion de cascade

- Elle permet de définir l'ordre d'application des propriétés lors d'un conflit
- L'ordre est défini par priorité croissante:
  - Style par **défaut** du navigateur
  - Source **externe** fichier.css
  - Source **interne** avec un style défini dans la page
  - Source en **ligne** avec un style défini dans l'élément HTML (haute priorité)

Commentaire

- Le texte présent entre /* et */ sera ignoré
- Il ne sera pas affiché par le navigateur

Propriétés CSS

- Texte et police
  - Couleur: la propriété 'color' spécifie la couleur du **texte**
- La couleur peut être spécifiée de **trois** manières:
  - Nom de la couleur (ex. color:' blue')
  - RGB (ex. color:'rgb(0,0, 255)'
  - Hexadécimale (ex. color:'#0000ff')

Alignement du texte:

- La propriété ‘text-align’ permet de:
  - Centrer le texte text-align:center;
  - Le positionner à gauche text-align:left;
  - Le placer à droite text-align:right;
  - Le justifier text-align:justify;

Décoration du texte

- La propriété ‘text-decoration’ permet de:
  - Souligner le texte : text-decoration:underline;
  - Rayer ou surligner le texte: text-decoration:overline;
  - Laisser le texte normal text-decoration:none;

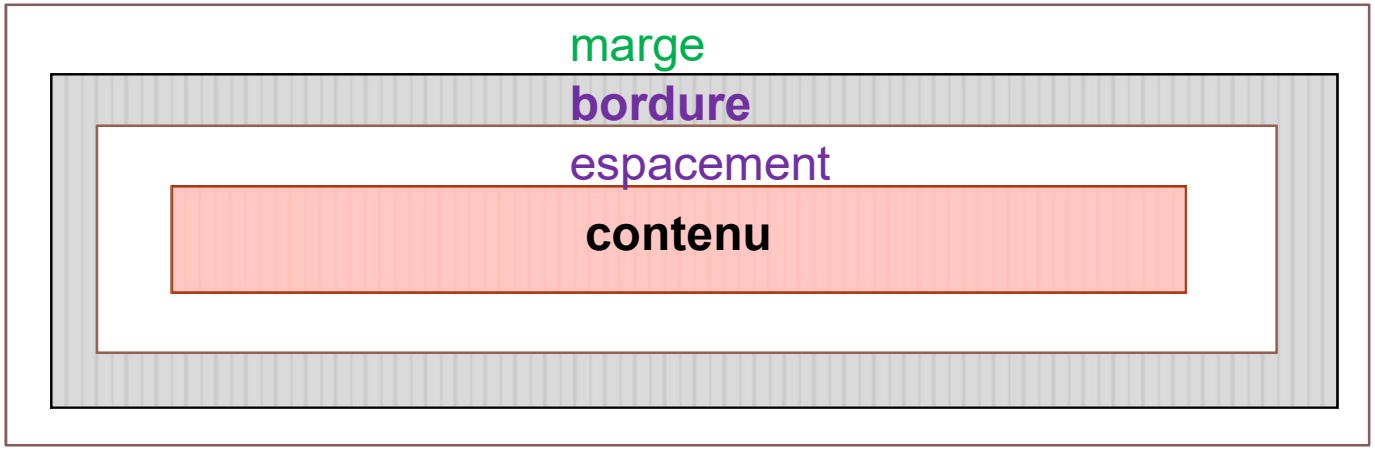
- Espacement

- Lettre: La propriété 'letter-spacing ' permet de:
  - Diminuer l'espace entre les lettres d'un mot (valeur **négative**) letter-spacing:-2px;
  - Augmenter l'espace entre les lettres d'un mot (valeur **positive**) letter-spacing:2px;
- Mot: La propriété 'word-spacing ' permet de:
  - Diminuer l'espace entre les mots (valeur **négative**) word-spacing:-2px;
  - Augmenter l'espace entre les mots (valeur **positive**) word-spacing:2px;

- Ligne: La propriété ‘line-height ’ permet de spécifier l’espace entre les lignes d’un texte. Elle est exprimée en pixel, pourcentage, em, cm et pt (point)
- Ex. line-height:20px;
  line-height:20%;

Modèle en boite

- En CSS, un élément est traité comme une **boite**
- A chaque élément HTML correspond une zone rectangulaire
- Chaque boite ou élément possède des **marges** (margin), **bordures** (border), **espacements** (padding) et un **contenu** (content)



- À l' intérieur de la bordure, on trouve les espacements (padding) et le contenu (content)

- À l'extérieur de la bordure, on trouve les marges (margin)

Propriétés

❑ Contenu

- ➢ Width et height: dimension de la zone de **contenu** (largeur et hauteur). Les valeurs sont exprimées en **pixels** (px) ou en **pourcentage** (%)
  - Elles peuvent aussi prendre la valeur auto
- ➢ Overflow: parfois la zone de contenu est trop petite pour contenir la totalité du texte. Par conséquent, le texte va déborder. Pour y remédier, on utilise la propriété overflow
  - Cette dernière peut prendre les valeurs: visible, hidden, scroll, auto

❑ Padding et margin

- Padding: espace vide entre le contenu et la bordure
- Margin: espace entre la bordure et les éléments adjacents

- Ils sont **transparents** (on ne peut pas changer leur couleurs)
- Il existe 04 syntaxes différentes:

- Ex. les 03 premières notations sont équivalentes
  - Margin-top:10px;
    Margin-right:20px;
    Margin-botom:10px;
    Margin-left:20px;
  - Margin: 10px 20px 10px 20px;
  - Margin: 10px 20px;
  - Margin:10px; Toutes les marges sont identiques

❑ Bordure

- La bordure possède les caractéristiques suivantes: **largeur**, **style** et **couleur**.

➢ Largeur

border-width:thin | medium | thick | nombre positif

➢ Style

border-style:none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset

- Cette règle s'applique au 4 bordures:
  - border-top-style:none
  - border-right-style: solid
  - border-bottom-style:none
  - border-left-style:solid

➢ **Couleur**

border-color:rgb(255,0,0)|red|#FF0000|transparent

- **Définition globale**: border:largeur|style|couleur

  Ex. border:3px solid red;

- **Bordure arrondie**: border-radius:20px;

Marges

- Définir une marge autour de chaque élément pour aérer le contenu de la page
- Margin:valeur: Cette valeur s'applique à **toute** les marges (haute, droite, basse et gauche) dans le sens des aiguillés d'une montre
- Il est possible de définir les marges individuellement
- Ex.
  - margin-top: 3px; /* marge **haute**
  - margin-right: 3px; /* marge **droite**
  - margin-bottom: 2px; /* marge **basse**
  - margin-left: 2px; /* marge **gauche**

Validation

- Service de validation du code CSS en ligne:

  jigsaw.w3.org/css-validator

- La vérification du code peut se faire, soit par:
  - Adresse URI
  - Chargement du fichier CSS
  - Saisie directe du code CSS

Université Paris Nanterre

Université Paris Nanterre

Programmation web et Introduction à PHP

L2 MIASHS

M. NAFI

m.nafi@parisnanterre.fr

Année: 2022/2023

Objectifs du cours

- Création de **sites web statiques, interactifs** et **dynamiques** à l'aide de:
  - Langage **html**
  - Langage **CSS**
  - Langage **JavaScript**
  - Langage **PHP**

Plan du cours

1. Introduction
2. Langage Html
3. Langage CSS
4. Langage Javascript
5. Langage PHP

Plan du cours

1. Introduction
2. Langage HTML
3. Langage CSS
4. **Langage Javascript**
5. Langage PHP

Prérequis

- Avoir des connaissances préalables sur les langages:
  - HTML
  - CSS

Langage JavaScript

Historique

- Créé en **1995** par Brendan Eich à Netscape, et nommé à l'origine LiveScript

- JavaScript est une mise en œuvre de ECMAScript

- ECMAScript (ou ES) est une norme de langage de script qui définit les spécifications de base du langage ie la syntaxe, la sémantique et les fonctionnalités du langage.

Introduction (1/3)

- JavaScript est un langage de programmation qui ajoute de **l'interaction** et du **comportement** au sites web
- C'est un langage de script léger exécuté du coté client ie **interprété** par le navigateur
- Il peut être utilisé aussi bien pour les développements côté client que côté serveur
- JavaScript peut modifier dynamiquement le HTML et le CSS des pages web ie changer le contenu et le style

Introduction (2/3)

- JavaScript est implémenté dans tous les principaux navigateurs web tels que Chrome, Firefox, Safari, Internet Explorer et Edge
- Le code JavaScript est exécuté par les moteurs JavaScript des navigateurs

Introduction (3/3)

- JavaScript est un langage interprété ie pas besoin de compilateur pour transformer le code en une autre forme avant son exécution
- Langage dynamiquement typé ie une variable peut recevoir différents types
- Tout les éléments, attributs des pages web peuvent être accessible par un script en utilisant le DOM (Document Object Model)

Exemples d'utilisation

- Principalement pour le développements de sites web interactifs
- Validation de formulaire (voir si les champs sont correctement remplis)
- Gestion des événements (clic et mouvement de la souris)
- Animations
- Développement de jeux
- etc

Intégration du JS dans une page

- L'élément <script> permet d'incorporer du code JavaScript dans un document HTML ou faire référence à un fichier externe

❑ Script externe

- Le code est écrit dans un fichier séparé sous l'extension .js
- Inclure le fichier avec la balise <script src="script.js></script> => balises sans contenu
- **Avantages**

- appliquer le même script à plusieurs pages web
- séparer le code HTML et JavaScript
- etc

❑ **Script interne** (dans la page web)

- Le code JavaScript est inséré entre les balises <script> et </script>

- Le code JS peut être placé n'importe où dans une page HTML ie dans la partie head ou body

- En général, le code est inséré entre les balises <head> et </head> ou **juste avant** </body>

- Problème: Si le code JavaScript inséré dans <head> manipule des éléments sur la page (le DOM), il ne fonctionnera pas s'il est chargé et analysé avant le code HTML

- Solution: l'attribut async indique au navigateur de continuer à charger le contenu HTML une fois que l'élément de la balise <script> a été atteint
- Ex. <script src="script.js async></script>

- Solution: Le meilleur emplacement sur la page est juste avant la balise fermante </body>, car à cet instant là, le navigateur aura parsé tout le document et son DOM

  => rapide

Attributs async et defer

- Ils contrôlent la façon dont les scripts sont chargés et exécutés
  - defer: le chargement des scripts est différé jusqu'à ce que la page soit entièrement chargée
- les scripts sont exécutés dans l'ordre dans lequel ils apparaissent dans la page
  - async: pas forcement dans l'ordre

Commentaires

- Il en existe deux types:
  - Sur une ligne: //
  - Sur plusieurs lignes ou Multiligne: /* */ comme CSS

Variable et constante

- Conteneur d'information: sert à stocker des valeurs
- Le nom d'une variable ou constante doit commencer par une *lettre* ou _. Pas d'espace, pas de caractère spéciaux
- Une **variable** se déclare avec les mots clés var (**avant** ES2015), let (**après** ES2015), (valeurs peuvent changer)
- Une **constante** se déclare avec le mot clé const (valeurs ne changent pas)

- Ex. var x=3;

  let y='3';

  const pi=3.14;
- JavaScript est sensible à la casse
- Ex.

  Y et y sont deux variables différentes

- var et let?

  var x=5;

  var x='5';

=> Déclaration permise (pas d'erreur)

  let x=5;

  let x='5';

=> **Erreur**

Portée d'une variable (1/2)

- Globale: utilisée dans le script entier
  - Elle est définie à l'extérieur d'une fonction

    var x=2;
  - Ou à l'intérieur d'une fonction sans le mot clé « var ». x=2;
- Locale: utilisée dans la fonction où elle est déclarée
  - Elle est définie à l'intérieur d'une fonction
  - Le nom de la variable est précédée par le mot clé « var ». var x=2;

Portée d'une variable (2/2)

- Ex. function double(num){
  total = num + num;
  return total;
  }
  var total = 10;
  var number = double(20);
  alert(total);

- Résultat: 40

Types de données

- Langage **dynamiquement typé** ie une variable peut recevoir différents types
- Undefined: déclarer une variable sans lui affecter de valeur var x; x n'a pas encore été définie
- Null: Absence de valeur var x=null;
- Numbers: valeur numériques ou nombres var x=5 et y=5.5;
- String: chaine de caractères var x= "55";
- Boolean: valeurs logiques var x=true;
- Object: objet
- etc

Opérateurs

- Mathématiques ou arithmétiques
- Comparaison
- Logiques

Opérateurs arithmétiques

- Addition (+)
- Soustraction (-)
- Multiplication (*)
- Division (/)
- Modulo (%)

Opérateurs de comparaison

- Égal (==)
- Différent (!=)
- Identique: égal et de même type (===)
- Pas identique (!==)
- Supérieur (>)
- Supérieur ou égal (>=)
- Inferieur (<)
- Inferieur ou égal (<=)

Opérateurs logiques

- ET (&&)
- OU (||)
- NOT (!)

Structures de contrôles

- Elles modifient le flux d'exécution des instructions d'un programme
- Les trois structures de contrôle les plus courantes en JS sont:
  - Instructions conditionnelles (if et switch)
  - Boucles (for, while, do … while)
  - Instructions de saut: break et continue

Instruction If else

- Exécute un bloc de code si une condition est vraie et un autre bloc sinon

- Ex.

```
if (Moy >= 10) {
    console.log("Vous êtes admis."); }
else {
    console.log("Vous êtes ajourné."); }
```

- Ex. if (note >= 16) {
  console.log("Très bien"); }
  else if (note >= 14) {
  console.log("Bien"); }
  else if (note >= 12) {
  console.log("Assez bien"); }
  else if (note >= 10) {
  console.log("Passable"); }
  else {console.log("Échec"); }

Plusieurs tests => il est préférable d'utiliser switch

Instruction switch

- Alternative à l'instruction if else pour tester plusieurs options
- Ex. switch(note) {

```
case 16: console.log("Très bien");
         break;
case 14: console.log(" Bien");
         break;
case 12: console.log("Assez bien");
         break;
case 10: console.log("Passable");
         break;
default: console.log("Echec");
         break;  }
```

Boucle

- Permet de répéter des actions ou l'exécution d'un ensemble d'instructions.
- for
- do … while
- while

For

- for(Exp_Initiale; condition; Exp_Increment){

  Instruction1;

  Instruction2;

  …

};

Ex. for(i=0;i<10;i++){

  console.log(i);

};

while

- while(condition){

  Instruction1;

  Instruction2;

  …

  };

Ex. i=0;

```
while(i<10){
    console.log(i);
    i++;
    };
```

Do ... while

- do {

  Instruction1;

  Instruction2;

  …

  }

  while(condition);

Ex. i=0;

do{

console.log(i);

i++;

} while(i<10);

Fonction (1/4)

- Une fonction est un ensemble d'instructions (ou morceau de code) qui réalise une tâche spécifique une fois appelée
- Dans un script, on appelle une fonction par son nom nomFonction()
- Il existe deux types de fonctions:
  - Prédéfinies ou natives
  - Personalisées: définies par l'utilisateur

Fonction prédéfinie (2/4)

- parseInt(): convertit une chaine de caractères en nombre entier
- parseFloat(): convertit une chaîne de caractères en nombre flottant
- alert(): affiche le texte passé en argument et un bouton 'ok' dans une boite de dialogue
- console.log(): affiche le texte passé en argument dans la console du navigateur
- prompt(): affiche une boite de dialogue avec un message et un champ de saisie
- etc

Fonction personnalisée (3/4)

- Une fonction doit être définie avant son utilisation
- La définition d'une fonction se fait comme suit:

```
function nom(paramètres) {
                    //Ensemble d'instructions
                    }; //instruction
```

```
Ex. function somme(a, b){
                        return a+b;
                    }
```

Fonction personnalisée (4/4)

- Une fonction peut aussi être définie comme suit:

```
var nom = function (paramètres){ bloc
d'instructions} // expression de fonction
```

```
Ex. var somme=function (a, b){
                    return a+b;
                  }
```

Tableaux (1/5)

- Un tableau est un objet contenant un ensemble d'éléments ou valeurs.
- Les éléments d'un tableau peuvent être de différents types ie nombres, chaines de caractères, objets, etc
- Déclaration et initialisation

```
Ex. let tab=[ ]; // tableaux vide
    let tab=[3,4,5]; // tableau à 3 éléments
    let tab=Array.of(3,4,5);
```

Tableaux (2/5)

- L'accès aux éléments d'un tableau se fait en utilisant la notation crochet
- Ex. let Tab=[3,4,5];

  Tab[0]=3, Tab[1]=4, Tab[2]=5, Tab[3] // undefined
- La taille d'un tableau s'obtient avec la propriété 'length'
- Ex. let Tab=[3,4,5];

  let taille=Tab.length // taille=3
- Un tableau est un **objet** de type **Array**
  - Typeof(Tab) => object
  - Array.isArray(Tab)=> true

Tableaux (3/5)

❑ Quelques méthodes

- push(): permet **d'ajouter** des éléments à la **fin** d'un tableau

Ex. let Tab=[3,4,5];

let x=Tab.push(6,7); / / Tab=[3,4,5,6,7], x=5 (Taille)

- pop(): permet de **supprimer** le **dernier** élément d'un tableau

Ex. let Tab=[3,4,5];

let x=Tab.pop() / / Tab=[3,4], x=5

Tableaux (4/5)

- unshift(): permet **d'ajouter** des éléments au **début** d'un tableau

Ex. let Tab=[3,4,5];

Tab.unshift(1,2); // Tab=[1,2, 3, 4,5]

- shift(): permet de **supprimer** le **premier** élément d'un tableau

Ex. let Tab=[3,4,5];

let x=Tab.shift() // x=3, Tab=[4,5]

Tableaux (5/5)

- slice(): crée une copie d'une partie d'un tableau. Elle prend **deux** arguments facultatifs: indice de début (inclus) et indice de fin (non inclus)

Ex. let Tab=["a", "b", "c", "d"];

let Tab1=Tab.slice(1, 3); // Tab1=["b", "c"]

- splice(): ajoute ou supprime des éléments d'un tableau. Elle prend au moins deux arguments: indice début de la MAJ, nombre d'éléments à supprimer, éléments à insérer (facultatifs)

Ex. let Tab=["a", "b", "c", "d"];

let Tab2=Tab.splice(1, 2); // Tab=["a", "d"]

let Tab3=Tab.splice(1, 0, "b", "c"); // Tab =["a", "b", "c", "d"]

Objets du navigateur

- JavaScript peut contrôler les éléments d'une page web et manipuler les parties de la fenêtre du navigateur
- En JavaScript, un navigateur est un objet window
- Cet objet possède des propriétés et des méthodes
- Ex. propriétés: event, history, location, status
- Ex. méthodes: alert(), close(), confirm(), focus()

Document Object Model (DOM)

- Le DOM (Document Object Model) est une interface de programmation API de documents HTML, XML
- Il permet **d'accéder** et de **manipuler** le contenu HTML (éléments, attributs et textes) et le style CSS d'une page web
- Transforme le document en format compréhensible par les langages tel que JavaScript

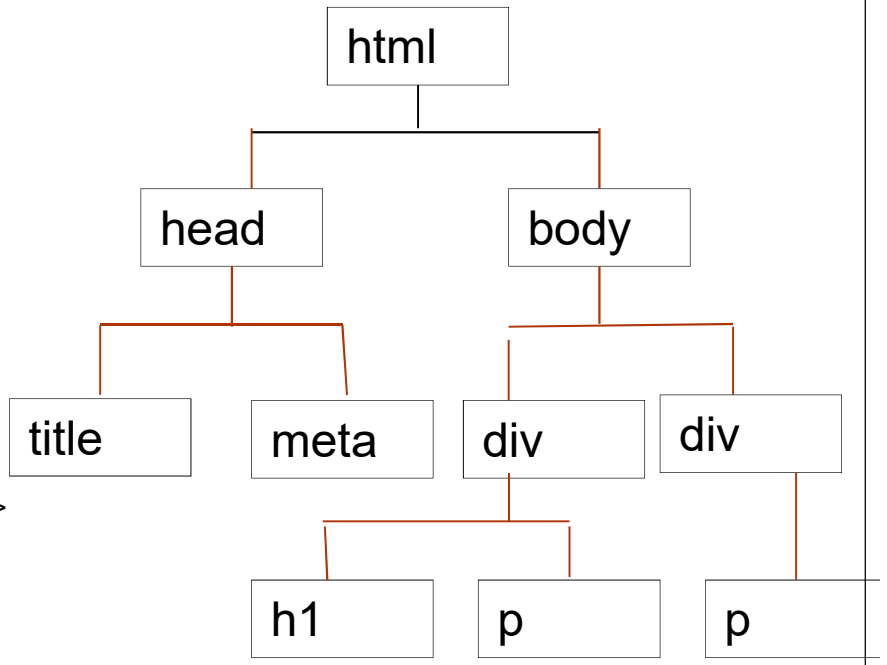
=>Un document HTML est représenté sous forme d'une arborescence d'objets.

Document Object Model (DOM)

- Le DOM est un ensemble de nœuds
- Chaque nœud représente un élément du document tel qu'une balise HTML, un attribut ou un texte
- Avec le DOM, on peut **ajouter**, **modifier** ou **supprimer** des éléments, **modifier** les styles CSS, ou réagir à des événements

Exemple

```
<!DOCTYPE html>
<html>
        <head>
                <title>titre</title>
                <meta charset="utf-8">
        </head>
        <body>
                <div>
                        <h1>titre</h1>
                        <p>Paragraphe</p>
                </div>
                <div>
                        <p>paragraphe.</p>
                </div>
        </body>
</html>
```



Accès au DOM

- L'objet 'document' identifie la page web
- Cet objet possède des propriétés et des méthodes
- Ex. var x = document.getElementById("id1").innerHTML;

Méthodes d'accès aux nœuds

- Plusieurs méthodes permettent d'accéder au DOM à partir d'un script JavaScript. On peut citer:

❑ **getElementById()** permet d'accéder à un élément HTML **spécifique** en utilisant l'attribut id de l'élément. Elle retourne un **seul** élément.

Ex. <img src="photo.jpg" alt="" id="photo1">
var photo = document.getElementById("photo1");

- **getElementsByClassName()** permet d'accéder à **tous** les éléments qui ont une certaine classe CSS. Elle retourne un **ensemble** d'éléments

Ex. Y =document.getElementsByClassName("CLASSE");

elle sélectionne **tous** les éléments de la classe " CLASSE "

- **getElementsByTagName()** permet d'accéder à tous les éléments qui ont un certain nom de balise HTML.

Ex. X=document.getElementsByTagName("p")

Elle retourne **tous** les paragraphes de la page

X[0]=premier paragraphe

- **querySelector()** permet d'accéder à un élément HTML spécifique en utilisant des sélecteurs CSS. Elle retourne le **premier** élément correspondant au sélecteur spécifié.
  - Ex. var y =document.querySelector(".classe1"); Elle retourne le premier élément qui contient la classe CSS 'classe1'
- **querySelectorAll()** permet d'accéder à un ensemble d'élément HTML en utilisant des sélecteurs CSS. Elle retourne une **liste** d'éléments correspondant au sélecteur(s) spécifié(s).
  - Ex. var elts = document.querySelectorAll(".classe2"); elle sélectionne tous les éléments de la classe CSS 'classe2'
- **etc**

Méthodes de manipulation des noeuds

- **setAttribute()** permet de changer la valeur d'un attribut. Elle prend deux arguments: l'attribut et la nouvelle valeur

  Ex. var Image = document.getElementById("image1");
  Image.setAttribute("src", "image.jpg");

- **innerHTML** permet d'accéder et de changer le texte d'un élément

  Ex.var d = document.getElementsByClassName("classe1");
  d[0].innerHTML = " paragraphe ";

- **createElement()** permet l'ajout d'éléments à la page web. Cette méthode prend un seul élément en argument qui est le nom de la balise

  Ex. var d = document.createElement("div");

- **createTextNode()** permet de créer un nouveau nœud de texte.

  Ex. var t = document.createTextNode("Texte.");

  t: noeud de texte

  Texte: son contenu

- **appendChild()** ajoute un nœud à la **fin** de la liste des enfants d'un nœud parent spécifié
- Elle prend un argument qui est le nœud à insérer au DOM

Ex. Ajouter un paragraphe qui contient le texte « paragraphe1 » à l'élément « div ».

```
var d = document.getElementById("div1");
var p = document.createElement("p");
var text = document.createTextNode(" paragraphe1");
p.appendChild(text );
d.appendChild(p);
```

- **replaceChild()** remplace un nœud enfant par un autre. Elle prend deux arguments

```
parent.replaceChild(newChild, oldChild);
```

- **removeChild()** retire un nœud enfant. Elle prend un seul argument

```
noeud.removeChild(enfant);
```

Ex.
```
var parent = document.getElementById("parentid");
var enfant = document.getElementById("enfantid");
parent.removeChild( enfant );
```

- **insertBefore()** insère un nœud avant un autre nœud dit de référence.

  parent.insertBefore(newNode, RefNode)

  Ex. Div.insertBefore(H, P); insère le nœud « H » avant le nœud « P » au niveau du nœud parent « Div »

- **style**

  Le DOM permet d'ajouter, modifier, supprimer un style d'un élément

  Ex. var x= document.getElementById("id1");

  x.style.color = "red";

Evènements en JS

- Un évènement est une action qui peut être détectée avec JavaScript
- Il est identifié par ce qu'on appelle un gestionnaire d'événement ou event handler tels que:
- **onload** lance un script lors du chargement de la page
- **onmouseover** lance un script lorsqu'un utilisateur survole un élément avec la souris

- **onclick** lance un script lorsqu'un utilisateur clique sur un élément
- **onsubmit** lance un script lorsqu'un utilisateur clique sur un button 'envoyer'
- **onkeypress**: lance un script lorsqu'une touche du clavier est enfoncée
- etc

Il existe **trois** méthodes pour appliquer les event handlers aux éléments d'une page

- **Attribut**: spécifier la fonction à exécuter dans un attribut de la page HTML

  Ex. <**h1** onclick="F();"> // la fonction F s'execute lorsqu'un utilisateur clique sur le titre de niveau 1

- **Méthode**: la fonction est attachée à un élément

  window.onclick = F; /* la fonction F s'exécute lorsqu'un utilisateur clique dans la page web */

- **addEventListener(e,f)** elle prend deux arguments:
  - e: évènement
  - f: fonction à exécuter

Ex. window.addEventListener("click", Fonction);

window.addEventListener("click", function(e) { // code de la fonction });

Validation

- La validation est importante pour protéger les données de formulaire contre les hackers

- La fonction **htmlspecialchars()** convertit les **caractères spéciaux** en **entités** HTML pour sécuriser les valeurs

=> Evite l'exécution de codes malveillants tapés dans les champs de saisie du formulaire

Validation avec PHP

- Passer toutes les variables via la fonction htmlspecialchars()
- Si un hacker insère le code JavaScript suivant: <script>location.href('http://www.hacked.com')</script>
- Ce script ne sera pas exécuté car il sera sauvegardé comme suit: <script>location.href('http://www.hacked.com')</script>

- La fonction **trim()** supprime les caractères **inutiles** (espaces, tabulations, saut de ligne, retour chariot, etc)

Syntaxe:

trim(chaine à **traiter**, chaine à **supprimer**): string

Ex. Trim(" HTML, PHP ") // supprime les espaces

Trim("HTML, et PHP", "et")// supprime la chaine de caractères 'et'

=> Sortie : " HTML, PHP "

- La fonction **stripslashes()** enlève les antislashs (\)
- On peut créer une fonction qui fait tout:

```
function validation($data) {
        $data = trim($data);
        $data = stripslashes($data);
        $data = htmlspecialchars($data);
      return $data;
      }
```

Exemple

- La variable **$_SERVER["PHP_SELF"]** est une variable superglobale qui renvoie le **nom** du script en cours d'exécution

- **Code vulnérable** aux attaques

```
<form method="POST" action="<?php echo
$_SERVER["PHP_SELF"];?>">
```

- **Code sûr**

```
<form method="POST" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

=> Cela empêche l'injection de code malveilant

Programmation orientée objet

- Une **classe** est un **moule**, **prototype** ou un **modèle** pour les objets

- Une classe possède des **propriétés** (*attributs*) et des **méthodes** (*comportements*)

- Une classe définit **l'état** et les **comportements** d'un objet

- Un **objet** est une **instance** d'une classe

- Les objets héritent de toutes les propriétés et de tous les comportements de la classe

- Chaque objet aura des valeurs différentes pour les propriétés

- Dans une classe, les **variables** sont appelées propriétés et les **fonctions** sont appelées méthodes

Exemples

- **Classe**: Voiture

  Objets: Nissan, Toyota, Renault, …etc

- **Classe**: Animal

  Objets: Chat, Chien, Singe, …etc

- **Classe**: Fruit

  Objets : Pomme, Banane, Orange, ..etc

- En PHP, une classe est définie par le mot clé **class** suivi par le **nom** de la classe, une accolade ouvrante, les propriétés et méthodes puis une accolade fermante)
- **Ex**.

```
<?php
        class Personne {
        // Proprietés
        public $nom;
        public $adresse;
        // Méthodes
        function afficher_nom() {
          return $this->nom;
         } }
    ?>
```

- $this fait référence à l'objet courant, et n'est disponible qu'à l'intérieur des méthodes

- Les objets d'une classe sont créés à l'aide du mot clé **new**

```
<?php
    $etudiant = new Personne();

    echo $etudiant->afficher_nom();

    ?>
```

- La valeur d'une propriété peut changer soit:
  - À l'intérieur d'une classe en utilisant *$this* dans une méthode

```
<?php
    class Personne {
    public $nom;
   function set_name($name) {
      $this->nom = $name;  }}
  $etudiant = new Personne();
  $etudiant->set_name("Jean");
  echo $etudiant->nom;
?>
```

  - Directement à l'extérieur d'une classe

```
<?php
    class Personne {
    public $nom;}
   $etudiant = new Personne();
  $etudiant->nom = "Jean";
  echo $etudiant->nom;
 ?>
```

Gestion de base de données MySQL

MySQL est un SGBDR

- Les étapes de gestion d'une base de données MySQL avec PHP sont:

1. **Connexion** au serveur MySQL
2. **Envoi** des **requêtes** SQL au serveur
3. **Récupération** et traitement des résultats des requêtes
4. **Fermeture** de connexion

- La gestion de BDD peut se faire avec:

  - MySQLi: spécifique à MySQL
    - Accès procédural
    - Accès objet

  - PDO (PHP Data Objects): supporte **différentes** base de données (MySQL, SQLite, PostgreSQL, oracle, etc)

❑ MySQLi: Accès procédural

- Connexion au serveur MySQL: mysqli_connect

  $idcon= mysqli_connect($host, $user, $pass, [$db]);

- Envoi des requêtes SQL au serveur:

  $res=mysqli_query($requête, [idcon])

- Récupération et traitement des résultats des requêtes: mysqli_fetch_array($res), mysqli_fetch_object

  mysqli_fetch_assoc($res) et mysqli_fetch_row($res)

- Fermeture de connexion: mysqli_close($idcon)

❑ MySQLi: Accès objet

- Connexion au serveur MySQL: $mysqli=new mysqli()

  $idcon =new mysqli($host, $user, $pass, [$db]);
- Envoi de requêtes SQL au serveur:

  $res=$idcon->query($requete)
- Récupération et traitement des résultats des requêtes: $res-> fetch_array(), $res-> fetch_assoc(), $res-> fetch_row(), $res–>fetch_object()
- Fermeture de connexion: $idcon->close()

❑ Accès PDO à MySQL

- Connexion au serveur MySQL:

```
$idcon= new PDO("mysql:host=$host;dbname=$base ,$user, $pass");
```

- Envoi de requêtes SQL au serveur:

```
$res= $idcon->query($requête), $res= $idcon->exec($requête)
```

- Récupération et traitement des résultats des requêtes: $res->fetch(), $res->fetchAll(), $res->fetchObject()
- Fermeture de connexion: $idcon=NULL

Session vs Cookie

- SESSION : les variables de session sont stockées sur le serveur **le temps** de la présence d'un visiteur sur le site

- COOKIE : les cookies sont enregistrés sur l'ordinateur du visiteur pendant **plusieurs mois**

Session

- Une session est un mécanisme qui permet de conserver des données ou variables entre plusieurs requêtes d'un utilisateur sur un site web
- En PHP, une session peut être créée à l'aide de la fonction **session_start()**
- Cette fonction doit être appelée au **début** de chaque script PHP qui utilise des variables de session
- Elle associe un **identifiant** de session unique à l'utilisateur
- Une fois que la session démarrée, on peut définir des variables de session à l'aide de la superglobale **$_SESSION**
- **Ex.** $_SESSION['nom'] = 'Piere';

Fonctions de session

- **session_start():** démarre une session. Lorsqu'un visiteur arrive sur le site, un numéro ou ID de session est généré pour lui. C'est un nombre hexadécimal.
- Cette fonction est appelée au **début** des pages où on a besoin de conserver des variables de session
- **session_destroy():** ferme la session du visiteur
- Elle est automatiquement appelée lorsque un visiteur **se déconnecte** ou **ne charge plus** la page pendant plusieurs ***minutes*** (timeout)
- **session_id():** indique l'identifiant de la session
- **etc**

Cookie

- Un cookie est un petit fichier que l'on enregistre sur l'ordinateur du **visiteur**
- Ce fichier contient du texte et permet de conserver des informations sur le visiteur (son pseudo, ses préférences, etc)
- Chaque cookie a une *date d'expiration*. Après cette date, il sera automatiquement supprimé par le navigateur
- Les cookies sont donc des informations temporaires stockées sur l'ordinateur des visiteurs

- Pour écrire un cookie, on utilise la fonction PHP **setcookie()** qui possède, en général, **trois** paramètres:
    1. le **nom**
    2. la **valeur**
    3. la **date d'expiration** (timestamp)

- Ces informations sont sauvegardées dans la superglobale $_COOKIE

Modèle MVC

- Le modèle MVC (Modèle-Vue-Contrôleur) est un modèle de **conception** logicielle

- Il est souvent utilisé pour le développement d'applications **Web**

- Il permet la séparation de la **présentation**, la **logique** et les **données** de l'application

- Il permet de mieux **organiser**, **faciliter** la maintenance et **améliorer** la réutilisabilité du code

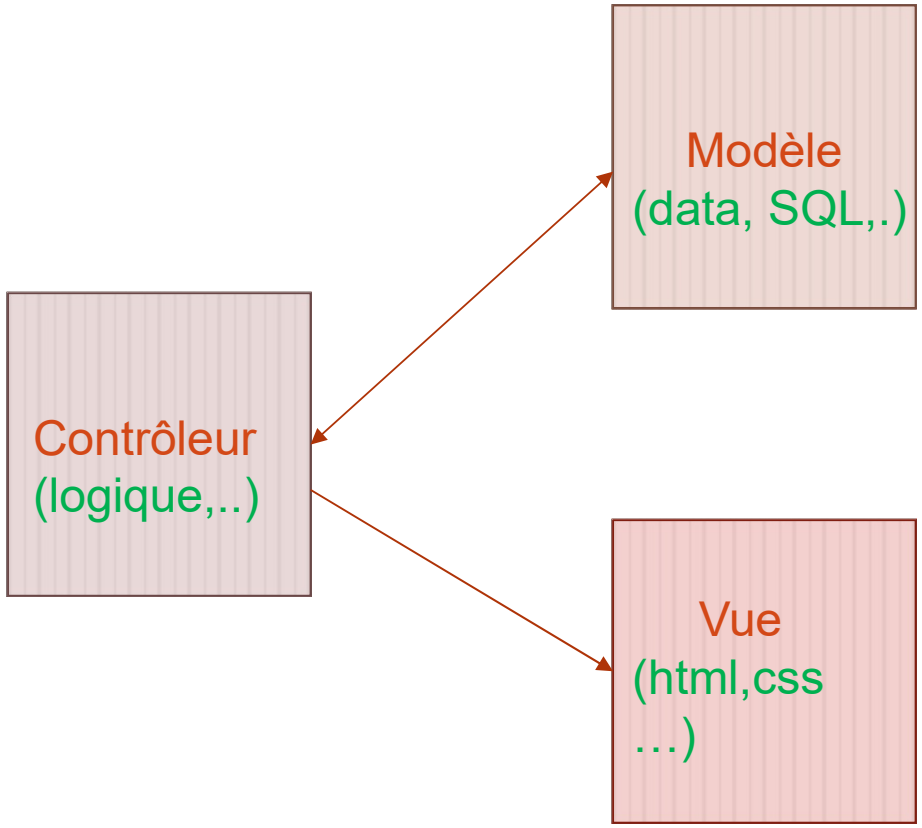
Modèle MVC

- Ce modèle se compose de **trois** parties : Modèle, Vue et Contrôleur
- **Modèle** (Model) gère les **données** et les opérations qui y sont associées. Il peut s'agir d'une *base de données*, d'un *système de fichiers*, d'un *service web* ou de toute autre source de données. On y trouve des requêtes SQL

- **Vue** (View) est responsable de **l'affichage** des données au format HTML, XML ou autre. Elle n'a pas de connaissance directe de la source de données. On y trouve du HTML et un peu de PHP

- **Contrôleur** (Controller) : gère les **interactions** de l'utilisateur avec l'application. Il sert d'intermédiaire entre le modèle et la vue. Il gère la **logique** du code et prend des **décisions**. Il *reçoit* les entrées de l'utilisateur, *effectue* des opérations sur le modèle et *renvoie* une réponse à la vue.

Fig. Architecture MVC



Processus d'échange d'information

1.Le contrôleur demande les données au modèle

2. Le modèle traduit la demande en une requête SQL,
- Récupère les données et les renvoie au contrôleur

3. Le contrôleur transmet les données récupérées à la vue

4. La vue affiche ces données

Exemple site e-commerce

- **Modèle**: une **base de données** contenant les produits, les clients et les commandes, ainsi qu'un ensemble de fonctions pour gérer les interactions avec ces données
- **Vue**: les pages **HTML** générées dynamiquement à partir des données du modèle. Par exemple, une page qui affiche la liste des produits avec leur nom, leur prix et un bouton "Commander"
- **Contrôleur**: Par exemple, lorsqu'un utilisateur clique sur le bouton "Commander ", le contrôleur reçoit la demande, ajoute le produit au panier du client dans le modèle, puis renvoie une réponse à la vue pour afficher la mise à jour du panier

Merci pour votre attention

Fonctions

- PHP possède plus de 1000 fonctions prédéfinies
- Une fonction est un bloc d'instructions
- Elle ne s'exécute pas automatiquement
- On peut créer nos propres fonctions

```
function NomFonction (parameteres) {
        // code à executer;
}
```

- *NomFonction* doit commencer par une **lettre** ou « _ »
- Une fonction doit être définie avant d'être appelée

Exemple

- Ecrire une fonction qui détermine le maximum entre deux nombres
- On peut la nommer Max

```
<?php
    Function Max (int $x, int $y) {
        If ($x>$y){
            echo "le maximum est: ".$x;
        else
            echo "le maximum est: ".$y;
                }
        }
    Max(12, 3); // appel de la fonction Elle affiche le maximum est:12
    ?>
```

```php
<?php
function maxi(int $x, int $y=3) {
  if ($x>$y){
  echo "le maximum est:".$x;
  }
   else {
   echo "le maximum est: ".$y;
  }
}
maxi(4,5); // affiche le maximum est:5
maxi(2); // affiche le maximum est:3 utilise la valeur par défaut
?>
```

- La fonction qui retourne la somme de deux entiers

```
<?php
  function somme(int $a, int $b) {
  return $a + $b;
 }
 echo somme(2, “3 ");
 ?>
```

- Sortie 5 // comme strict n’est pas activé, “3" est transformé en int(3), et la fonction retourne 5

- Langage PHP est faiblement typé

```
<?php
    declare(strict_types=1); // types doivent être respectés
    function somme(int $a, int $b) {
    return $a + $b;
    }
    echo somme(2,“3 ");
?>
```

Sortie: PHP Fatal error

Quelques fonctions mathématique

- **pi():** renvoie la valeur de **pi**
- **min():** renvoie la valeur **minimale** d'un tableau
- **max():** renvoie la valeur **maximale** d'un tableau
- **abs():** calcule la valeur **absolue** d'un nombre
- **sqrt():** calcule la **racine carrée** d'un nombre
- **round():** arrondit un nombre à la valeur **la plus proche**
- **Rand():** renvoie un nombre **aléatoire**
- **etc**

Tableaux (array)

- Un tableau stocke **plusieurs** valeurs dans une seule variable
- On peut accéder aux valeurs en se référant aux numéros d'indice ou clés
- En PHP, la notation **[ ]** ou la fonction **array()** sont utilisées pour créer un tableau
- Ex. $tab=[ ] // tableau vide

  $tab=array() // tableau vide

- Il existe **trois** types de tableaux:
  - **Indexés**: utilisent des indices *numériques*
  - **Associatifs**: utilisent des *clés* nommées (clé=>valeur)
  - **Multidimensionnels**: contenant un ou *plusieurs* tableaux

Tableau indexé

- Il existe deux façons de créer un tableau indexé:
  - L'index est attribué *automatiquement*

```
$couleurs=array('rouge', 'vert', 'bleu');
```

  - L'index est affecté *manuellement*

```
$couleurs[0]='rouge';
$couleurs[1]='vert';
$couleurs[2]='bleu';
```

  - On peut utiliser la boucle **for** pour parcourir tous les éléments d'un tableau indexé.
  - Ex.

```
For($x=0; $x<count($couleurs);$x++)
    echo $couleurs[$x];
```

Tableau associatif

- Utilise des *clés nommées*
- Il existe deux façons de créer ce type de tableau :
  - $age = array("Peter"=>“25", "Ben"=>"30", "Jean"=>"40");
  - $age['Peter'] = “25";
    $age['Ben'] = "30";
    $age['Jean'] = "40";
- On peut utiliser la boucle foreach pour parcourir les éléments d’un tableau associatif.

- Ex. <?php

```
$age = array("Peter"=>“25", "Ben"=>"30“);

foreach($age as $x => $value) {
        echo “Clé=" . $x . ",Valeur=" . $value;
        echo "<br>";
}
?>
```

Tableau multidimensionnel

- Dans un tableau à **deux** dimensions, on a besoin de **deux** indices pour sélectionner un élément.
- Ex. $personne=array(

  array('Peter',' 25'),

  array('Ben',' 30'));

Nom	Age
Peter	25
Ben	30

- Pour accéder à l'age de 'Peter', on utilise deux indices $personne[0][1]='25' ou deux boucles **for**

Quelques fonctions

- Count(): compte le nombre d'éléments d'un tableau (taille)
- array_reverse(): inverse les éléments d'un tableau
- array_unique(): supprime les doublons
- sort(): trie un tableau indexé dans l'ordre croissant des ses valeurs
- etc

- Ex. <?php

```
$couleur = array("rouge", "bleu", "vert");
echo count($couleur);
?>
```

- Résultat: 3

Tri des éléments d'un tableau

- Les éléments d'un tableau peuvent être triés par ordre alphabétique ou numérique, croissant ou décroissant.
- Les fonctions de tri sont:
  - **sort():** tableau indexé, ordre croissant de ses valeurs
  - **rsort()**: tableau indexé, ordre décroissant de ses valeurs
  - **asort()**: tableau associatif, ordre croissant de ses valeurs
  - **ksort()**:tableau associatif, ordre croissant de ses clés
  - **arsort()**: tableau associatif, ordre décroissant de ses valeurs
  - **krsort()**: tableau associatif, ordre décroissant de ses clés

Exemple 1

```
<?php
    $colors = array("red", "blue", "green");
    rsort($colors); // ordre alphabétique décroissant
    $taille = count($colors);
    for($x = 0; $x < $taille; $x++) {
    echo $colors[$x]; // affichage des éléments du tableau
    echo "<br>";
  }
?>
```

Résultat: red

green

blue

Exemple 2

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

- Résultat:

  Key=Peter, Value= 35
  Key=Ben, Value= 37
  Key=Joe, Value= 43

Exemple 3

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

  Key=Ben, Value=37
  Key=Joe, Value=43
  Key=Peter, Value=35

Exemple 4

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

  Key=Joe, Value= 43
  Key=Ben, Value= 37
  Key=Peter, Value= 35

Exemple 5

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

  Key=Peter, Value=35
  Key=Joe, Value=43
  Key=Ben, Value=37

Formulaires

- GET et POST sont deux méthodes utilisées pour ***collecter*** les données de formulaire.
- **Ex**

```
<form method='POST' action='file.php'>
        Nom:<input type='text' name='name'>
        <input type='submit'>
</form>
```

- File.php

```
<?php
        echo 'le nom est: $_POST[name]';
?>
```

- $_POST[name] crée un tableau associatif contenant des **clés** et des **valeurs** (clé1=>valeur1, clé2=>valeur2 …)

  - Clé: nom d'un champ de formulaire
  - Valeur: valeur saisie par l'utilisateur

Get vs Post

- **GET**
  - Données **visibles** (noms et valeurs sont affichées dans l'URL)
  - Quantité de données à envoyer est **limitée** (2000 caractères)
  - Utilisée pour l'envoi des données **non sensibles**
- **POST**
  - Données sont **invisibles** pour les autres (noms et valeurs sont encapsulés dans le **corps** de la requête HTTP)
  - **Aucune** limite sur la taille des données à envoyer
  - Utilisée pour l'envoi des données **sensibles** comme les *mots de passes*.

NB: Il est donc préférable d'utiliser POST pour la soumission des données.

Université Paris Nanterre

Université Paris Nanterre

Programmation web et Introduction à PHP

L2 MIASHS

M. NAFI

m.nafi@parisnanterre.fr

Année: 2022/2023

Objectifs du cours

- Création de **sites web statiques, interactifs** et **dynamiques** à l'aide de:
  - Langage **html**
  - Langage **CSS**
  - Langage **JavaScript**
  - Langage **PHP**

Plan du cours

1. Introduction
2. Langage Html
3. Langage CSS
4. Langage Javascript
5. Langage PHP

Plan du cours

1. Introduction
2. Langage HTML
3. Langage CSS
4. Langage Javascript
5. **Langage PHP**

Introduction

- PHP (PHP Hypertext Preprocessor)
- Il a été créé en 1994 par Rasmus Lerdorf
- Un langage de script utilisé pour le développement de pages web **dynamiques**
- Les **scripts** ou codes PHP sont exécutés sur le serveur. Le résultat est retourné au client (navigateur) sous forme de document ou code **HTML**

- Syntaxe du PHP est **proche** de celle du langage C, Perl ou Java

- PHP est **multiplateformes** (Windows, Linux, Mac OS, etc)

- **Compatible** avec la plupart des serveurs web (Apache, IIS, etc)

- Supporte un large éventail de **bases de données** (MySQL, MariaDB, PostgreSQL, etc)

- **Open source** et gratuit

- Un fichier PHP porte l'extension **.php** et contient du texte, html, css, javascript et le code php.

Exemple

```
<!DOCTYPE HTML>
<HTML>
    <head>
        <title> titre </title>
    </head>
    <body>
        <h1> <?php echo "Salut";?>  </h1>
    </body>
</HTML>
```

Serveur


```
<!DOCTYPE HTML>
<HTML>
    <head>
        <title> titre
        </title>
    </head>
    <body>
     <h1> Salut </h1>
    </body
</HTML>
```

Client

Installation

Besoins

- Installer un serveur web (Apache)
- Installer l'interpréteur PHP
- Installer un SGBD (MySQL)

Ou installer **XAMPP**(Apache MySQL PHP Perl)

- **W**AMP: Windows
- **L**AMP: Linux
- **M**AMP: Mac OS

Syntaxe

- Un fichier PHP porte l'extension **.php** et contient du texte, html, css, javascript et le code ou script php.
- Un script PHP est délimité par les deux balises: <script language='php'> et </script> ou **<?php** et **?>**
- Une instruction se termine par **';'** (séparateur)

Ex.

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

Variables

- Les variables sont des **conteneurs** pour le stockage des informations

- En PHP, une variable commence par le signe $, suivi du nom de cette variable

- Le nom de la variable doit commencer par une **lettre** ou le caractère de **soulignement** '_'.

- Le nom des variables est **sensible** à la casse ($x ≠ $X)
- Le nom des variables ne doit pas commencer par un nombre
- Le nom d'une variable ne peut contenir que des caractères alphanumériques et des caractères de soulignement (A-z, 0-9 et _)

- Nb: Donner des noms **significatifs** aux variables (**$age** au lieu de **$a)**

- Ex.<?php
  $x = “PHP!";
  $y = 1;
  ?>

- Après exécution de ce script PHP, la variable:
- **$x** contiendra la chaine de caractère “PHP!“
- **$y** la valeur entière “1”

Portée d'une variable

- **Locale**: déclarée et accessible seulement **à l'intérieur** d'une fonction
- **Globale**: déclarée et accessible **en dehors** d'une fonction

- Le mot-clé « global » est utilisé pour accéder à une variable globale à l'intérieur d'une fonction

- **Statique**: on utilise le mot clé « static » pour **garder** la valeur d'une variable locale (ie elle ne sera pas supprimée à la fin de l'exécution de la fonction)

Variables superglobales

- Toujours accessibles de n'importe où et quel que soit leurs portées
  - $_POST
  - $_GET
  - $_REQUEST
  - $GLOBALS
  - $_SERVER
  - $_FILES
  - $_ENV
  - $_COOKIE
  - $_SESSION

- $_POST: utilisée pour collecter les données de formulaire soumises avec la méthode ‘Post’
- $_GET: utilisée pour collecter les données de formulaire soumises avec la méthode ‘Get’
- $_REQUEST: utilisée pour collecter les données après la soumission d'un formulaire HTML
- $GLOBALS: utilisé pour accéder aux variables globales depuis n'importe quel endroit du script PHP
- $_SERVER: contient des informations sur les en-têtes, les chemins et les emplacements des scripts

Types de données

- PHP attribue **automatiquement** un type de données à la variable, en fonction de sa valeur

- Les variables peuvent stocker des données de **différents** types tels que: Integer, float, string, array, boolean, object, Null, resource

Integer

- Nombre entier **négatif** ou **positif**

  Ex. $x = 20;

- Quelques fonctions sur les entiers
  - is_int()
  - is_integer()
  - is_long()

Float

- Nombre décimal ou à virgule flottante
- Ex. $x = 20.5;
- Quelques **fonctions** sur les nombres réels
  - is_float()
  - is_double()

String

- Une **séquence** ou une **chaine** de caractères
- Texte entre **guillemets**

  Ex. $x= "Apprendre PHP"
- Voici quelques fonctions couramment utilisées pour manipuler les chaînes de caractères:
  - **strlen()** retourne la **taille** d'une chaine de caractères
  - **strrev()** retourne **l'inverse** d'une chaine de caractères
  - **strpos()** retourne **la position** d'une sous chaine
  - **str_replace()** **remplace** certains caractères par d'autres

Boolean

- **Deux** valeurs ou états: vrai ou faux

Ex. $x=true;

$y=false;

Array

- Variable contenant plusieurs valeurs

Ex.

$couleur = **array** **(“rouge",“orange",“vert");**

Object

- Concept de la programmation **orientée objet**
- Une **instance** d'une classe
- Un objet hérite toutes les **propriétés** et tous les **comportements** de la classe

Null

- Aucune valeur ou une seule valeur:null

Fonction vardump()

- Fonction var_dump() est utilisée pour connaitre le type et la valeur d'une variable.

- Ex.

```
<?php
   $x=10;
   echo var_dump($x);
?>
```

Résultat: int(10)

Opérateurs

- Affectation (=, +=, -=,*=,/=,%=)
- Arithmétiques (+,-,*,/,%)
- Concaténation (.)
- Logiques (&& (et), || (ou), ! (non))
- Comparaison (==,!=,<,<=, >,>=)

Structures de contrôle (1/2)

- Instructions conditionnelles
  - If () else
  - If () elseif () else
  - Switch

Structures de contrôle (2/2)

Boucles

- **While**: Tant que la condition est satisfaite, un bloc de code est exécuté.
- **Do …while**: Répéter un bloc de code tant que la condition est vérifiée (le bloc est exécuté au **moins** une **seule** fois)
- **For**: exécuter un bloc de code un nombre de fois **spécifique**
- **Foreach**: utilisé pour les array

While

- Syntaxe

```
while (condition est vraie) {
        code à executer;
}
```

Ex.

```
<?php
    $x = 1;
    while($x <= 10) {
    echo "$x <br>";
    $x++;
    }
?>
```

Do ... while

- Syntaxe

```
Do{
        code à executer;
} while (condition est vraie)
```

- Ex.

```
<?php
    $x = 1;
    Do{
        echo "$x <br>";
        $x++;
    } while($x <= 10)
?>
```

For

- Syntaxe

```
for (init;  test;  increment)
  {
       Code à executer;
   }
```

- Ex.

```
<?php
    $x = 1;
    for ($x=1;$x <= 10;$x++) {
    echo "$x <br>“;
   }
?>
```

Foreach

- Syntaxe

```
foreach ($array as $valeur){
        code à executer;
}
```

- Ex.

```
<?php
    $x = array("1","2","3");
    foreach ($x as $valeur) {
    echo "$valeur <br>;
   }
?>
```