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);
```