



Développement Web: Javascript & PHP

CHAPITRE 1

INTRODUCTION À JAVASCRIPT

PLAN

- ▶ Introduction
- ▶ Insertion de JavaScript Dans HTML
- ▶ Syntaxe et commentaires
- ▶ Affichage dans un document HTML
- ▶ Les boîtes de dialogue

Introduction

- ▶ JavaScript est un langage de programmation créé en 1995 par Netscape.
- ▶ JavaScript est aujourd'hui l'un des langages de programmation les plus populaires et il fait partie des langages web dits « standards » avec le HTML et le CSS.
 - ▶ Les principaux navigateurs web (Google Chrome, Safari, Firefox, etc.) savent tous « lire » ou « interpréter » ces langages et les interprètent généralement de la même façon
 - ▶ Un même code va généralement produire le même résultat dans chaque navigateur.
- ▶ Son évolution est gérée par le groupe ECMA International qui se charge de publier les standards de ce langage.

Introduction

- ▶ JavaScript est un langage dynamique, basé évènement :
 - ▶ Ce langage nous permet de manipuler des contenus HTML ou des styles CSS et de les modifier en fonction de divers évènements: par exemple un clic d'un utilisateur à un certain endroit de la page.
- ▶ JavaScript est un langage (principalement) côté client ;
 - ▶ Généralement exécuté dans le navigateur des utilisateurs qui demandent la page.
 - ▶ Peut être utilisé côté serveur (en utilisant Node.js par exemple).
- ▶ JavaScript est un langage interprété ;
 - ▶ Il peut être exécuté directement par le logiciel interpréteur (pas besoin d'une compilation préalable de la totalité du code)
- ▶ JavaScript est un langage orienté objet.

“

INSERTION DE JAVAScript DANS HTML

”

Insertion de JavaScript Dans HTML

- ▶ Il existe 3 méthodes pour ajouter du code JavaScript à votre document HTML
 1. Directement dans une balise Html : généralement associé à un évènement
 2. JavaScript interne à une page Html
 3. JavaScript dans un fichier externe

Directement dans une balise HTML

► Exemple:

```
<body>  
  <p onclick="alert('Paragraphe')"> Un paragraphe </p>  
</body>
```

Code JavaScript

JavaScript interne à un fichier HTML

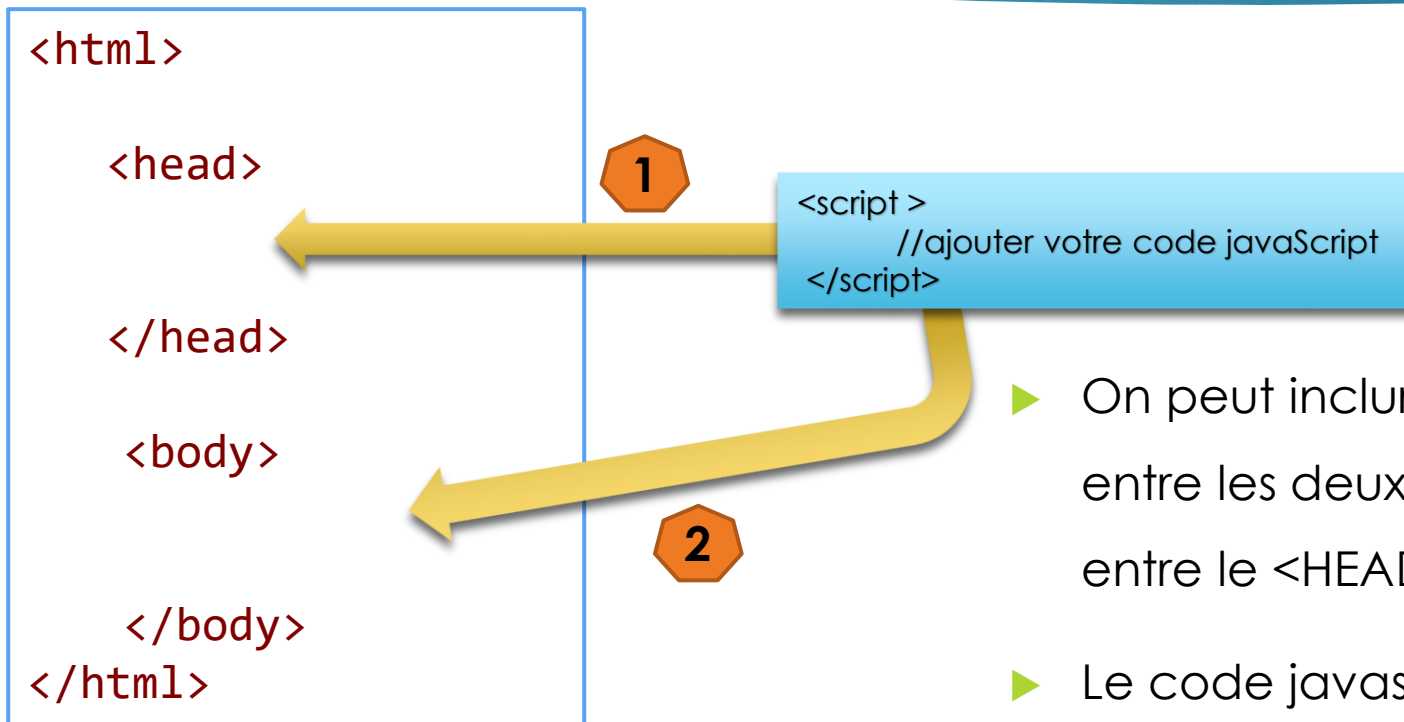
- ▶ Tout code JavaScript à intégrer dans une page HTML doit être placé entre les balises « script » :
- ▶ L'élément HTML **<script>** est utilisé pour intégrer ou faire référence à un script (en JavaScript ou un autre langage de script)

```
<script >  
    //ajouter votre code javascript  
</script>
```

- ▶ Il est possible de spécifier les attributs **type** et/ou **language** de la balise **<script>**

```
<script language="javascript" type="text/javascript">  
    //ajouter votre code javascript  
</script>
```


JavaScript interne à un fichier HTML



Fichier HTML

- ▶ On peut inclure un ou plusieurs codes JavaScript entre les deux balises `<BODY>` et `</BODY>` ou même entre le `<HEAD>` et `</HEAD>` Selon les besoins
- ▶ Le code javascript placé dans le HEAD sera exécuté avant la première ligne du BODY

JavaScript dans un fichier externe

- Il est possible d'utiliser des fichiers externes (avec l'extension **.js**) pour les scripts JavaScript et les intégrer dans un fichier HTML. La balise script devient :

```
<script language="javascript" src="chemin/nom_fichier.js">  
    //ajouter votre code javascript  
</script>
```

exemple.html

```
<html>  
<head> ...</head>  
<body>  
<!--Appel du script externe-->  
    <script language="javascript" src="myscript.js">  
    </script>  
</body>  
</html>
```

myscript.js

```
let msg = "Hello world";  
document.write("mon message "+ msg);
```

“

SYNTAXE ET COMMENTAIRES

”

Syntaxe de base

- ▶ JavaScript emprunte la plupart des éléments de sa syntaxe à Java, C et C++ mais sa syntaxe est également influencée par Perl et Python.
- ▶ En JavaScript, les instructions sont séparées par des points-virgules.
- ▶ Il n'est pas nécessaire d'inclure un point-virgule si l'on écrit une instruction sur une nouvelle ligne
- ▶ JavaScript est **sensible à la casse**
- ▶ Un commentaire sur une ligne s'écrit après un double slash : `// Ceci est un commentaire`
- ▶ Un commentaire sur plusieurs lignes s'écrit entre deux balises `/*` et `*/`

```
/*  
  Ceci est un commentaire  
  sur plusieurs lignes  
*/
```

“

AFFICHAGE DANS UN FICHIER
HTML

”

Affichage dans un fichier HTML

- ▶ La fonction **document.write(texte)** Ecrit le paramètre texte dans le document, à la position de l'appel du script.
- ▶ Le paramètre texte peut être une chaîne de caractères classiques, du code(HTML , CSS, ...)
- ▶ Utiliser le symbole '+' pour concaténer plusieurs chaîne dans document.write

```
<body>  
  <script language="javascript" >  
    bonjour = "Bonjour !";  
    question = "Comment allez vous ";  
    phrase = bonjour + "<BR>" + question;  
    document.write(phrase, "aujourd'hui ?");  
  </script>  
</body>
```

Bonjour !
Comment allez vous aujourd'hui ?

Affichage dans un fichier HTML

Code JavaScript

```
<head>
  <script>
    var message = "Hello World";
  </script>
</head>
<body>
  <p>Voici un texte Html</p>
  <script>
    document.write("Voici un texte Js");
    document.write("<h2> Voici un titre Js </h2>");
  </script>
  <p> Afficher une variable en js</p>
  <script>
    document.write("message = "+ message);
  </script>
</body>
```

Voici un texte Html

Voici un texte Js

Voici un titre Js

Afficher une variable en js

message = Hello World

“

LES BOITES DE DIALOGUE

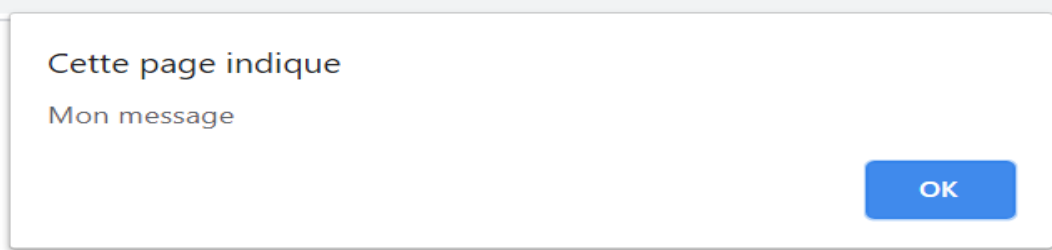

”

Les boîtes de dialogue

- ▶ Une boîte de dialogue est une fenêtre qui s'affiche au premier plan suite à un événement, et qui permet
 - ▶ Soit d'avertir l'utilisateur
 - ▶ Soit le confronter à un choix
 - ▶ Soit lui demander de compléter un champ pour récupérer une information
- ▶ JavaScript propose 3 boîtes de dialogue par défaut (de l'objet window) qui permettent d'interagir avec l'utilisateur
 - ▶ **alert()** : permet d'afficher un message.
 - ▶ **confirm()** : permet de récupérer une valeur booléenne
 - ▶ **prompt()** : permet de récupérer une valeur textuelle.

alert()

- Alert(): Elle permet d'afficher un texte et/ou le contenu d'une variable et un bouton « OK ».

| Code | résultat |
|--|---|
| <code>alert("Mon message");</code> |  |
| <code>let x = 20; alert("Mon age est "+ x);</code> |  |

confirm()

- `confirm()`: Elle permet d'afficher un message et deux boutons : un bouton « OK » et un bouton « Annuler ». Cette fonction retourne une valeur booléenne qui vaut « **true** » si c'est le bouton « OK » qui est cliqué et retourne « **false** » si c'est le bouton « Annuler » qui est cliqué.

```
let choix = confirm( "Cliquer sur un bouton" );  
if(choix==true)  
    alert("vous avez cliqué sur oui");  
else  
    alert("vous avez cliqué sur annuler");
```

Cette page indique
Cliquer sur un bouton

OK

Annuler

Si vous cliquez sur OK

vous avez cliqué sur oui

OK

Si vous cliquez sur Annuler

vous avez cliqué sur annuler

OK

prompt()

- ▶ `prompt()`: Elle permet d'afficher un message, un champ à remplir (input de type texte) et un bouton « OK ». Elle retourne la valeur qui a été entrée dans le champ par l'utilisateur.
- ▶ La méthode `prompt()` requiert deux arguments :
 - ▶ Le texte à afficher dans la boîte de message
 - ▶ la chaîne de caractères par défaut dans le champ de saisie

```
// La valeur à saisir dans l'input sera sauvegardé dans la variable "age"  
var age = prompt("saisir votre age", 20);
```

Valeur par
défaut

Cette page indique

saisir votre age

20

OK Annuler



Développement Web:

Javascript & PHP

CHAPITRE 2

VARIABLES ET CONSTANTES

PLAN

- ▶ LA DECLARATION ET L'AFFECTATION
- ▶ LES TYPES DE VARIABLE
- ▶ Les opérateurs

Les variables

- ▶ Les variables sont des conteneurs dans lesquels on peut stocker des valeurs.
- ▶ En Javascript, les noms de variables doivent répondre à certains critères :
 - ▶ Il doit commencer par une lettre ou un "_"
 - ▶ Il peut comporter des lettres, des chiffres et les caractères _ et &
- ▶ Javascript est faiblement typé : le langage déterminera automatiquement le type d'une variable, on n'a donc pas besoin de spécifier le type d'une variable à sa création.
- ▶ Il est possible de modifier le type d'une variable au cours de l'exécution du programme.

Déclaration

- ▶ La déclaration consiste à donner un nom à la variable, alors que l'affectation consiste à donner une valeur à la variable.
- ▶ Le type n'est pas précisé lors de la déclaration.

| Symbole | Rôle | Exemple |
|--------------|---|--------------------|
| var | On déclare une variable dont la portée est globale | var age = 20 |
| let | On déclare une variable dont la portée est celle du bloc courant. La plus utilisée. | let nom = 'Ahmed'; |
| const | | const PI=3.14; |

- ▶ La syntaxe de déclaration des variables avec let correspond à la nouvelle syntaxe. La syntaxe avec var est l'ancienne syntaxe qui est vouée à disparaître.

let vs. var

► Remontée (Hoisting)

- Avec var, on n'est pas obligé de déclarer la variable avant de la manipuler dans le code.
- Cela est possible car le JavaScript va traiter les déclarations de variables effectuées avec var avant le reste du code JavaScript.
- Ce comportement, appelé remontée ou hoisting, a été jugé comme inadapté dans les versions récentes de JavaScript et a donc été corrigé dans la déclaration de variables avec let

```
//Ceci fonctionne  
prenom = "Toto";  
var prenom;
```

```
//Ceci ne fonctionne pas et renvoie une erreur  
nom = "Toto";  
let nom;
```

let vs. var

► La re-déclaration de variables

- Avec l'ancienne syntaxe var, on avait le droit de déclarer plusieurs fois une même variable en utilisant à chaque fois var (ce qui avait pour effet de modifier la valeur stockée).
- La nouvelle syntaxe avec let n'autorise pas cela. Pour modifier la valeur stockée dans une variable avec la nouvelle syntaxe, il suffit d'utiliser le nom de la variable et de lui affecter une autre valeur.

```
//Ceci fonctionne  
var prenom = "Toto";  
var prenom="Tata";  
  
//Ceci ne fonctionne pas et renvoie une erreur  
let nom = "Toto";  
let nom="Tata";
```

“

LES TYPES DE VARIABLES

”

Les types de variables

- ▶ En JavaScript, il existe 8 types de valeurs différents. Chaque valeur qu'on va pouvoir créer et manipuler en JavaScript va obligatoirement appartenir à l'un de ces types:
 - ▶ String
 - ▶ Number
 - ▶ BigInt
 - ▶ Boolean
 - ▶ Null
 - ▶ Undefined
 - ▶ Symbol
 - ▶ Object
- ▶ Voir: https://www.w3schools.com/js/js_datatypes.asp

Les types de variables

- ▶ JavaScript a des types dynamiques. Cela signifie que la même variable peut être utilisée pour contenir différents types de données :

```
let x;           // Now x is undefined  
x = 5;           // Now x is a Number  
x = "John";      // Now x is a String
```

Chaîne de caractères ou « String »

- ▶ Une chaîne de caractères est une séquence de caractères, ou ce qu'on appelle communément un texte.
- ▶ Toute valeur stockée dans une variable en utilisant des guillemets ou des apostrophes sera considérée comme une chaîne de caractères, et ceci même dans le cas où nos caractères sont à priori des chiffres comme "29" par exemple.

```
// Using double quotes:  
let carName1 = "Volvo XC60";  
// Using single quotes:  
let carName2 = 'Volvo XC60';
```

```
// Single quote inside double quotes:  
let answer1 = "It's alright";  
// Double quotes inside single quotes:  
let answer3 = 'He is called "Johnny"';
```

Les nombres « Number »

- ▶ En JavaScript, et contrairement à la majorité des langages, il n'existe qu'un type prédéfini qui va regrouper tous les nombres qu'ils soient positifs, négatifs, entiers ou décimaux (à virgule) et qui est le type **Number**.

```
let x = 10; // x un entier positif  
let y = -2; // y un entier négatif  
let z = 3.14; // z un nombre decimal positif
```

▶ BigInt:

- ▶ JavaScript BigInt est un nouveau type de données (ajouté en 2020) qui peut être utilisé pour stocker des valeurs entières trop grandes pour être représentées par un nombre JavaScript normal (64-bit floating-point)

Booléen ou « Boolean »

- Le type booléen ne contient que deux valeurs : les valeurs true (vrai) et false (faux).

```
let x = 5;  
let y = 5;  
let z = 6;  
(x == y)      // Returns true  
(x == z)      // Returns false
```


Les types de valeurs Null et Undefined

- ▶ La valeur **undefined** correspond à une variable « non définie », c'est-à-dire une variable à laquelle on n'a pas affecté de valeur.
- ▶ En JavaScript, une variable sans valeur a la valeur undefined. **Le type est également undefined.**

```
let car;    // Value is undefined, type is undefined
```

- ▶ La définition de Undefined ressemble à celle de **null** et pourtant ces deux valeurs sont différentes. Si on déclare une variable et qu'on lui passe **null**, alors son **type sera Object**.

```
let car = null;    // The value is null, the typeof is object
```

- ▶ Attention à ne pas confondre avec les valeurs vides

```
let car = "";    // The value is "", the typeof is string
```

L'opérateur typeof

- L'opérateur **typeof** renvoie le type d'une variable ou d'une expression :

```
typeof ""           // Returns "string"
typeof "John"       // Returns "string"
typeof "John Doe"   // Returns "string"

typeof 314           // Returns "number"
typeof 3.14          // Returns "number"
typeof (3)           // Returns "number"
typeof (3 + 4)       // Returns "number"
```

Conversion de types

- ▶ Les types de données sont convertis automatiquement durant l'exécution du script.
- ▶ Lorsque des expressions impliquent des chaînes de caractères et des valeurs numériques ainsi que l'opérateur `+`, JavaScript convertit les nombres en chaînes de caractères :

```
x = "La réponse est " + 42; // "La réponse est 42"  
y = 42 + " est la réponse"; // "42 est la réponse"
```

- ▶ Avec des instructions impliquant d'autres opérateurs, JavaScript ne convertit pas nécessairement les valeurs numériques en chaînes de caractères.

```
"37" - 7; // 30  
"37" + 7; // "377"
```

Conversion de type

- ▶ Plusieurs fonctions permettent la conversion de type en Javascript:
- ▶ Convertir une chaîne en un nombre:
 - ▶ **Number()** : Renvoie un nombre, converti à partir de son argument
 - ▶ **parseFloat()** : Analyse une chaîne et renvoie un nombre à virgule flottante
 - ▶ **parseInt()** : Analyse une chaîne et renvoie un entier

```
Number("3.14") ; // returns 3.14
Number(Math.PI); // returns 3.141592653589793
Number(" "); // returns 0
Number(""); // returns 0

Number("99 88");// returns NaN
Number("John"); // returns NaN
```

Conversion de type

```
parseInt("-10"); // returns -10  
parseInt("10.33"); // returns 10  
parseInt("10 20 30"); // returns 10  
parseInt("10 years"); // returns 10  
parseInt("years 10"); // returns NaN
```

```
parseFloat("10"); // returns 10  
parseFloat("10.33"); // returns 10.33  
parseFloat("10 6"); // returns 10  
parseFloat("10 years"); // returns 10  
parseFloat("years 10"); // returns NaN
```

Conversion de type

- ▶ Une chaîne non numérique est convertie en **NaN** (Not a Number)
- ▶ La fonction **isNaN()** renvoie true si une valeur est NaN.

```
isNaN(0/0); //true  
isNaN(''); //false  
isNaN('A'); // true  
isNaN(true); //false  
isNaN(false); //false
```

Conversion de type

- On utilise les même fonctions pour convertir les booléens

```
Number(false)    // returns 0
Number(true)     // returns 1

String(false)    // returns "false"
String(true)     // returns "true"
```

Conversion de type

- ▶ Plusieurs fonctions permettent la conversion de type en Javascript:
- ▶ Convertir un nombre en une chaîne:
 - ▶ `String()` : Renvoie une chaîne, converti à partir de son argument

```
let x = 123;  
String(x);           //123 returns a string from a number variable x  
String(123) ;        //123 returns a string from a number literal 123  
String(100 + 23);    //123 returns a string from a number from an expression
```


“

LES OPÉRATEURS

”

Les opérateurs arithmétiques

| Signe | Nom | Signification | Exemple | Résultat |
|-------|---------------|--------------------------|----------|----------|
| + | Plus | addition | $x + 3$ | 14 |
| - | Moins | soustraction | $x - 3$ | 8 |
| * | multiplié par | multiplication | $x * 2$ | 22 |
| / | divisé par | division | $x / 2$ | 5.5 |
| % | modulo | reste de la division par | $x \% 5$ | 1 |
| = | Affectation | à la valeur | $x = 5$ | 5 |

Les opérateurs de comparaison

| Signe | Nom | <i>Exemple</i> | Résultat |
|-------|-------------------|----------------|----------|
| == | Egal | x == 11 | true |
| < | Inférieur | x < 11 | false |
| <= | Inférieur ou égal | x <= 11 | true |
| > | Supérieur | x > 11 | false |
| >= | Supérieur ou égal | x >= 11 | true |
| != | Différent | x != 11 | false |

Les opérateurs logiques

| Signe | Nom | Exemple | Signification |
|-------|-----|------------------------------|---------------------------------|
| && | et | (condition1) && (condition2) | condition1 <u>et</u> condition2 |
| | ou | (condition1) (condition2) | condition1 <u>ou</u> condition2 |

Les opérateurs d'incrémentation

| Signe | Description | Exemple | Signification | Résultat |
|-------|----------------|---------|---------------|----------|
| x++ | incrémentation | y = x++ | y = x + 1 | 6 |
| x-- | Décrémentation | y = x-- | y = x - 1 | 4 |



Questions?