# Semaine 1 : Introduction à JavaScript

*Cours_ UVCI*
*PRW2103-1*

*DIABATE Nabègna*,
Université Virtuelle de Côte d'Ivoire

juin 2018
1.0

# Table des matières

**I - Objectifs** **3**

**II - Introduction** **4**

**III - Section 1 - Définition et fonctionnement de JavaScript** **5**

1. 1. Qu'est ce que JavaScript ? ..... 5

2. 2. Comment et où écrire du code JavaScript ? ..... 6

3. Exercice ..... 8

**IV - Section 2 - Syntaxe du Langage** **9**

1. 1. La notion de cadrage ..... 9

2. 2. Les commentaires et les variables en JavaScript ..... 9

3. 3. Les opérateurs ..... 11

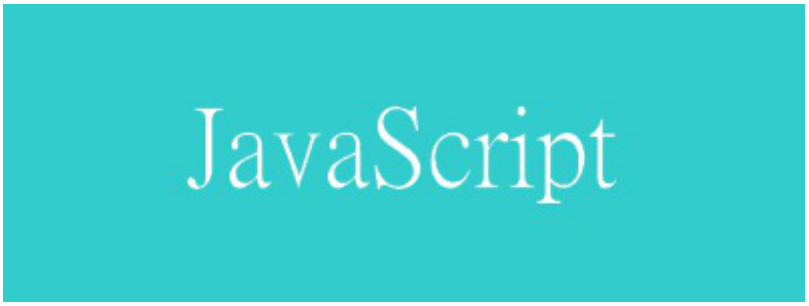
4. Exercice : Quiz d'entraînement ... ..... 13

# *Objectifs*

À la fin de cette leçon, vous serez capable de :

- *Expliquer* l'utilisation et le fonctionnement de JavaScript;
- *Décrire* la structure générale d'un programme et la syntaxe du langage JavaScript.

# Introduction




Le JavaScript est un langage de programmation crée par Brendan Eich, orienté vers les objets-prototypes. Il est traditionnellement interprété par un navigateur Web au sein d'une page web et permet, entre autres choses, de modifier l'organisation de la page.

Le JavaScript est un langage « client », c'est-à-dire exécuté chez l'utilisateur lorsque la page Web est chargée. Il a pour but de dynamiser les sites Internet.

# Section 1 - Définition et fonctionnement de JavaScript

## 1. 1. Qu'est ce que JavaScript ?

Le JavaScript (JS) est un langage dit "client" qui est exécuté par le navigateur web c'est-à-dire qu'il n'est pas nécessaire de lire ses pages depuis un serveur web comme pour le PHP. Il est désactivable sur la plupart des navigateurs web en modifiant le menu "options".

Ce langage permet entre autres de développer l'interactivité d'un site (pop-up, pop-under, contrôle de formulaire, chargement d'images...) à l'aide de scripts et de donner donc un aspect de "vivant" à une page Web.

Un script, c'est tout simplement une suite d'instructions qui vont être interprétées par un programme.

Ainsi, pour lire du code JavaScript, il va nous falloir un interpréteur. Heureusement, tous les navigateurs (Google Chrome, Safari, etc.) possèdent leur propre interpréteur JavaScript.

JavaScript permet de faire beaucoup de choses. Avec le HTML dynamique, nous pouvons manier les balises d'une page web. il aussi est possible même d'échanger des informations avec le serveur sans rechargement de la page grâce à une collaboration avec des langages comme *AJAX*.

Cependant, il faut prendre des précautions dans l'utilisation de JavaScript. Car le code JavaScript peut être vu par les internautes au même titre que votrs code HTML. Il faut donc veuillez à ne pas stocker des informations ou données sensibles comme les mot de passe dans votre code JavaScript.

Ce qui fait toute la puissance du JavaScript, c'est qu'on va pouvoir l'utiliser pour manipuler dynamiquement le code HTML d'une page.

Faîtes cependant bien attention à ne pas confondre *JavaScript* et *Java*! Ces deux langages, bien que syntaxiquement assez proches à la base, reposent sur des concepts fondamentaux complétement différents et servent à effectuer des tâches totalement différentes.

⚠ *Attention : Les limites de JavaScript*

Le JavaScript facilite la navigation au sein d'un site Web. Au fil du temps, des améliorations et des fonctionnalités sont apportées avec Ajax, par exemple. Cependant, compte-tenu des différences d'interprétation d'un navigateur à un autre, il est conseillé de tester chaque script sur différents navigateurs voire sous différents systèmes d'exploitation pour assurer une portabilité du code.

En Outre, on peut dégager trois "limites" essentielles dans le JavaScript en tant que langage de présentation des données :

- Le JavaScript ne possèdent pas de fonctionnalité graphique.
- Le JavaScript ne peut pas lire et encore moins écrire des fichiers sur le disque dur du visiteur (hormis les cookies).
- Le JavaScript ne s'interface pas avec une base données de type MySQL ou SQL.

## 2. 2. Comment et où écrire du code JavaScript ?

Il ne faut pas d'éditeur particulier pour écrire des scripts en JavaScript. On peut insérer du code JavaScript dans une page HTML grâce à un simple éditeur de texte comme nous le faisons déjà bien avec Notepad++.

Il existe cependant plusieurs autres outils ou plates-formes proposant l'écriture de code JavaScript. Ce qu'il faut retenir, c'est que l'usage de ces outils vous faciliteront l'écriture de code en vous permettant :

- De Vérifier la syntaxe du code grâce à la coloration automatique.
- De disposer de l'auto-complétion.
- D'avoir un bon outil pour le débogage ; et souvent,
- De connaître la valeur d'une variable lors de l'exécution du script.

Comme pour le CSS, le code JavaScript peut se placer à trois endroits différents :

1. Dans un fichier externe à la page web avec l'extension obligatoire .js.
2. Entre les balises <head> et </head> ou dans la balise <body>de la page web.
3. Directement dans une balise de la page à l'aide des évènements.

Dans le premier cas, on doit inclure le fichier *.js* (par référence) directement dans notre page web et toujours entre les balises <head> et </head> comme ceci :

```
...
<head>

<script src="./dossier_javascript/fichier.js"></script>

</head>
...
```

C'est très souvent la méthode recommandée dans le cas de gros projets car elle permet une meilleure maintenabilité du code grâce à la séparation des langages, et car on va pouvoir réutiliser un même code JavaScript dans plusieurs fichiers HTML.

Dans ce cas, nous allons devoir lier nos fichiers HTML et JavaScript en utilisant à nouveau un élément *script* et son attribut *src*.

En valeur de l'attribut *src*, nous allons indiquer le chemin relatif du fichier *.js* par rapport au fichier *.html*. Si nos deux fichiers sont dans le même dossier, par exemple, il suffira d'indiquer le nom du fichier JavaScript.

Dans le deuxième cas, le code doit se tenir entre les balises <script> et </script> comme ceci :

```
...
<head>

<script>
// Votre script
</script>

<!-- OU -->

<script language="javascript">
// Votre script
</script>

<!-- OU -->

<script type="text/javascript">
// Votre script
</script>

```

```
</head>
...
```

En effet, le premier endroit où l'on peut placer du code JavaScript est dans l'élément head d'un fichier HTML.

Dans ce cas, il faudra placer le JavaScript à l'intérieur d'un élément *script*.

On placera généralement l'élément *script* à la fin de notre élément *head*.

On peut également écrire notre code JavaScript au sein de l'élément *body* d'un fichier HTML.

Dans ce cas là également, nous utiliserons un élément script et préférerons le placer à la fin de notre page.

Lorsqu'on écrit du JavaScript dans un fichier HTML, que ce soit dans l'élément *head*, le *body* ou les deux, on peut utiliser autant d'éléments *script* que l'on veut.

Cependant, pour des raisons évidentes de performance et de clarté, on essaie souvent de placer tout notre code JavaScript dans un même élément script.

Dans le dernier cas, nous verrons cela plus tard dans le cours mais voici un exemple très simple que vous avez déjà dû apercevoir un jour :

```
<input type="button" onClick="..." />
```

Il s'agit d'un événement. JavaScript gère très bien beaucoup d'événements : du simple clic sur un bouton (exemple ci-dessus) jusqu'à la fermeture de la fenêtre en cours.

Comme premier script, voyons comment écrire un texte à l'écran une fois la page chargée :

```
<!DOCTYPE html>
<html lang="FR">
  <head>
    <meta charset="utf-8" />
    <title>Page modele en Javascript</title>
    <script>
      document.write("Bonjour tout le monde, je suis du Javascript");
    </script>
  </head>
  <body>

  </body>
</html>
```

Testez ce code dans votre navigateur et vous verrez le résultat !

Il faut savoir que tous les navigateurs internet n'autorisent pas ou ne supportent pas forcément le JavaScript. Certains navigateurs proposent même à l'utilisateur d'autoriser ou non le JavaScript. Si vous voulez quand même passer des informations aux visiteurs ayant un navigateur ne supportant pas JavaScript, il faudra utiliser la balise *<noscript>*.

Exemple :

```
<script language="javascript">
document.write ("bonjour");
</script>
<noscript> votre navigateur ne peut pas lire le code JavaScript</noscript>
```

Dans cet exemple, le navigateur affichera Bonjour s'il est compatible ou le texte figurant entre les balises *<noscript>* et *</noscript>* s'il ne l'est pas.

# 3. Exercice

**Exercice**

*1) Quelles sont les abréviations possible qu'on peut donner au langage JavaScript ?*

- ☐ JS
- ☐ JSCript
- ☐ Java
- ☐ JSC

**Exercice**

*2) Avec lesquelles des bases de données suivantes peut-ont utiliser (ou interfacer) Javascript ?*

- ☐ MySQL
- ☐ SQL
- ☐ PostgreSQL
- ☐ Aucune

**Exercice**

*3) Quelle est l'extension d'un fichier JavaScript ?*

**Exercice**

*4) Où peut-on placer du code JavaScript dans une page HTML ?*

- ☐ Dans la section <head>
- ☐ Dans la section <Body>
- ☐ Partout dans le code après ouverture de la balise <body>

**Exercice**

*5) Quelle balise nous permet de tester si un navigateur est compatible ou supporte JavaScript ?*

# Section 2 - Syntaxe du Langage


## 1. 1. La notion de cadrage

C'est le seul point assez complexe du langage qui fait qu'on pourrait le traiter de capricieux qu'est JavaScript. En effet, le JavaScript se base sur des objets de la page où il est présent. De ce fait, nous utiliserons souvent l'écriture :

```
window.document.form.champ.xxxxx
```

Les éléments nommées ci-dessus sont des objets organisés de façon hierarchique :

- L'objet *window* pointe sur la fenêtre du navigateur (c'est donc le niveau le plus bas dans la hiérarchie possible en *js*).
- L'objet *document* est à un niveau un peu supérieur. En effet, cela pointe sur la page mais au niveau de son contenu brut (sa source en l'occurrence).
- L'objet *form* (parmi tant d'autres) est un objet qui vise les formulaires présents dans la page. C'est donc un niveau encore plus précis que document.
- Enfin, l'objet *champ* est une occurrence de form. C'est un objet appartenant à *form*.

Et les "*xxxxx*" sont les méthodes et les caractères de chaque occurrences de l'objet *form*.

Il faut donc retenir qu'*une fenêtre possède plusieurs niveaux en JavaScript*. Du plus bas (window) au plus haut qui est un caractère d'une occurrence d'un niveau inférieur tel que *form*.

Vous verrez par la suite que ce qui paraît compliqué en théorie s'avère simple en pratique !

```
// On avait ça :
document.write("Bonjour tout le monde !");
// Cela équivaut à cela :
window.document.write("Bonjour tout le monde !");
```

Ici, on peut constater que write n'est en fait qu'une méthode (fonction) de l'ensemble document. Cela peut faire penser à la POO (Programmation orientée objet) mais cela ne va pas jusque là.

Mais ceci est une erreur :

```
write("Bonjour tout le monde !");
```

Car on ne précise pas à quel ensemble appartient la méthode *write*. Cela n'affiche rien ou un message d'erreur.

## 2. 2. Les commentaires et les variables en JavaScript

### *Les commentaires*

Comme pour l'immense majorité des langages de programmation, on va également pouvoir commenter en JavaScript. Commenter, c'est laisser des messages ou des indications au sein de son code qui servent à expliquer le code en question.

En JavaScript, il existe deux types de commentaires qui vont s'écrire différemment : les commentaires mono-ligne et les commentaires multi-lignes.

Les commentaires ne sont pas pris en compte dans le traitement des instructions, ils ne servent que de repères et d'aide pour le codeur du script.

En JavaScript, les commentaires ressemblent à ceux utilisés dans le CSS :

```
1 // Deux slashes pour un commentaire sur une ligne
2
3 /* Un slash et une étoile pour un commentaire sur
4    plusieurs lignes...
5 */
```

### *Les variables*

Une variable JavaScript est un conteneur servant à stocker temporairement une information, comme un nombre ou une chaîne de caractères.

Les variables en JavaScript n'ont pas nécessairement besoin d'être déclarées avant de les initialiser ou des les utiliser; mais il est plus correct de le faire.

On doit observer différentes règles lorsque l'on nomme une variable en JavaScript :

- Le nom des variables doit commencer par une lettre ;
- Le nom ne peut *contenir* que des *lettres* (non accentuées), *des chiffres ou les signes* « _ » et « $ » ;
- Le nom des variables est *sensible à la casse* (« a » est différent de « A ») ;
- Le JavaScript possède des *mots « réservés »* comme le mot « var » qu'on ne peut utiliser pour créer une variable. Nous verrons ces mots au fil de ce cours.

La syntaxe pour déclarer une variable est la suivante :

```
1 // Déclaration de la variable nombre
2 var nombre;
```

La déclaration de variable se fait avec l'instruction *var* (variables en anglais (et en français)). Inutile de déclarer le type de variable comme le feraient certains langages de programmation. On peut immédiatement déclarer une valeur par défaut mais cela n'est nullement obligatoire :

```
1 var entier = 5;
2 var float = 1.999992; // type float
3 var chaine = "J'adore l'UVCI!"; // type string
4 var booleen= true; // ou false
5 var objet = null; // type object
```

### *Complément : Nature des variables Javascript*

Les variables en JavaScript peuvent stocker différents types de valeurs.

Très souvent, on parlera souvent de « *types de variables* » au lieu de parler de « *types de valeurs stockées par les variables* ».

Les types de valeurs vont avoir une influence sur notre code, puisqu'on ne stockera pas de la même façon un chiffre ou une chaîne de caractères (un texte) par exemple dans une variable.

Nous ne pourrons pas non plus effectuer les mêmes opérations sur les variables selon le type de valeurs qu'elles stockent.

Les types de valeurs les plus courants que peuvent stocker nos variables en JavaScript sont :

- *Number*
- *String*
- *Boolean*
- *Float*
- *Object*
- *Function*

*Connaître le type de la variable*

L'opérateur *typeof*(`nom_de_la_variable`) permet de connaître le type de la variable si vous ne

les avez pas déclaré.

## 3. 3. Les opérateurs

Tout comme les autres langages, JavaScript dispose d'opérateurs permettant d'effectuer les opérations informatiques de bases.

Le grand avantage des variables est avant tout que l'on va pouvoir les manipuler et notamment effectuer des opérations entre variables.

Pour des variables stockant des valeurs de type Number, on va donc pouvoir effectuer les mêmes opérations qu'avec des nombres en mathématiques.

Ainsi, nous allons pouvoir additionner (les valeurs de) deux variables entre elles, les soustraire l'une à l'autre, les multiplier entre elles, etc.

### *Opérateurs binaires d'affectation*

Comme vu dans les exemples précédents, le symbole "=" (égal) permet d'affecter une valeur (nombre ou chaîne de caractères) à une variable.

```
1 var nombre = 18;
2 var chaine = "Bonjour tout le monde !";
3
4 // Et aussi
5    var a;
6    var b;
7    a = b;
8
9 // Et...
10   a = b = 0; // a et b valent toutes les deux 0
```

### *Opérateurs binaires de calcul*

- L'*addition* s'effectue en utilisant le traditionnel symbole "+" (plus)
- La *soustraction* avec le symbole "-" (moins)
- La *multiplication* avec le symbole "*" (étoile)
- La *division* se sert du symbole "/" (slash)

```
1 var a = b = c = 2; // On initialise a, b et c
2 var d = a/(b+c)-a*4; // d = 2/(2+2)-2*4 = 1/2-8 = -7.5
3 var e = 3*(3+3); // e = 3*6 = 18
```

Il existe aussi les opérateurs +=, -=, *= et /= qui s'utilisent lorsque l'opération s'effectue sur une seule variable. Voici un exemple très simple :

```
1 var a = 3;
2 var b = a-2;
3
4 a -= b; // a = a-b = 3-1 = 2
5 a += b; // a = a+b = 3+1 = 4
6 b *= a; // b = b*a = 1*3 = 3
7 b /= b; // b = b/b = 1
```

### *Opérateurs de comparaisons*

Les opérateurs de comparaison permettent de comparer deux valeurs et renvoient une valeur de type booléen (true, vrai ou false, faux) selon les cas. La syntaxe est simple à comprendre :

*Valeur1 Opérateur de comparaison Valeur2*

- > : est strictement supérieur

- < : est strictement inférieur
- >= : est supérieur ou égal
- <= : est inférieur ou égal
- == : est strictement égal
- != : est strictement différent

Bien entendu il ne faut pas confondre l'opérateur "=" d'affectation et l'opérateur "==" de comparaison

```
var a = b = 10;

document.write(a>b); // false
document.write(a>=b); // true
document.write((a+1)>b); // true
document.write(b!=a); // false
document.write(b==a); // true
document.write(b<=a); // true
```

### *Opérateur de négation*

Comme dans l'exemple précédent, il s'agit du point d'exclamation "!". Ainsi, true devient false et vice-versa.

Exemple :

```
var a = 3;
var b = 5;

document.write(a<b); // affiche true car 3<5
document.write(!(a<b)); // affiche false car l'expression équivaut à a>=b
```

### *Concaténation*

La concaténation est une technique qui consiste à rassembler deux chaînes de caractères pour n'en former qu'une seule. C'est en quelque sorte le collage de ces deux chaînes. Pour ce faire, utilisez le signe "+" et "+=". C'est le même principe que précédemment.

Voici comme vous faisiez avant pour afficher deux chaînes à la suite :

```
var chaine_a = "Bonjour";
var chaine_b = "tout le monde";

// On affiche les deux chaînes à la suite
document.write(chaine_a);
document.write(chaine_b);
```

Et bien désormais, vous pouvez faire ainsi :

```
var chaine_a = "Bonjour";
var chaine_b = "tout le monde";

// On affiche les deux chaînes à la suite avec la concatenation
document.write(chaine_a+chaine_b);
```

On décline un peu...

```
var chaine_a = "Bonjour";
var chaine_b = "tout le monde";

chaine_a += chaine_b; // chaine_a vaut Bonjourtout le monde
```

Notez bien qu'il existe une ambiguïté entre l'opérateur de concaténation et le signe de l'addition en

JavaScript. Cependant, l'interpréteur reconnaît s'il s'agit d'une chaîne donc concaténation ou bien d'un nombre (addition).

### *Opérateurs d'incrémentation et de décrémentation*

Ils sont deux et permettent soit de diminuer de 1 la valeur d'une variable (décrémentation), soit de l'augmenter de 1 (incrémentation).

- *++* : incrémente de 1
- -- : décrémente de 1

```
1 var a = 0;
2 a++; // a = a+1 = 0+1 = 1
3 a--; // a = a-1 = 1-1 = 0
4 a--; // a = -1
```

Notez que si vous placez ces opérateurs devant la variable à incrémenter, l'action sera faite immédiatement ; voici un exemple simple :

```
1 var a = 0;
2 document.write(a++); // Affiche 0
3 document.write(++a); // Affiche 2
4 document.write(--a); // Affiche 1
```

## 4. Exercice : Quiz d'entraînement ...

**Exercice**

*1) Quel(s) est(sont) la (les) syntaxe(s) correcte(s) pour déclarer une variable en JavaScript ?*

- ☐ int var monChiffre = 2;
- ☐ int monChiffre = 2;
- ☐ Number monChiffre = 2;
- ☐ var monChiffre = 2;

**Exercice**

*2) Comment déclare-t-on une chaîne de caractères ?*

- ☐ var chaine = "ma chaine";
- ☐ var maChaine = 'ma chaine';
- ☐ Les deux versions sont pas correctes

**Exercice**

*3) Soit la portion de code JavaScript suivante :*

```
1 var a = 20;
2 var b = c = a-12;
3 var d;
```

*Quelle est la valeur de d +=b ?*

## Exercice

*4) Quel type de commentaire est valide commentaire JavaScript ?*

- ☐ <!-- Commentaire -->
- ☐ /* Commentaire */
- ☐ // Commentaire

## Exercice

*5) Soit la portion de code JavaScript suivante :*

```
1 var a = 10;
2 var b = c = a-12;
3 document.write(++c);
4 document.write(c++);
```

*Quelles affirmations suivantes sont vraies ?*

- ☐ A la fin de ce code, les variables *b* et *c* on la même valeur.
- ☐ La ligne 3 du code affiche la valeur *-1*
- ☐ La ligne 4 affiche la valeur *-2*
- ☐ A la fin de ce code, *c* vaut *0*
- ☐ A la fin de ce code, *c* vaut *-1*
- ☐ La ligne 4 affiche la valeur *-1*