# JavaScript

## CM1-1 : JavaScript

Mickaël Martin Nevot

V4.3.0

# JavaScript


I. Présentation
II. JS
III. Types/opérateurs
IV. Avancé
V. Ajax
VI. DOM
VII. XHR
VIII. JSON
IX. jQuery
X. JS/Web 2.0

# JavaScript

JavaScript ≠ Java



- Extension de fichier : `.js`
- Rappel : méthode recommandée d'utilisation :

```
<head>
    <script src="js/myFile.js"></script>
</head>
```

- Principales caractéristiques :
  - Interprété **coté client**
  - Langage de programmation de **script** sensible à la casse
  - Conçu pour le développement d'**applications web**
  - Langage **orienté objet non-typé**
  - Standard **ECMAScript** (comme ActionScript)
  - Pas de lecture/écriture ou d'exécution d'autres programmes
- Mode strict : `"use strict";`

# Commentaires JavaScript

- Non interprétés par le navigateur
- Visibles dans le code source
- Commentaires de type C/C++ et Shell Unix
- Exemple :

```
// Commentaire JavaScript(une seule ligne).
# Autre commentaire JavaScript (une seule ligne).
/* Autre commentaire JavaScript (une ou plusieurs lignes). */
/*
    Commentaire JavaScript
    (sur plusieurs lignes).
*/
```

# Variable et constante

- Définition de type :
  - **Pas de déclaration explicite** du type d'une variable
  - Type d'une variable déterminé par le **contexte d'utilisation**
  - **Conversion automatique**
- Variable : `let`

```
let a;
let name = "John";
```

- Constante (variable non modifiable) : `const`

```
const B = 3;
B = 2;   // Erreur !
const C; // Erreur !
C = 4;   // Erreur !
```

- Portée : lexicale
  - Globale : durée de vie dans tout le script
  - Locale : durée de vie limitée au bloc de sa déclaration

# Structures conditionnelles

## Structure conditionnelle :

```
if (a == 0) {
    ++a;
} else if (a == 1) {
    --a;
} else {
    a = 0;
}
```


## Branchement conditionnel :

```
switch(i) {
  case "jambon":
      alert("salé");
      break;
  case "tarte":
  case "bonbon":
  case "biscuit":
      alert("sucré");
      break;
  default:
      alert("autre");
  }
```

# Boucles

- Boucle `for` :

```
let res = "";
for (let i = 0 ; i < 10 ; ++i) {
   res += i + ", ";
}
alert(res);
// 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, .
```

- Boucle `for ... in` :

```
let res = "";
let arr = [1, 2, 3, 4, 5];
for (let val in arr) {
    res += (val * 2) + ", ";
}
alert(res);
// 2, 4, 6, 8, 10, .
```

- Boucle `while` :

De 0 à n fois

```
let i = 0;
while(i < 0) {
   ++i;
}
alert(i);
// 0.
```

- Boucle `do ... while` :

De 1 à n fois

```
let i = 0;
do {
   ++i;
} while (i < 0);
alert(i);
// 1.
```

# Break/continue

- Sortir d'une boucle ou d'un `switch ... case : break`

```
while(1) {
   if (i == 10) {
      break;
   }
   ++i;
}
// Exemple de break dans un switch sur la diapositive de structures conditionnelles.
```

- « Sauter » à l'itération suivante d'une boucle : `continue`

```
let a = 0;
for  (let i = 0 ; i < 10 ; ++i) {
   if (i % 2 == 0) {
      continue;
   }
   a += i;
}
// ?
```

# Instruction

- Se termine par un `;`
- Type d'instruction :
  - Déclaration : `let a;`
  - Affectation : `a = 10;`
  - Appel de fonction : `myFunction();`
  - Instruction conditionnelle : `if (a == b) … ;`
  - Instruction vide : `;`
  - Bloc (d'instruction) :

```
{
    instruction_1
    instruction_2
    …
}
```

# Crédits

## Auteur

Mickaël Martin Nevot
mmartin.nevot@gmail.com

Carte de visite électronique

## Relecteur

Cours en ligne sur : www.mickael-martin-nevot.com