

# Apprendre à programmer en JavaScript

Cours de Développement Côté Serveur - TD

1

## PARTIE 1 : Écrire ses premières lignes de code

Cours de Développement Côté Serveur - TD

2

1

# 1. Introduction à la programmation

- Un **ordinateur** est une machine qui se contente d'exécuter automatiquement, vite et sans erreur, les opérations qu'on lui demande d'effectuer
- Un **programme** est une liste d'ordres, de commandes, indiquant à l'ordinateur ce qu'il doit faire
- Le rôle du **programmeur** (développeur) est de créer ces programmes
- Pour cela, il utilise des **langages de programmation** :
  - assembleur,
  - Python,
  - php,
  - Java,
  - C, C++,
  - JavaScript, ...
- Le langage de programmation définit la manière de donner les ordres à l'ordinateur. Tout langage a
  - son **vocabulaire** (ensemble de mots-clés) ;
  - sa **syntaxe** (ensemble des règles de grammaire définissant la manière d'écrire les programmes dans ce langage)
- Chaque langage a sa propre syntaxe mais les **concept**s sont les mêmes dans tous les langages :
  - variables,
  - fonctions,
  - boucles,
  - tests, ...

3

```

1 // Expérience du personnage
2
3 const aurora = {
4   nom: "Aurora",
5   sante: 150,
6   force: 25,
7   xp: 0,
8
9   // Renvoie la description du personnage
10  decrire() {
11    return `${this.nom} a ${this.sante} points de vie, ${
12      this.force
13    } en force et ${this.xp} points d'expérience`;
14  }
15 };
16
17 // "Aurora a 150 points de vie, 25 en force et 0 points d'expérience"
18 console.log(aurora.decrire());
19
20 console.log("Aurora apprend une nouvelle compétence");
21 aurora.xp += 15;
22
23 // "Aurora a 150 points de vie, 25 en force et 15 points d'expérience"
24 console.log(aurora.decrire());
25

```

## Un exemple de code JavaScript

- Chaque ligne de commande s'appelle une **instruction**
- L'ensemble des instructions d'un programme s'appelle le **code source**
- Pour être compris par l'ordinateur, le programme doit respecter les règles du langage utilisé

4

## La notion d'algorithme

- Avant d'écrire un programme, il faut analyser le problème
- Un **algorithme** est une suite ordonnée d'opérations qui permet de résoudre le problème : il décompose un problème complexe en une suite d'opérations simples
- On peut distinguer différents types d'actions :
  - des actions **simples** ("Sortir une casserole") ;
  - des actions **conditionnelles** ("Si on préfère le beurre à l'huile...") ;
  - des actions **qui se répètent** ("Tant que les pâtes sont trop fades...").

```

1 Début
2 Sortir une casserole
3 Mettre de l'eau dans la casserole
4 Ajouter du sel
5 Mettre la casserole sur le feu
6 Tant que l'eau ne bout pas
7   Attendre
8 Sortir les pâtes du placard
9 Verser les pâtes dans la casserole
10 Tant que les pâtes ne sont pas cuites
11   Attendre
12 Verser les pâtes dans une passoire
13 Egoutter les pâtes
14 Verser les pâtes dans un plat
15 Goûter
16 Tant que les pâtes sont trop fades
17   Ajouter du sel
18   Goûter
19 Si on préfère le beurre à l'huile
20   Ajouter du beurre
21 Sinon
22   Ajouter de l'huile
23 Fin
  
```



C'est prêt !

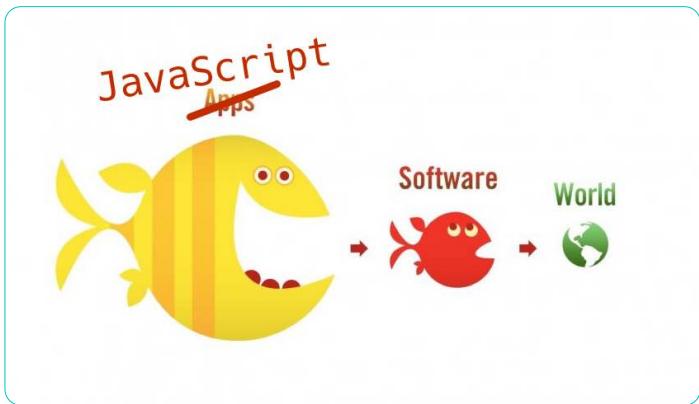
5

## 2. Présentation de JavaScript

- **JavaScript** est le langage de programmation du Web
- Il a été **inventé** en **1995** par **Brendan Eich**, qui travaillait pour Netscape (l'ancêtre de Firefox)
- Idée de départ : créer un langage simple pour rendre dynamiques et interactives les pages Web
- Petit à petit, il est intégré à l'ensemble des navigateurs, tous capables aujourd'hui d'exécuter du code JS
- L'avènement du Web 2.0, basés sur des pages riches et interactive, le rend de plus en plus populaire. Il devient un langage très performant
- **2009** : apparition de la plate-forme **Node.js** (permet d'écrire en JS des applications Web très rapide) et de MongoDB (monde des bases de données)
- L'arrivée des smartphones et tablettes dotés de systèmes différents et incompatibles (iOS, Android, Windows Phone) a conduit à l'apparition d'outils de développement dits multi-plateformes, presque toujours basés sur... JavaScript !

6

3



## JavaScript : un langage essentiel

- Historiquement créé pour animer les pages web, JS est maintenant présent partout
- Sa connaissance vous ouvrira les portes de la programmation côté navigateur Web (développement front-end), côté serveur (back-end) ou côté mobile
- À l'heure actuelle, beaucoup le considèrent comme la technologie la plus importante dans le monde du développement logiciel
- Excellent 1<sup>er</sup> langage de programmation car :
  - Dimension universelle
  - Facilité d'accès

7

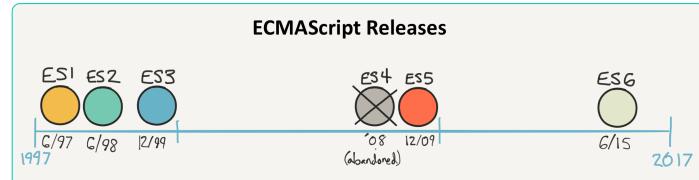
## Les versions de JavaScript

○ JavaScript a été **standardisé** en **1997** sous le nom d'**ECMAScript**. Depuis, il a subi plusieurs séries d'améliorations (corrections, nouvelles fonctionnalités)

○ On verra dans ce cours

- d'abord **la version ES5** de JavaScript pour les bases
- et puis **ES6 (ES2015)**. Cette version est maintenant bien supportée par les navigateurs modernes.

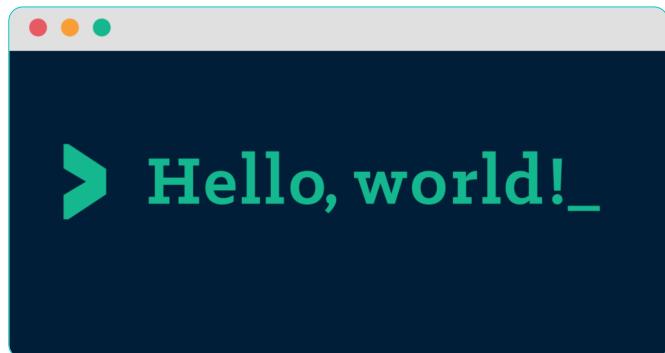
ECMAScript Releases



8

3.

3, 2, 1...  
Codez !



9

## Un premier programme

- C'est le moment de faire vos premiers pas avec JS !

Voici votre tout premier programme :

```
1 console.log("Bonjour en JavaScript !");
```

javascript

- Pour le tester, il vous suffit de taper les instructions dans la console JavaScript de votre navigateur
- L'instruction JavaScript `console.log()` permet d'afficher une information dans la console
- N.B. Il existe d'autres possibilités pour tester du code JavaScript :
  - Utiliser un "bac à sable" en ligne, comme par exemple CodePen ;
  - Installer la plate-forme Node.js sur votre ordinateur.

10

## 3. 3, 2, 1... Codez !

### 3. 2. Valeurs et types

- Une **valeur** est un morceau d'information utilisé dans un programme informatique
- Les valeurs peuvent être de différents types : nombre, texte, booléen, ...
- Le **type** d'une valeur détermine son rôle et les opérations qui lui sont applicables
- Il y a des données
  - de **type simple**, on dit aussi de **type primitif** (une seule valeur)
  - de **type complexe** (comme les tableaux ou les objets, que nous verrons plus tard)
- Commençons par les types simples

11

## 3. 3, 2, 1... Codez !

### 3. 2. Valeurs et types

#### 1. TYPES PRIMITIFS

Voici les 5 types primitifs disponibles en JavaScript :

1. le type **nombre**
2. le type **chaîne**
3. le type **booléen**
4. le type **undefined**
5. le type **null**

12

## 3. 3, 2, 1... Codez !

### 3. 2. Valeurs et types – TYPES PRIMITIFS

#### 1. Le type nombre

- Une **valeur** de type **nombre** (Number) représente une valeur numérique, une quantité
- Les nombres servent à compter
- Avec des valeurs de type nombre, on peut faire des opérations mathématiques
- Ces opérations produisent un **résultat** de type nombre

OPÉRATEUR	DESCRIPTION	USAGE	RÉSULTAT
+	Addition	2+2	4
-	Soustraction	4-2	2
*	Multiplication	2*5	10
/	Division	10/5	2
++	Ajouter 1 à un nombre	2++	3
--	Soustraire 1 à un nombre	3--	2
%	Calculer le reste d'une division	12 % 5	2

13

#### Un exercice avec l'opérateur modulo

- L'opérateur **modulo** donne le reste de la division entière d'un nombre par un autre.

- **EXEMPLE 1 : Tapez dans la console 4%2**

- **4 % 2 = 0 :**  
le reste de la division entière de 4 par 2 est 0

$$\begin{array}{r} 4 \mid 2 \\ 4 \overline{)2} \\ 0 \end{array}$$

- 4 est divisible par 2,  
4 est un multiple de 2

- **EXEMPLE 2 : Tapez dans la console 9%2**

- **9 % 2 = 1 :**  
le reste de la division entière de 9 par 2, est 1

$$\begin{array}{r} 9 \mid 2 \\ 8 \overline{)2} \\ 1 \end{array}$$

- 9 n'est pas divisible par 2  
9 n'est pas un multiple de 2

- Le modulo permet de savoir si un nombre est divisible par un autre, ou si un nombre est un multiple d'un autre : **si le modulo vaut 0, c'est qu'on a un multiple.**

14

## 3. 3, 2, 1... Codez !

### 3. 2. Valeurs et types – TYPES PRIMITIFS

#### 2. Le type chaîne

- Une valeur de type **chaîne de caractères** (*String*) représente un texte
- Ces valeurs sont délimitées par une paire de guillemets simples ou doubles :

`"Ceci est une chaîne"`

- On ne peut pas additionner ou soustraire des valeurs de type chaîne
- Mais on peut appliquer l'opérateur `+` à deux valeurs de type chaîne :
  - il fusionne les deux chaînes en une seule. Cette opération est appelée **concaténation**.
  - Par exemple : `"Bon"+"jour"` produit le résultat "Bonjour"

15

## Quelques remarques sur les valeurs de type chaîne



Il est également possible de délimiter une chaîne de caractères avec une paire de guillemets simples : `'Ceci est aussi une chaîne'`. Par convention, nous emploierons les guillemets doubles dans ce cours. L'important est d'être cohérent : utilisez l'une ou l'autre notation, mais ne mélangez pas les deux.



Il ne faut surtout pas oublier de "fermer" une chaîne : simples ou doubles, les guillemets vont toujours par deux !



Pour inclure dans une chaîne certains caractères spéciaux, on utilise le caractère `\` (qui se prononce "antislash" ou "backslash" en anglais) qui donne un sens particulier au caractère suivant. Par exemple, `\n` permet d'ajouter un retour à la ligne dans une chaîne.

16

## 3. 3, 2, 1... Codez !

### 3. 2. Valeurs et types – TYPES PRIMITIFS

#### 3. Le type booléen

Le type **booléen** (Boolean) est un type de données logique qui ne possède que deux valeurs possibles :

soit `true` (vrai, 1)

soit `false` (faux, 0)

#### 4. Le type `undefined`

Le type **undefined** est le type attribué à une variable qui ne possède pas encore de valeur (qui n'a pas encore été définie)

#### 5. Le type `null`

Le type **null** représente une non-valeur : un élément qui a été défini mais qui n'a aucune valeur intrinsèque

17

## 3. 3, 2, 1... Codez !

### 3. 3. Structure d'un programme

Un programme informatique se compose de plusieurs lignes de code, plusieurs ordres, qui s'exécutent successivement.

#### 1. Instructions

- Chaque ordre est appelé une **instruction**
- Une instruction est délimitée par un **point virgule**
- Un programme est donc une suite d'instructions

```
1 let a;
2 a = 3.14;
3 console.log(a);
```

18

## 3. 3, 2, 1... Codez !

### 3. 3. Structure d'un programme

#### 2. Déroulement de l'exécution

- Par défaut :
  - Lorsqu'un programme est exécuté, les instructions qui le composent sont "lues" les unes après les autres
  - Chaque instruction produit un résultat, et c'est la combinaison de ces résultats qui produit le résultat final du programme
  - Par défaut, les instructions s'exécutent **toutes**, les unes après les autres, **dans l'ordre où elles ont été écrites dans le code**

19

## 3. 3, 2, 1... Codez !

### 3. 3. Structure d'un programme

#### 2. Déroulement de l'exécution

- On peut utiliser des **structures de contrôle** pour contrôler la manière dont les instructions vont s'exécuter :
    - les **tests (expressions conditionnelles)** permettent de **créer des branchements** et de ne **pas** exécuter certaines instructions :
      - si la condition est vraie, alors on exécute une instruction ;
      - sinon, on en exécute une autre

C'est ce qu'on fera avec des instructions comme `if / else`, ou `switch` par exemple.
    - les **boucles** permettent de répéter plusieurs fois une même instruction avant de passer à la suite : tant qu'une condition n'est pas remplie, on répète
- C'est ce qu'on fera avec des instructions comme `while`, ou `for`)

20

## 3. 3, 2, 1... Codez !

### 3. 3. Structure d'un programme

#### 3. Commentaires

- Les commentaires sont des portions de code non interprétées.
- Ils permettent de documenter le fonctionnement d'un programme
- En JS, on délimite les commentaires
  - par // ... pour commenter une ligne ;
  - par /\* ... \*/ (comme en CSS) :

```
1 /* Un commentaire
2 sur plusieurs
3 lignes */
4
5 // Un commentaire sur une seule ligne
```

21

## 3. 3, 2, 1... Codez !

### 3. 4. À vous de jouer !

1. EXERCICE 1 : Présentation
2. EXERCICE 2 : Mini-calculatrice
3. EXERCICE 3 : Valeurs affichées

22