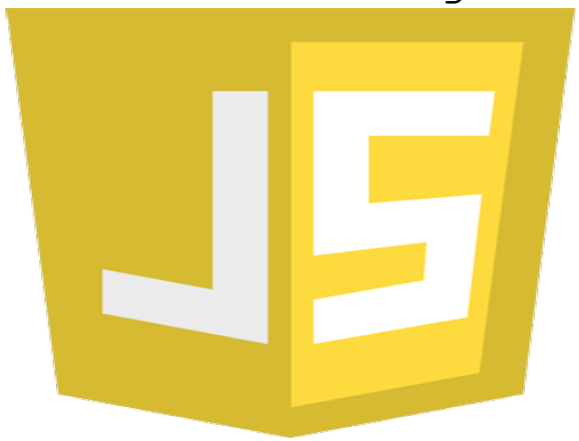
JavaScript : introduction

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com



Plan

JavaScript

JavaScript

- Langage de programmation de scripts
  - procédural
  - orienté objet (à prototype)
- Créé par **Brendan Eich**
- Présenté par **Netscape** et **Sun Microsystems** en décembre 1995
- Standardisé par **EcmaScript** depuis juin 1997
- Permettant de
  - compléter l'aspect algorithmique manquant au **HTML** (et **CSS**)
  - rendre plus vivant le site web avec notamment des animations, effets et de l'interaction avec l'internaute (le visiteur, le client...).

JavaScript

Autres langages de programmation de scripts

- **AppleScript**
- **JScript**, **VBScript** et **TypeScript** (de **Microsoft**)
- **LiveScript** (de **Netscape**) puis **JavaScript**
- **ActionScript** (de **MacroMedia**)
- **CoffeeScript** (Open-source)
- ...

JavaScript

Spécificités du **JavaScript**

- langage faiblement typé
- sensible à la casse
- syntaxe assez proche de celle de **Java**, **C++**, **C**...
- possibilité d'écrire plusieurs instructions sur une seule ligne en les séparant par ;

JavaScript

JavaScript : langage séquentiel

- l'exécution du code suit l'ordre des instructions : de haut en bas
- pour une ligne de code donnée, l'exécution se fait de gauche à droite, selon l'ordre des opérations et la syntaxe du langage

JavaScript

JavaScript : langage séquentiel

- l'exécution du code suit l'ordre des instructions : de haut en bas
- pour une ligne de code donnée, l'exécution se fait de gauche à droite, selon l'ordre des opérations et la syntaxe du langage

Cependant

Il est important de noter que **JavaScript** possède également des caractéristiques qui permettent une exécution non séquentielle notamment avec les fonctions asynchrones et les callback.

JavaScript

Quel organisme gère t-il la standardisation du langage **JavaScript** ?

Le comité 39 d'un organisme international appelé l'**ECMAScript**.

JavaScript

ECMA

- Pour **E**uropean **C**omputer **M**anufacturers **A**ssociation.
- Créé en 1961.
- Devenu **Ecma International** - European association for standardizing information and communication systems en 1994.
- Organisme de standardisation pour plusieurs domaines de l'informatique
  - les langages de script
  - les produits de sécurité
  - la structure de fichier
  - ...

JavaScript

ECMAScript

- Ensemble de normes sur les langages de programmation de type script
  - **JavaScript**
  - **VBScript**
  - **AppleScript**
  - ...
- Standardisé par **Ecma International** depuis 1994

JavaScript

Quelques versions

- Version 5 (**ES5**) ou **ES2009**
- Version 6 (**ES6**) ou **ES2015** (compatible avec les navigateurs modernes)
- Version 7 appelé **ES2016** (ne s'appelle plus ES7)
- Version 8 appelé **ES2017**
- Version 9 appelé **ES2018**
- Version 10 appelé **ES2019**
- Version 11 appelé **ES2020**
- Version 12 appelé **ES2021**

JavaScript

Remarques

- Tous les navigateurs web doivent respecter au moins la version **ECMAScript 5.1**.
- Tous les navigateurs web modernes respectent les **ECMAScript** de 1 à 6.

JavaScript

De quoi on a besoin ?

- un navigateur
- un éditeur de texte

JavaScript

Quelques navigateurs

- Google chrome : `https://www.google.com/chrome/`
- Mozilla firefox : `https://www.mozilla.org/fr/firefox/new/`
- Edge (installé par défaut sous **Windows**)
- ...

JavaScript

Quelques éditeurs de texte

- **Sublime text** : `https://www.sublimetext.com/3`
- **Atom** : `https://atom.io/`
- **Notepad++** : `https://notepad-plus-plus.org/`
- **Brackets** : `http://brackets.io/`
- ...

JavaScript

Quelques éditeurs de texte

- **Sublime text** : `https://www.sublimetext.com/3`
- **Atom** : `https://atom.io/`
- **Notepad++** : `https://notepad-plus-plus.org/`
- **Brackets** : `http://brackets.io/`
- ...

CodePen : une solution en ligne

Trois éditeurs en parallèle : un pour **HTML**, un pour **CSS** et un pour **JavaScript**
`https://codepen.io/`

JavaScript

Utiliser un **IDE** (**E**nvironnement de **D**éveloppement **I**ntégré) ?

- Console auto-intégrée
- Auto-complétion
- Auto-compilation
- Coloration syntaxique
- Meilleure structuration du projet

JavaScript

Quel **IDE** pour **JavaScript** ?

- **Visual Studio Code**
- `code.visualstudio.com/download`

JavaScript

Quel **IDE** pour **JavaScript** ?

- **Visual Studio Code**
- `code.visualstudio.com/download`

Visual Studio Code (ou VSC) , pourquoi ?

- Gratuit
- Multi-langage
- Multi-système d'exploitation
- Extensible via l'installation de quelques centaines d'extensions

JavaScript

Quelques raccourcis **VSC**

- Pour activer la sauvegarde automatique : aller dans `File > AutoSave`
- Pour indenter son code : `Alt` `Shift` `f`
- Pour commenter/décommenter : `Ctrl` `:`
- Pour sélectionner toutes les occurrences : `Ctrl` `f2`
- Pour sélectionner l'occurrence suivante : `Ctrl` `d`
- Pour placer le curseur dans plusieurs endroits différents : `Alt`

JavaScript

Pour créer un projet sous **VSC**

- Allez dans `File > Open Folder...`
- Cliquez sur `Nouveau dossier` et saisissez `cours-js`
- Cliquez sur le dossier `cours-js` puis sur le dossier `Sélectionner un dossier`
- Créez un fichier `index.html` dans `cours-js`
- Dans `index.html`, saisissez `html:5` ou `!` puis cliquez sur `Entree`

JavaScript

Code généré

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>

<body>

</body>

</html>
```

JavaScript

Extension **Live Server**

- Installez l'extension **Live Server**
- Faites un clic doit sur `index.html`
- Cliquez sur `Open with Live Server`

JavaScript

Intégrer **JavaScript** dans **HTML**

Trois façons pour définir des scripts **JavaScript**

- comme valeur d'attribut de n'importe quelle balise **HTML**
- dans une balise `<script>` de la section `<head>` d'une page **HTML**
- dans un fichier d'extension `.js` référencé dans une page **HTML** par la balise `<script>`

JavaScript

Première méthode

```
<button onclick="alert('Hello World');">
    Cliquer ici
</button>
```

JavaScript

Deuxième méthode

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>First JS Page</title>
    <script type="text/javascript">
        function maFonction() {
            alert("Hello World !");
        }
    </script>
</head>
<body>
  <button onclick="maFonction()">
    Cliquer ici
  </button>
</body>
</html>
```

JavaScript

Deuxième méthode

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>First JS Page</title>
    <script type="text/javascript">
        function maFonction() {
            alert("Hello World !");
        }
    </script>
</head>
<body>
  <button onclick="maFonction()">
    Cliquer ici
  </button>
</body>
</html>
```

L'attribut `type="text/javascript"` n'est plus nécessaire depuis **HTML 5**.

JavaScript

Troisième méthode

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>First JS Page</title>
    <script src="script.js"></script>
</head>

<body>
    <button onclick="maFonction()"> Cliquer ici</button>
</body>

</html>
```

JavaScript

Troisième méthode

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>First JS Page</title>
    <script src="script.js"></script>
</head>

<body>
    <button onclick="maFonction()"> Cliquer ici</button>
</body>

</html>
```

Contenu du `script.js`

```
function maFonction() {
    alert("Hello World !");
}
```

JavaScript

Pour une box d'affichage avec confirmation

```
var bin = confirm("Press a button!");
alert(bin);
```

JavaScript

Pour une box d'affichage avec confirmation

```
var bin = confirm("Press a button!");
alert(bin);
```

Pour une box d'affichage avec une zone de saisie

```
var str = prompt("Votre nom", "John Wick");
alert(str);
```

JavaScript

Ajoutons `debugger` dans la fonction à déboguer

```
function maFonction() {
    debugger
    alert("Hello World !");
}
```

JavaScript

Ensuite, il faut

- ouvrir le `DevTools` du navigateur
- aller dans l'onglet `Sources`
- cliquer sur le fichier **JavaScript**
- cliquer sur le bouton, ensuite sur le bouton ▷ pour l'exécution pas-à-pas.

JavaScript

La console, pourquoi ?

- permet de contrôler l'avancement de l'exécution d'un programme
  - en affichant le contenu de variables (déboguer)
  - en vérifiant les blocs du code visites (tracer)

JavaScript

La console, pourquoi ?

- permet de contrôler l'avancement de l'exécution d'un programme
  - en affichant le contenu de variables (déboguer)
  - en vérifiant les blocs du code visites (tracer)

Modifions le contenu du `script.js`

```
function maFonction(){
  console.log("Hello World !");
}
```

JavaScript

Où peut-on trouver le message ?

- Pour les navigateurs suivants
  - Google chrome
  - Mozilla firefox
  - Internet explorer
- Cliquer sur F12

JavaScript

Existe t-il un autre moyen de tester un code **JavaScript** sans passer par un navigateur ?

- Oui, en utilisant **NodeJS** (pour télécharger `https://nodejs.org/en/`)
- Pour tester, utiliser une console telle que
  - Invite de commandes
  - Windows PowerShell
  - Cmder
- Lancer la commande `node nomFichier.js`

JavaScript

Modifions le contenu du `script.js`

```
function maFonction(){
    console.log("Hello World !");
}
maFonction();
```

JavaScript

Modifions le contenu du `script.js`

```
function maFonction(){
    console.log("Hello World !");
}
maFonction();
```

Lancer la commande `node script.js`

JavaScript

Il est aussi possible de définir un raccourci de `console.log`

```
var cl = console.log;
cl("Hello World !");
```

JavaScript

Commentaire sur une seule ligne

```
// commentaire
```

JavaScript

Commentaire sur une seule ligne

```
// commentaire
```

Commentaire sur plusieurs lignes

```
/* le commentaire
la suite
et encore la suite
*/
```

JavaScript

Commentaire sur une seule ligne

```
// commentaire
```

Commentaire sur plusieurs lignes

```
/* le commentaire
la suite
et encore la suite
*/
```

Commentaire pour la documentation

```
/** un commentaire
pour
la documentation
*/
```