Introduction au langage Javascript

1. Préambule

Le Javascript est aujourd'hui un langage incontournable du developpement Web.
C'est un langage de programmation **multi-paradigme** : orienté objet à prototype, impératif et fonctionnel *(il partage ces deux dernières caractéristiques avec Python par exemple)*.

Le langage a été créé *Netscape* (fin 1995) est standardisé sous le nom d'ECMAScript (c'est peut-être un terme que vous rencontrerez - notamment sous sa forme abrégée pour désigner les différentes versions du standard : *ES5*, *ES6*, etc.).

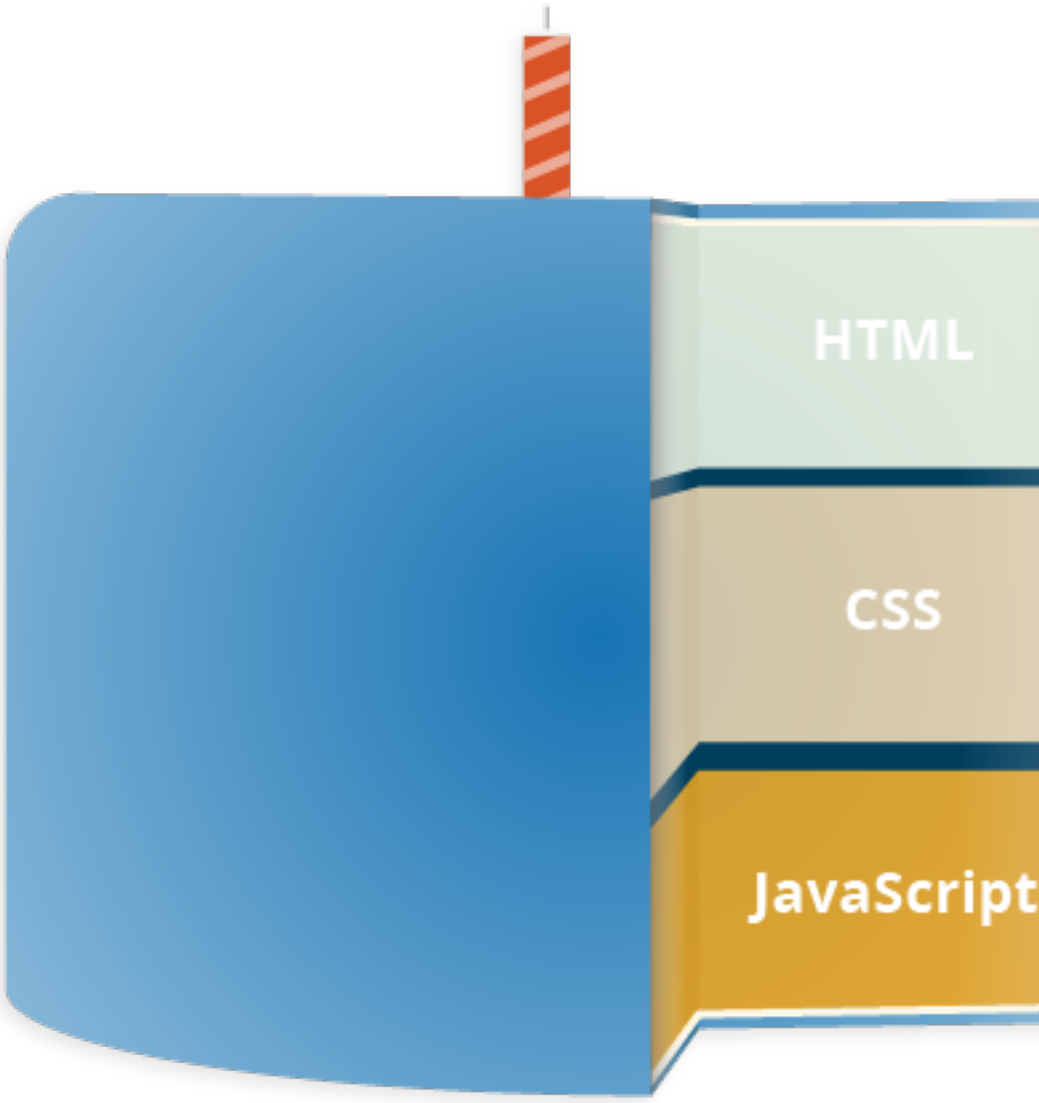
Sa syntaxe est dérivée de celle du C/C++.

```
for (let i = 0; i < 10; i++) {
  console.log("Iteration n°" + i);
}

var j = 0;
while (j < 10) {
  console.log("...");
  j++;
}
```

Ses autres particularités sont les suivantes :

- Langage **interprété** (pas de compilation, exécution directe dans un interpréteur)
- **Typage dynamique** (laisse l'ordinateur réaliser l'opération de typage à la volée)
- Modèle objet à base de **prototypes**
- Langage **fonctionnel** (les fonctions sont des *objets de première classe* : on peut passer des fonctions en argument à des fonctions, les assigner à des variables, les stocker dans des structures de données, etc.)



Pourquoi le JavaScript ?

Source: https://developer.mozilla.org/fr/docs/Learn/JavaScript/First_steps/What_is_JavaScript

- Couche **“Comportement”** (JavaScript) qui s’ajoute naturellement à la couche **“Présentation”** (CSS) et à la couche **“Contenu structuré”** (HTML) des technologies web.
- Création de contenus complexes, dynamiques et interactifs dont la logique est exécutée coté client, le moment venu.
- Les navigateurs web disposent de plusieurs **API** *(Application Programming Interfaces)* qui permettent de manipuler le document (*API DOM*), de manipuler des contenues 2D ou 3D (*API Canvas* et *API WebGL*) ou les informations de géolocalisation (*API Geolocation*) par exemple ; ces API sont accessibles en JavaScript.

1.1 Quelques idées fausses

- Ce n'est toutefois pas le langage du Web. En effet d'autres solutions ont été developpées pour s'éxecuter coté client (c'est par exemple le cas de Adobe Flash ou des *applets* Java). En revanche, avec la normalisation du langage et un meilleur support par tous les navigateurs et avec l'introduction de HTML5 (et de ses nombreuses API JavaScript), le Javascript a effectué un retour en force dans l'écosystème Web coté client
- Le Javascript n'est pas seulement utilisé coté client mais également de plus en plus comme un langage de script à part entière, notamment depuis l'existence de node.js.

1.2 Environnements d'exécution JavaScript

Un environnement d'exécution Javascript est composé d'un **moteur d'exécution** JavaScript *(runtime)* et d'objets JavaScript spécifiques à l'environnement en question.

Les principaux environnement d'exécutions sont les **navigateurs Web** (*Chrome* utilise le moteur *V8*, *Firefox* utilise *SpiderMonkey* et *Safari* utilise *WebKit*).

Node.js est un environnement d'éxecution célèbre (utilisant lui aussi le moteur *V8*) qui n'est lié à un aucun navigateur Web ; il permet ainsi l'exécution de code Javascript hors du contexte du navigateur.

1.3 Intégration dans une page HTML

- L'intégration de code JavaScript se fait au moyen de la balise `<script> </script>`.
- Elle peut se faire à deux endroits de la page HTML : dans l'en-tête ou dans le corps de la page HTML.

- Le chargement et l'exécution de ces scripts respecte l'ordre d'apparition dans la page.

```
<html>
  <head>
    <script>
      // Code Javascript Ici
    </script>
    <script src="monScript.js"></script>
  </head>
  <body>
    <header>Contenu ...</header>
    <main>
        Contenu de la page
    </main>
    <footer>Contenu ...</footer>
    <script>
      // Code Javascript Ici
    </script>
    <script src="unAutreScript.js"></script>
  </body>
</html>
```

> **Remarque**: * Les scripts placés dans l'en-tête (<head>) sont executés avant la contruction et l'affichage de la page. * Les script placés dans le corps (<body>) sont exécutés au fur et à mesure que la page HTML est lue et que le DOM se construit.

Comme pour l'ajout de règles CSS, il existe donc bien deux manières *(rédactions du code JavaScript dans la page HTML ou inclusion d'un fichier externe)* d'intégrer un script JavaScript.

1.4 Votre environnement de travail

Vous utiliserez les mêmes outils (éditeur de code de votre choix et navigateur Web) que lors des premières séance de TP.

Comme lors des autres séances **il est indispensable de travailler avec la fenêtre des outils pour développeurs (*DevTools* - accessible avec F12) de votre navigateur - celui-ci comprend en effet une console JavaScript**.

2. Les base du langages

> **Liens utiles sur le JavaScript:** * **Mozilla Developpers Network - Premier pas en JavaScript** : https://developer.mozilla.org/fr/docs/Learn/JavaScript/First_steps * **W3schools Javascript Tutorial** : https://www.w3schools.com/js/ * **Eloquent JavaScript** - 3rd edition (online, PDF, E-PUB, etc.) : https://eloquentjavascript.net/ * Pour se faire la main en JavaScript il faut pratiquer… des exercices comme ceux de W3schools puis des sites comme JavaScript30 peuvent donner de bonnes occasions de pratiquer !

2.1 Éléments de base pour JavaScript dans le navigateur

→ Ouvrez un navigateur et une console JavaScript et saisissez les commandes qui suivent une-à-une (validation avec Enter) et observez bien le résultat!

```
console.log("Contenu du message à afficher");
```

```
document // Pour manipuler des objets sur la page
```

```
// Ne va pas fonctionner si aucun élément de la page n'a cet id !
// Trouver l'id d'un élément de la page et modifiez le de la manière
 ↪  suivante !
document.getElementById('myElem').innerHTML = "New content !"
```

```
// Ne va pas fonctionner si aucun élément de la page n'a cet id !
// Trouver l'id d'un élément de la page et modifiez le de la manière
 ↪  suivante !
document.querySelector('#myElem').innerHTML = "New content !"
```

```
// Ne va pas fonctionner si aucun élément 'div' n'a cette classe !
document.querySelector('div.myClass').innerHTML = "New content too !"
```

```
document.title
```

```
window // Objet relatif à la fenêtre
```

```
window.alert("Contenu de l'alerte !")
```

```
window.innerHeight // un entier
```

```
window.innerWidth // un entier
```

```
window.screen // un Object JavaScript
```

```
 // Un Object décrivant l'URL actuelle du document
window.location
```

```
 // On accère à la propriété 'href' de cet objet
 // ... c'est la chaine de caractère qui correspond à l'URL actuelle:
window.location.href
```

2.2 Variables

Elles sont utilisées pour stocker des données dans la mémoire de l'ordinateur *(on associe une valeur à un nom de variable)*.

Vous rencontrerez deux type de variables :

- types primitifs : notations litérales (2, `3.14`, `"Chaine de caractère"`, `'une autre chaine'`) et valeurs natives *(built-ins)* (`true`, `false`, `undefined`, `null`, `NaN`, `Infinity`)
- objets JavaScript (par exemple `Array` pour un tableau)

```
// On peut déclarer un variable sans lui affecter de valeur:
var a; // équalivent à `var a = undefined;`
var b, c;

console.log(a); // undefined

// Affectation avec la forme littérale
// (chaine de caractère, entier, réel)
a = "Coucou";
b = 1;
c = 3.14;

// On peut affecter une valeur lors de la déclaration
var d = "Hello";

// On peut ensuite affecter une valeur d'un autre type
a = 12;


// Tableau d'entiers (forme littérale)
var myTableau = [1, 2, 3, 4];

// Objet JavaScript (forme littérale)
var myObj = {
  "rate": 5,
  "comment": "Awesome place ! Great Food !"
};
```

> **Remarque:** Historiquement la déclaration de variable se fait avec `var`. Depuis **ES6** la bonne pratique est d'utiliser `let` ou `const`. Cette fonctionnalité est maintenant supportée par les différents navigateurs Web.

- `const` :

```
const myConst = 12;

// On ne peut pas faire :
myConst = 15; // TypeError : invalid assignement to const 'myConst'
```

- `let`:

```
let isValid = false;

// later in your code :
isValid = true;
```

2.3 Opérateur arithmétiques, logiques et de comparaison

- `+`, `-`, `*`, `/`, `%`
- `++`, `--`
- `==`, `!=`, `>`, `<`, `>=`, `<=`
- `||` (pensez à `or` en Python), && (pensez à `and` en Python), `!` (pensez à `not` en Python)

2.4 Contrôle du flux d'exécution

- Lorsque le navigateur web rencontre du code JavaScript il l'exécute dans l'ordre d'apparition (à quelques exceptions qui ne sont pas abordées).
- Contrairement au Python, l'indentation ne joue pas de rôle dans la logique du code *(celà ne veut pas dire qu'il ne faut pas respecter certaines règles! pensez à la lecture du code par vous ou d'autres développeurs!)*
- Instructions conditionnelles : `if`, `else`, `else if` `switch`
- Instructions itératives : `for`, `while`

```
// Définition d'une fonction créant un paragraphe
// et l'ajoutant à la fin de la page
// (la fonction est définie, on pourra l'utiliser avec son nom
 ↪  'createParagraph'
// mais elle n'est pas exécutée lors de sa définition, seulement lorsqu'elle
// sera invoquée par la suite).
function createParagraph() {
  let para = document.createElement('p');
  para.textContent = 'You clicked the button!';
```

```
  document.body.appendChild(para);
}

// Créé un tableau référencant tous les éléments 'button':
let buttons = document.querySelectorAll('button');

// Utilisation d'une boucle pour parcourir le tableau dans son ensemble
for(let i = 0; i < buttons.length ; i++) {

  // Ajoute un "click event listener" sur chaque bouton
  // lors du clic, la fonction 'createParagraph'
  // sera appelée:
  buttons[i].addEventListener('click', createParagraph);

  // Un message est écrit dans la console en fonction de la valeur de "i"
  if (i < 12) {
    console.log('La valeur de la variable "i" est actuellement inférieure à
    ↪  12 !');
  } else if (i < 25) {
    console.log('La valeur de la variable "i" est actuellement inférieure à
    ↪  25 (mais supérieure ou égale à 12) !');
  } else {
    console.log('La variable "i" vaut au moins 25 maintenant !')
  }
}
```

2.5 Commentaires

```
/*
 Commentaire
 multi lignes ...
*/
var a = 12;

// Commentaire sur une seule ligne
```

3. Retour commenté sur le code JavaScript vu précédemment

```
<body>
  <!--
    Le reste du corps de notre page
```

```
-->
<script src="https://unpkg.com/leaflet@1.6.0/dist/leaflet.js"
 ↪  integrity="sha512-
 ↪  gZwIG9x3wUXg2hdXF6+rVkLF/0Vi9U8D2Ntg4Ga5I5BZpVkVxlJWbSQtXPSiUTtC0TjtGOmxa1AJPuV0CP
 ↪  crossorigin=""></script>
<script type="text/javascript">
  var mymap = L.map('mapid').setView([45.192, 5.768], 14);

  var osmLayer = L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
    attribution: '© OpenStreetMap contributors',
    maxZoom: 19
  });

  mymap.addLayer(osmLayer);

  function onMapClick(e) {
    // On affche les coordonnées dans la console:
    console.log("You clicked the map at " + e.latlng);
    // Les 3 instructions suivantes permettent d'interagir avec le DOM
    // lors de l'exécution de la fonction. On appelle ici les éléments
    // grace à leur 'id'.
    // On change la valeur des éléments du formulaire:
    document.getElementById("inputLong").value = e.latlng.lng;
    document.getElementById("inputLat").value = e.latlng.lat;
    // On active le formulaire (en définissant la valeur de 'disabled' sur
     ↪  false)
    document.getElementById("fieldsreview").disabled = false;
    // On indique où va être laissé l'avis avec un tooltip.
    // C'est un objet de la bibliothèque 'Leaflet':
    var popup = L.popup();
    popup
        .setLatLng(e.latlng)
        .setContent("Your review will appear here")
        .openOn(mymap);
    // On déplace le document jusqu'à l'élément qui a l'id "topfieldset"
    // en 'scrollant' avec un effet de transition.
    // Cela ne se produit que dans la version smartphone quand les éléments
    // sont empilés.
    // Dans la version desktop, l'élément est déjà en haut de la vue
    // de l'utilisateur.
    document.getElementById("topfieldset").scrollIntoView({behavior:
     ↪  "smooth", block: "start", inline: "nearest"});
  }
```

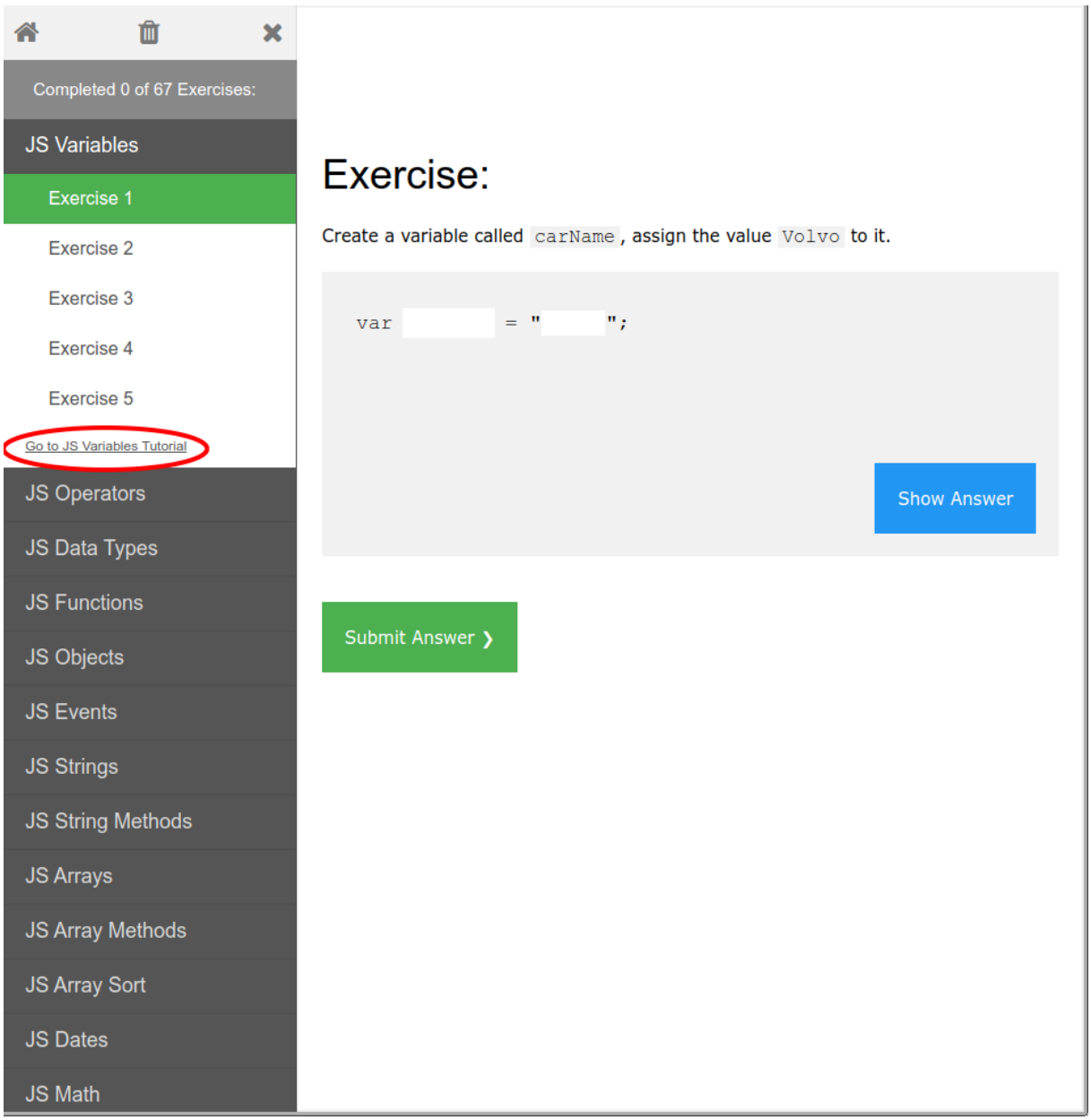
```
    mymap.on('click', onMapClick);
  </script>
</body>
```

4. Mise en pratique

→ Rendez-vous sur la page https://www.w3schools.com/js/exercise_js.asp et commencez les exercices dans l'ordre.

Ces exercices simples mettent l'accent sur votre bonne compréhension du mécanisme de base de l'écriture du code JavaScript. Ils représentent également un bon moyen d'apprendre certaines de ces notions par la pratique.

Soyez attentif aux énoncés et suivez votre instinct ! N'hésitez pas à demander et/ou à vous rendre sur le tutoriel correspondant avant de cliquer sur `Show Answer` :)



5. Amélioration de notre plateforme de dépôt d'avis grâce au JS

→ Essayez d'implémenter chacun des comportements suivants en JavaScript :

- Changement de la couleur du marker en fonction de la note correspondante (par exemple 1: rouge / 2-3-4: orange / 5: vert - voir ici par exemple)
- Centrage de la carte calculé en fonction de l'emprise totale des avis laissés
- Utilisation d'un autre fond de carte par Leaflet

- Ajout d'un widget permettant de changer le fond de carte (et en proposant au moins deux différents - comme ici par exemple)
- Changement du fond de carte en fonction du niveau de zoom (comme ici par exemple)

Nous ferons un point de cours à la fin de la séance.