Javascript ES6

Louis Harang - Dylan Correia
Wizards Technologies

Qui sommes-nous ?

Dylan Correia
Formation :
- DUT Informatique (Orsay)
- Master Ingénierie du Web (ESGI)

24 ans
Email : dcorreia@wedigital.garden

Louis Harang
Formation :
- Master Ingénierie du Web (ESGI)

27 ans
Email : cours@narah.io

Développeurs Full Stack @ Wizards Technologies

Faisons connaissance

Présentez vous !

- Prénom
- Dans quelle boîte vous travaillez et avec quelles technos
- Ce que vous connaissez du javascript
- Ce que vous attendez du cours

Sommaire

*** TP dans le chapitre**

Modalités d’évaluation

50% TP notés + 50% projet de groupe

Organisation des TP

- Chaque TP sera **numéroté**
- Le rendu doit être mis sur **Github** et envoyé par email
- Chaque repository doit comporter un readme avec :
    - Votre nom
    - Votre prénom
    - Votre adresse email
- Un seul repo pour tous les TP avec 1 dossier par TP comme ci-dessous:
        - TP1
        - TP2
        - TP3
        - ...
        - README.MD
- 1 fichier par exercice sauf si mention contraire dans l'énoncé
- Les TP sont des travaux personnels, si il y a la moindre ressemblance dans le code entre plusieurs personnes = 0 pour tout le monde

Projet de groupe

Plus loin dans le cours nous vous présenterons le sujet pour le projet de Javascript.

- Le sujet sera imposé
- Travail en groupe de 1 à 4 personnes (max).
- Le projet devra contenir un README avec le nom + prénom de tous les développeurs ainsi que vos emails.
- Vous devez utiliser **github** obligatoirement
- Plagiat ou triche = 0
- Tous les membres du groupe doivent travailler de manière égale sur le projet (nous regarderons les commits)

Le barème du projet vous sera transmis en même temps que le sujet.

Introduction

Histoire

Origine du javascript

Développé en dix jours en **mai 1995** pour le compte de la **Netscape Communications Corporation** par **Brendan Eich.** Le langage est inspiré de plusieurs langages, notamment **Java** mais en simplifiant la syntaxe.

Ce nouveau langage est appelé **LiveScript.**

Quelques jours avant sa sortie, Netscape change le nom de **LiveScript** pour **JavaScript.**

En **mars 1996,** Netscape met en œuvre le moteur JavaScript dans son navigateur web. Le succès de ce navigateur contribue à l'adoption rapide de JavaScript dans le **développement web orienté client.**

Netscape soumet alors **JavaScript** à **ECMA International** pour standardisation**.**

ECMAScript

ECMA quoi ?! 1/2

Ecma International - *European association for standardizing information and communication systems* - est une organisation de standardisation active dans le domaine de l'informatique.

ECMA développe des standards sur les langages de script et de programmation, les technologies de télécommunications les produits de sécurité et pleins d’autres sujets informatiques.

Les travaux de standardisation débutent en **novembre 1996** et se terminent en **juin 1997**. Ces travaux donnent naissance au standard ECMA-262 qui spécifie le langage **ECMAScript**.

ECMAScript

Le versionning de ECMAScript 2/2

- **ECMAScript 1 :** Juin 1997
- **ECMAScript 2 :** Juin 1998
- **ECMAScript 3 :** Décembre 1999
- **ECMAScript 4 :** Abandonné car trop complexe, les features de la version 4 ont été réintégrées dans la version 6
- **ECMAScript 5 :** Décembre 2009, pleins d'ajouts importants
- **ECMAScript 6 :** Juin 2015, révolution de la syntaxe et des fonctionnalités
- **ECMAScript 7 à ? :** Depuis la version 6, les versions ECMAScript prendront le nom d'ES<Année en cours>. Chaque année une nouvelle version sort avec des nouveautés.

Aujourd'hui nous sommes donc à **ES 2021**

Dans ce cours nous parlons d'ES 6 (ES 2015), ce n'est pas que nous allons nous arrêter à la version 6, mais simplement ES6 est le nom officieux du **Javascript moderne post-ES6**.

Utilisation du Javascript

Où et comment

Où peut on exécuter du javascript ?

Historiquement, dans votre **navigateur**

Aujourd'hui avec Node, il est aussi possible d'exécuter du **javascript dans le terminal**.

Il est aussi possible de lancer des **scripts JS dans MongoDB .**

Et même dans des **applications desktop** (Electron) et **mobile** (Flutter, React Native...) !

Dans notre cours, nous allons nous concentrer sur l'exécution du javascript dans le **navigateur** et dans le **terminal**.

Utilisation du Javascript

Pour commencer, vous pouvez exécuter du javascript dans la console de votre navigateur ou alors créer un page HTML avec une balise <script>

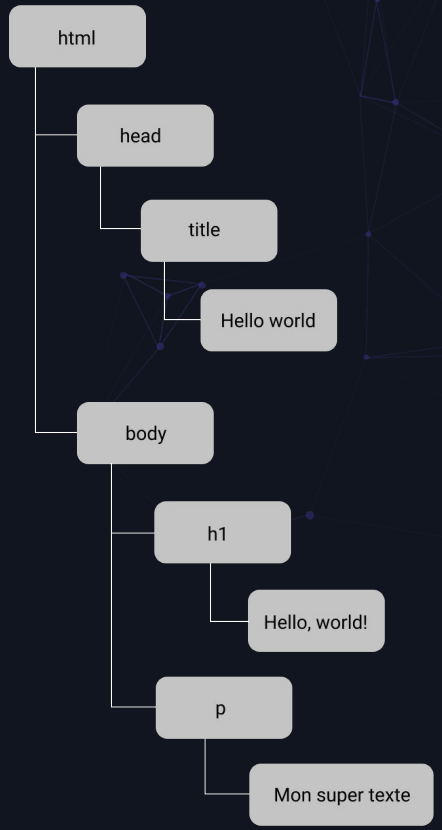
```
<!DOCTYPE html>
<html>
<head>
  <title>My first page</title>
</head>
<body>
</body>
<script type="text/javascript">
  alert('hello world !');
</script>
</html>
```

C’est quoi le DOM ?

L'arborescence d’une page web

Le DOM (Document Object Model) est une représentation d’un document HTML sous forme d’un **arbre d’objets**.

```html
<!DOCTYPE html>
<html lang="fr">
  <head>
    <title></title>
  </head>
  <body>
    <h1> Hello, world !</h1>
    <p>Mon super texte</p>
  </body>
</html>
```



Manipuler le DOM

Browser Object Model (BOM)

Le BOM est le coeur du Javascript dans le navigateur. Le BOM est composé de plusieurs objets qui permettent d'ajouter des fonctionnalités et de l'interaction avec une page web **indépendamment de son contenu**. Attention, même si les BOM sont semblables entre les navigateurs, ils ne sont pas tous forcément disponible. Chaque navigateur en a un implémentation différente.

- **window :** supporté par tous les navigateurs, **window** représentent la fenêtre du navigateur.
- **window.location :** permet de manipuler l'URL de la page
- **window.history :** permet de gérer l'historique de navigation de la page
- **window.document (document) :** document, c'est la représentation de la page web, c'est le DOM
- **window.alert (alert) :** permet d'afficher des modales d'alertes

Manipuler le DOM

Récupérer des infos

Via l'objet **document** on a accès à plusieurs méthode permettant de retrouver des éléments du DOM.

```
/*
 * On récupère l'element avec l'id header
 * Si il y a plusieurs element avec header,
 * la méthode retournera la première occurence
*/
const element = document.getElementById('header');

/*
 * On récupère les elements avec la classe button
 * La méthode retourne un tableau d'element
*/
const elements = document.getElementsByClassName('button');

/*
 * On récupère les elements avec le tag li
 * La méthode retourne un tableau d'element
*/
const elements = document.getElementsByTagName('li');
```

Manipuler le DOM

Récupérer des infos

Via l'objet **document** on a accès à plusieurs méthode permettant de retrouver des éléments du DOM.

```
/*
 * On peut récuperer des informations sur un element du DOM
 * Ici on récupère la valeur de l'attribut src
*/
const element = document.getElementById("avatar").getAttribute("src");

/*
 * On peut aussi récuprer d'autres champs
 *
*/
const element = document.getElementById("avatar").value("src");

/*
 * Pour modifier le texte entre la balise ouvrante et fermante d'un element du DOM
 * on utilise innerHTML
*/
const element = document.getElementById("bio").innerHTML = "Ma super bio";

/*
 * On peut aussi modifier le CSS d'un element
 *
 */
const element = document.getElementById("title").style.color = "red";
```

Manipuler le DOM

Créer des éléments HTML

On peut créer des éléments dans le DOM directement en javascript, pour cela il faut tout d'abord créer un élément puis l'attacher à un autre élément du DOM.

```
// On créer un élement p avec la méthode createElement
const myElement = document.createElement("p");
// On personnalise notre élément
myElement.innerHTML = "Hello world !";
myElement.style.color = "red";

// Il faut attacher notre nouvel élément à un élément existant dans le DOM
// On va récupérer le body et lui attacher notre élément
document.getElementsByTagName('body')[0].appendChild(myElement);
```

Manipuler le DOM

Créer des éléments HTML

On peut bien aussi supprimer un élément du DOM ou encore remplacer un élément par un autre

```
const element = document.getElementById('title');

// On supprime avec removeChild
document.removeChild(element);

// On peut aussi remplacer un élément avec replaceChild
document.replaceChild(newElement, oldElement);
```

Manipuler le DOM

Créer des éléments HTML

On peut aussi utiliser **document.write** pour écrire directement dans le stream HTML. **Attention** cette méthode remplace tout le HTML de la page.

```
document.write("Hello world !");
```

Méthodes à savoir

Pour vous aider à développer et déboguer vos applications client javascript, vous pouvez utiliser les méthodes ci-dessous pour dump vos variables.

```
// console.log permet d'afficher des variables dans la console du navigateur
console.log("foo"); // affiche dans la console : "bar"

// alert permet d'afficher une pop up dans votre navigateur
alert('hello world');
```

TP 1

Manipulation du DOM

Pour chaque exercice, faites l'exo dans un fichier HTML avec une balise script

1. Coder un script JS qui change la couleur de fond du body
2. Créer une page HTML avec 3 balises "p" contenant du texte, coder un script permettant de mettre en gras le 1er élément, en italique le 2ème et en uppercase le 3ème
3. Créer une page HTML avec 3 images, coder un script qui remplace l'attribut src de la 1ère et 2ème image par celui de la 3ème
4. Coder un script qui permet d'ajouter un élément div avec un fond violet, une largeur de 100% et une hauteur de 300px. Créer un autre élément "h1" avec une couleur de texte blanche, en majuscule. Attacher l'élément H1 à l' élément div.
5. Récupérer une page HTML sur internet, puis rajouter un script permettant de supprimer tous les éléments "p"
6. Créer une page HTML avec une liste "Toto" "Tata" "Titi" "Tutu". Créer un nouvel élément li et remplacer le 2ème élément de la liste par le nouvel élément.
7. Récupérer une page HTML sur internet, avec **console.log** afficher la largeur et la couleur de fond de la balise body

Les bases de Javascript

Les types

Déclarer des variables

Même si le Javascript est faiblement typé, il reste typé, il existe plusieurs types directement accessible :

Types primitifs :

- Undefined
- Boolean
- String
- Number

Types complexes :

- Object (Symbol, Array et null sont aussi des objets)
- Function

```
// Non rcommandé, ne fonctionne pas en strict mode
name = 'Jack';

var price = 100;

let isOpen = false;

const name = 'Javascript'
```

Let, const (& var)

Déclarer des variables

La version ES 6 de Javascript a introduit deux nouveaux mot-clés pour déclarer des variables : **const** & **let.** Pour déclarer des variables en JS rien de plus simple.

```
// Non rcommandé, ne fonctionne pas en strict mode
name = 'Jack';

var price = 100;

let isOpen = false;

const name = 'Javascript'
```

Ces 3 mot-clés sont différents à 3 niveaux :

- La portée
- La possibilité d'assigner une nouvelle valeur
- L'accès à la variable avant sa déclaration

Let, const (& var)

La portée des variables

La portée (qu'on appelle souvent **scope**) permet d'identifier où et si on peut utiliser une variable. Les variables peuvent exister à l'intérieur d'un bloc, à l'intérieur d'une fonction ou à l'extérieur d'une fonction et d'un bloc.

Qu'est-ce qu'un bloc ? Un bloc est une section du code que nous définissons à l'aide d'une paire d'accolades { }

```
{
  let message = "Hello world !";
}
```

Même chose pour les fonctions, selon le mot-clé, la portée est différente.

Toute déclaration en dehors d'un bloc ou d'une fonction sera une variable considérée comme **globale.**

```
function test() {
  var message = "hello";
}
```

Let, const (& var)

La portée des variables 2

La portée dans un bloc :

```
{
    let name  = 'Toto';
    const zipCode = 75011;
    var age = 18;
}

console.log(name); // Uncaught ReferenceError: name is not defined
console.log(zipCode);  // Uncaught ReferenceError: zipCode is not defined
console.log(age);  // 25
```

On voit bien ici que la variable déclarée avec **var** est accessible à l'extérieur du bloc, cela rend donc possible l'écrasement de cette variable et donc l'apparition de bogue.

A l'intérieur d'un bloc utilisez **const** & **let** uniquement qui possède une portée de bloc.

Let, const (& var)

La portée des variables 3

La portée dans une fonction :

```
function test(){
    let name  = 'Toto';
    const zipCode = 75011;
    var age = 18;
}
test();

console.log(name); // Uncaught ReferenceError: name is not defined
console.log(zipCode);  // Uncaught ReferenceError: zipCode is not defined
console.log(age);  // Uncaught ReferenceError: age is not defined
```

Aucune variable n’est accessible à l’extérieur, var à une portée au niveau de la fonction.

Let, const (& var)

La portée des variables 4

A retenir :

- **var** : niveau de portée fonctionnelle
- **let :** niveau de portée de bloc
- **const :** niveau de portée de bloc

Let, const (& var)

Assigner une nouvelle valeur

```
let name = "Toto";
const zipCode = 75011;
var age = 18;

// On réassigne des valeurs
name = "Bob"; // name value = 'Bob"
zipCode = 75001; // Uncaught TypeError: Assignment to constant variable.
age = 78; // age value = 78
```

Sur l'exemple ci-dessus, on peut voit qu'on peut tout à fait modifier les valeurs d'une variable déclarée avec **let** ou **var.**

Pour const, comme son nom le laissait présager, il n'est pas possible de modifier la valeur... enfin presque

Let, const (& var)

Assigner une nouvelle valeur, const

On peut modifier la valeur d'une variable déclarée avec const dans le cas où celle-ci est déclarée sous forme d'objet. Il est alors possible de modifier les propriétés de l'objet sans problème. Cependant il n'est pas possible d'attribuer un nouvel objet.

```
const links = {
    'github': 'https://github.com/javascript'
}

links.github = 'https://github.com/nodejs'; // Ok

links = {}; // Uncaught TypeError: Assignment to constant variable.
```

Let, const (& var)

Accéder à la variable avant sa déclaration

Vous ne devriez jamais essayer d'accéder à une variable sans la déclarer. Si le cas se présente, voyons comment les variables peuvent se comporter.

En mode non strict, on peut déclarer la variable avec var après lui avoir assignée une valeur.

Pourquoi ça marche alors que la variable est déclarée après ?

Cela ça s'appelle le **hoisting** (qu'on peut traduire par "hissé"). C'est le comportement de base de javascript qui va "remonter" toutes les déclarations en haut de la portée.

```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>
x = 5; // On assigne 5 à x

elem = document.getElementById("demo");
elem.innerHTML = x; // On affiche la variable

var x;
</script>
</body>
</html>
```

Let, const (& var)

Accéder à la variable avant sa déclaration

Si on tente de faire la même chose avec let, on aura une erreur :

Uncaught ReferenceError: can't access lexical declaration 'x' before initialization

Les variables déclarées avec **let** & **const** sont remontée en haut du bloc mais ne sont pas initialisées : le bloc est conscient de la variable mais ne peut pas l'utiliser car elle n'est pas déclarée.

Avec **const** ce n'est pas possible d'accéder à la variable avant sa déclaration car on ne peut pas modifier une variable déclarée avec const.

```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>
x = 5; // On assigne 5 à x

elem = document.getElementById("demo");
elem.innerHTML = x; // On affiche la variable

let x;
</script>
</body>
</html>
```

Let, const (& var)

Conclusion

Utilisez **let** si vous allez modifier votre variable dans le bloc.

Utilisez **const** si n'allez pas modifier votre variable dans le bloc.

N'utilisez pas var à cause des effets de bord qu'on vient d'évoquer.

Les fonctions

Javascript permet de créer des fonctions comme tous les langages de programmation, la syntaxe est basique.

Vous pouvez bien entendu rajouter des variables scopés (sauf var)

```
function getDate() {
  return new Date();
}

console.log(getDate());
```

```
function getDate() {
  const date = new Date()
  return date;
}
```

Les events

De l'interactivité avec vos pages web

Les events sont les différentes interactions, actions ou états d'une page HTML.

Il en existe plusieurs, parmis les plus connus :

- onclick
- onchange
- onmouseover
- onmouseout
- onkeydown
- onload

Pour déclencher une fonction lors d'un événement sur un élément du DOM :

```
<button onclick="console.log('click !')">click me</button>
```

Les events

Usage de this avec les éléments du DOM

Javascript possède un mot-clé **this** qui est un peu différent des autres langages de programmation.

Avec **this**, on peut passer la référence de l'élément à la méthode exécutée lors d'un événement.

Dans l'exemple ci-contre, on passe la référence de l'élément cliqué en paramètre de la fonction **changeColor** ce qui permet d'interagir directement avec l'élément du DOM.

```
<!DOCTYPE html>
<html>
<body>
<button onclick="changeColor(this)">click me</button>
<script>
  function changeColor(el) {
    el.style.color = "red";
  }
</script>
</body>
</html>
```

Les events

Usage de this avec les éléments du DOM

Il est aussi possible de récupérer les informations d'un événement en passant en paramètre "event" dans la méthode au niveau de la déclaration de l'event.

```
<!DOCTYPE html>
<html>
<body onkeydown="getKey(event)">
<p> you pressed the key <span id="value"></span></p>
<script>
  function getKey(event) {
    document.getElementById('value').innerHTML = event.key;
  }
</script>
</body>
</html>
```

TP 2

Events

1. Créer une page HTML avec une image. Lorsque vous cliquez sur cette image elle pivote dans le sens des aiguilles d'une montre de 90°.
2. Créer une page HTML avec une liste contenant des noms de ville. Au clic d'un élément de la liste, remonter l' élément en haut de la liste.
3. Créer une page HTML avec 2 images. Lorsque vous cliquez sur une des deux images, la largeur de l'image cliquée augmente de 20% de sa largeur actuelle et la largeur de l'autre image diminue de 20% de sa largeur actuelle. Attention : la largeur ne peut pas être négative.
4. Créer une page HTML contenant une div de 50 px sur 50px avec un fond rose. Avec un event, faite en sorte de pouvoir déplacer la div avec les touches fléchées de votre clavier.
5. Créer une page HTML avec plusieurs images, au survol d'une image faites un zoom x3 sur cette image.
6. Récupérer une page HTML sur internet, au survol de n'importe quel élément, changer la couleur du texte en rouge
7. **Bonus :** Récupérer une page HTML sur internet, au clic sur un élément de la page, supprimer l'élément du DOM
8. **Bonus 2 :** Récupérer une page HTML sur internet. Au clic dans le body de la page, mélanger tous les éléments du DOM (à l'intérieur de body)

Itérations & Conditions

Conditions

A l'instar d'autres langages, javascript donne la possibilité d'écrire des conditions logiques.

```
if (age < 18) {
   return 'Not allowed to drive a car';
}


if (age > 18) {
   return 'Adult';
}
else if (age > 10) {
  return 'Teenage';
}
```

```
switch(colorHex) {
case "red":
 return "#af4154";
case "blue":
 return "#0f0ade";
default:
    console.log('color not found');
    break;
}
```

Itérations & Conditions

Itérations

On peut utiliser **for** pour faire des itérations logiques en Javascript. On utilisera bien sûr **let** et pas **var** pour déclarer notre itérateur **i.** Comme vu dans les slides précédentes, la portée de **var** est fonctionnelle et donc si on déclare i en **var** on pourra y accéder en dehors de notre itération.

```
let str = '';

for (let i = 0; i < 9; i++) {
  str = str + i;
}

console.log(str);
// valeur = "012345678"
```

Itérations & Conditions

Itérations

Vous pouvez aussi utiliser **while** et **do while** pour faire des itérations

```
while (true) {
  // Instruction
}

do {
  // Instruction
} while (i < 5)
```

TP 3

Itérations & Conditions

1. Créer une page HTML avec 3 div carrés ayant un fond noir. Au clic sur une div, elle prend la couleur rouge. Si vous recliquez dessus, elle redevient noir.
2. Créer une page HTML vide. Grâce à une boucle for, généré aléatoirement entre 5 et 15 div de 50px de haut et une largeur de 100% avec une couleur aléatoire.
3. Copier l'exercice précédent. Au clic d'une des div, changer la couleur de fond en rouge. Lorsque une div rouge est de nouveau cliquée, elle récupère sa couleur d'origine
4. Créer une page HTML, coder un script qui permet d'enregistrer les touches tapées par l'utilisateur sous forme de liste. Chaque fois qu'une touche est pressée sur le clavier, le nom de la touche sera ajouter une liste ul. La liste ne doit pas faire plus de 20 entrées, si jamais la liste est plus longue vous pouvez enlever les éléments les plus anciens.
5. Créer une page HTML qui affiche la taille de la fenêtre dynamiquement. Lorsque vous redimensionner la page, la largeur et la hauteur en pixel sont affichés.