# Javascript ES 6 > Introduction

## Un peu d'histoire

En TB, vous avez fait du Javascript de base, celui de la version 1 sortie en 1997, aux débuts du web, et qui connut peu d'améliorations.

Retour vers le futur : la version dite ECMAScript 6, sortie en 2015, a apporté des nouvelles fonctionnalités très intéressantes qui ont posé les bases des frameworks actuels (Angular, React, VueJS etc.).

Désormais, les évolutions de Javascript sont déployées à un rythme annuel : ES 2019, ES 2020...

Pour bien maîtriser Javascript, c'est-à-dire ne pas trop galérer sur les messages d'erreur que produira votre code, une bonne compréhension et appropriation des concepts est nécessaire, d'autant que certains de ces concepts se montrent assez différents par rapport aux autres langages.

### L'objet

Dans Javascript, tout est objet : de votre fenêtre navigateur (objet natif) jusqu'aux objets utilisateurs (ceux que le développeur écrit), en passant par les tableaux.

Et qui dit objet dit propriétés et méthodes de ces objets. Nous y reviendrons en détail dans un cours dédié.

### Héritage par prototype

JavaScript n'utilise qu'une seule structure : les objets. Chaque objet possède une propriété privée qui contient un lien vers un autre objet appelé le prototype. Ce prototype possède également son prototype et ainsi de suite, jusqu'à ce qu'un objet ait *null* comme prototype. Par définition, *null* ne possède pas de prototype et est ainsi le dernier maillon de la chaîne de prototype.

> En savoir plus

## Hoisting

Le moteur d'interprétation de Javascript procède à une première lecture du code avant son exécution, et à ce stade remonte - hisse - les déclarations de variables et de fonctions au début de script : c'est le *hoisting*, OU hissage en français.

#### Exemple

Quand vous écrivez ce code :

```
var sPrenom = "Dave";

console.log("Bonjour "+sPrenom);

var sNom = "Loper";

bonjour(sNom);

function bonjour(sNom)
{
   console.log("Ca farte "+sNom+" ?");
}

var iAge = prompt("Quel est ton âge ?");
```

Javascript voit ceci :

```
/* Déclarations de variable et de fonctions remontées
* en début de script par le moteur d'interpréation
*/
var sPrenom;
var sNom;
var iAge;

function bonjour(sNom)
{
   console.log("Ca farte "+sNom+" ?");
}

/* Exécution du code */
console.log("Bonjour "+sPrenom);

bonjour(sNom);

iAge = prompt("Quel est ton âge ?");
```

**Remarquez bien que les déclarations de variables sont remontées SANS affectation de valeurs, c'est-à-dire uniquement la partie gauche avant le signe égal.**

> En savoir plus

## Mode strict

Le mode strict en Javascript (apparue avec ES 5 en 2009) offre à Javascript une écriture de code beaucoup moins permissive et modifie l'interprétation. L'avantage principal est de mieux traquer les erreurs et donc d'avoir un code op^timisé pour une interprétation (exécution) plsu rapide dans les navigateurs.

> MDN : Le mode strict

## Interprétation

Javascript est un langage interprété. Le code est interprété, c'est-à-dire rendu compréhensible par un ordinateur, par un "moteur", lequel est situé dans le navigateur. Il existe plusieurs moteurs, par exemple *V8* pour Chrome et SpiderMonkey sur Firefox, ce qui peut expliquer parfois les différences d'une même page web d'un navigateur à l'autre.

> https://blog.octo.com/dans-les-entrailles-de-javascript-partie-1

## L'écosystème Javascript

Vous avez peut-être croisé les noms CoffeeScript, TypeScript, ou encore NodeJS dans des offres d'emploi. De quoi s'agit-t-il ?

- TypeScript et exemples de code, site officiel
- CoffeeScript, site officiel
- NodeJS est un environnement d'exécution (et non pas un framework) du langage Javascript. NodeJS est notamment connu pour embarquer un serveur HTTP, ce qui offre la possibilité d'exécuter du Javascript côté serveur. Sortie en 2009, NodeJS a été largement adopté pour des projets d'envergure dans d'importantes entreprises etc.