



# **JAVASCRIPT**

Durée Totale du Module : 21H

Jean-François Pech

Relu, validé & visé par :

Jérôme CHRETIENNE
Sophie POULAKOS
Mathieu PARIS

Date création :

03/03/2023

Date révision :

10/03/2023









## Point sur var let ou const

En JavaScript de base pour déclarer une variable on utilise le mot « var », mais avec la version ES6 on a eu deux autres manières de déclarer des variables.

(Var cela permet de voir si un tutoriel commence à dater)

En fait var peut poser problèmes avec la notion de scope que l'on a vu précédemment

Rappel scope : Jusqu'où notre variable va être disponible

Avec let et const on gère le scope en fonction des **blocs** dans lesquels on se situe. **(Scope de bloc)** 

Au contraire var a un scope uniquement basé sur les fonctions

C'est plus de contraintes mais ça permet de garder une logique et un code plus propre

## Exercice: Quizz Var

Je suis stagiaire dans votre entreprise (sous traitant Nasa) et je vous envoi ce code en « revue de code »

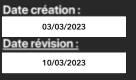
Que me répondez-vous ?

```
var voiture = "Renault";
console.log(voiture);
var voiture = "BMW";
console.log(voiture);
```

(Je vous jure cela fonctionne)













Cela fonctionne mais ce n'est pas correct dans la logique. JS est peut être un langage assez permissif, mais cela peut engendrer des problèmes.

Exercice: Quizz let

```
//!Quizz : ca bug
console.log(bolide);
let bolide = 'Jaguar'
```

```
    ♥ Uncaught ReferenceError: Cannot access 'bolide' before app.js:12 initialization at app.js:12:13 (anonymous) @ app.js:12
```

Exercice: Quizz function-var

```
function choixVoiture(){
    var uneVoiture = "Harley Davidson"
}
choixVoiture();
console.log(uneVoiture);
```

10/03/2023









## Solution function var

Pour le quizz précédent

Erreur : La var voiture est déclaré au sein de ma fonction et ne peut pas être utilisée en dehors.

```
var uneVoiture = "Harley Davidson"
function choixVoiture(){
}
choixVoiture();
console.log(uneVoiture);
```

lci ça fonctionne car var est déclaré au-dessus.













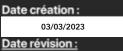
Exercice: Quizz if-var

```
var car = "Nissan";
if(car=="Nissan"){
   var vitesse = 800;
}
console.log(vitesse);
```

Ca fonctionne, pour vous est-ce normal?

800 <u>app.js:38</u>





10/03/2023









Var est basé sur un scope de <u>fonction uniquement</u>

If il ne le considère pas comme une fonction du coup ça fonctionne.

Si l'on y prête pas attention ce genre de soucis peut engendrer de plus gros problèmes quand le code de vos applications deviendra plus complexe

```
var theCar = "Nissan";
if(theCar=="Nissan"){
    let speed = 800;
}
console.log(speed);
```

⊗ ► Uncaught ReferenceError: speed is not defined at <u>app.js:46:13</u> <u>app.js:46</u>

Le let fonctionne comme le var sauf que : il ne rend les variables disponible que à un **bloc { }** Boucle, fonction, condition peu importe.

Reprenons les Bug de tout à l'heure mais en utilisant le mot clé let

```
console.log(moto);
let moto = 'Yamaha'
```

⊗ ► Uncaught ReferenceError: Cannot access 'moto' before initialization at app.js:48:13

<u>app.js:48</u>







On obtient dès lors des erreurs certes mais beaucoup plus précises.

Ici avec const : erreur voiture2 a déjà été déclarée

Exercice Quizz: if-let

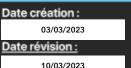
```
let superCar = 'BMW';
const superModel = 'Sport';

if(superCar == 'BMW') {
    const superVitesse = 900;
    let superCar = "Citroen";
    console.log(superCar);
}

console.log(superCar);
```

Le Stagiaire vous demande : « J'ai tout bon ? » Que peut-on lui répondre ???













## Solution if-let

Ici c'est Ok II faut bien comprendre que JS va créer 2 variable Différentes (le voiture du haut n'est pas le même qu'en bas) Importance du scope

## Const

```
const superConstante = 'YES';
superConstante = 12;
```

Une constante c'est une variable dont on sait qu'on ne va pas lui ré assigner une valeur, cela permet d'appliquer plus de restrictions dans notre code afin de ne pas créer plusieurs variable du même nom (des doublons) et qui servent à la même chose dans le programme.

Le code est mieux structuré donc plus lisible / compréhensible donc plus facile à maintenir.

Une légère exception ?

On peut quand même « appliquer une modification » à une constante si c'est un objet, on peut modifier une **propriété** de cet objet (ci dessous on ne peut pas modifier la structure (l'objet en lui même), mais seulement une de ses propriétés.

```
const MyTracklist = {
    track1:'lofteurs up and down',
    track2:'David Hallyday',
    track3:'Crazy Frog'
}
console.log(MyTracklist);
MyTracklist.track1 = 'félicien'
```

```
app.js:92
{track1: 'lofteurs up and down', track2: 'David Hallyday', track3:
'Crazy Frog'}

app.js:95

{track1: 'félicien', track2: 'David Hallyday', track3: 'Crazy Frog'}
```

Au final en JS moderne (ES6 et +) on va privilégier l'utilisation de **let** et **const** pour créer des variables.

