**HOISTING (HISSAGE)**

Javascript fait 2 passes sur le script pour effectuer une opération de hoisting. Le **hoisting** consiste à hisser les **déclarations** de fonctions et les **déclarations** de variables.

Exemple_1:

```
addition(5,7);
function addition(a , b) { console.log(a+b) }
```

=> ça fonctionne !! la fonction addition est déclarée.

maintenant si on fait

```
addition(5,7);
var addition = function (a , b) { console.log(a+b) }
```

=> ça ne fonctionne pas !! la fonction addition n'est pas déclarée.

Remarque: javascript ne déplace pas physiquement le code, les déclarations de variables et de fonctions sont mises en mémoire pendant la phase de *compilation*, mais restent exactement là où vous les avez tapées dans votre code.

Par contre :

```
addition(5,7);
var addition = function(a , b) { console.log( a + b ) }
// fonction anonyme
```

Ce n'est pas une **déclaration** de fonction. Ici la fonction est appelée une **fonction anonyme**.

TYPES PRIMITIFS FONCTIONNENT PAR VALEUR

var x = 5 ;

Memoire

var x = 5
var y = x

var x = 8

MAINTENANT SOIT UN OBJET

var x = {  name : "John" }

X espace memoire

OBJET à COTE => aura espace memoire

X a une certaine adresse
et objet à une autre adresse

Dans x on a un pointeur ( les objets marchent par reference )

maintenant

var y = x

y => a sa memoire

y = x pointeur vers le même objet que x

Rajoutons

y.name = "autre nom"

les l'objet référence par les 2 référence est modifié

y = { name : "toto" }

c'est un nouvel objet (nouvelle assignation)

https://fr.javascript.info/object-basics