Les variables

Il y a plusieurs mots-clefs disponibles lors de la création de variables en Javascript. Elle modifient ce qu'on appelle le "scope" ou la "portée" des variables.
Cela signifie que les variables créees sont seulement accessibles depuis certaines places et qu'en dehors de celle-ci si on y fait référence une erreur "ReferenceError" apparaîtra.
Une bonne utilisation du scope permet également de réutiliser les mêmes noms de variables à différents endroits sans faire de "collisions" c'est à dire sans réassigner leurs valeurs alors que ces variables auraient dû être indépendantes.

Il existe 4 types variables différentes:

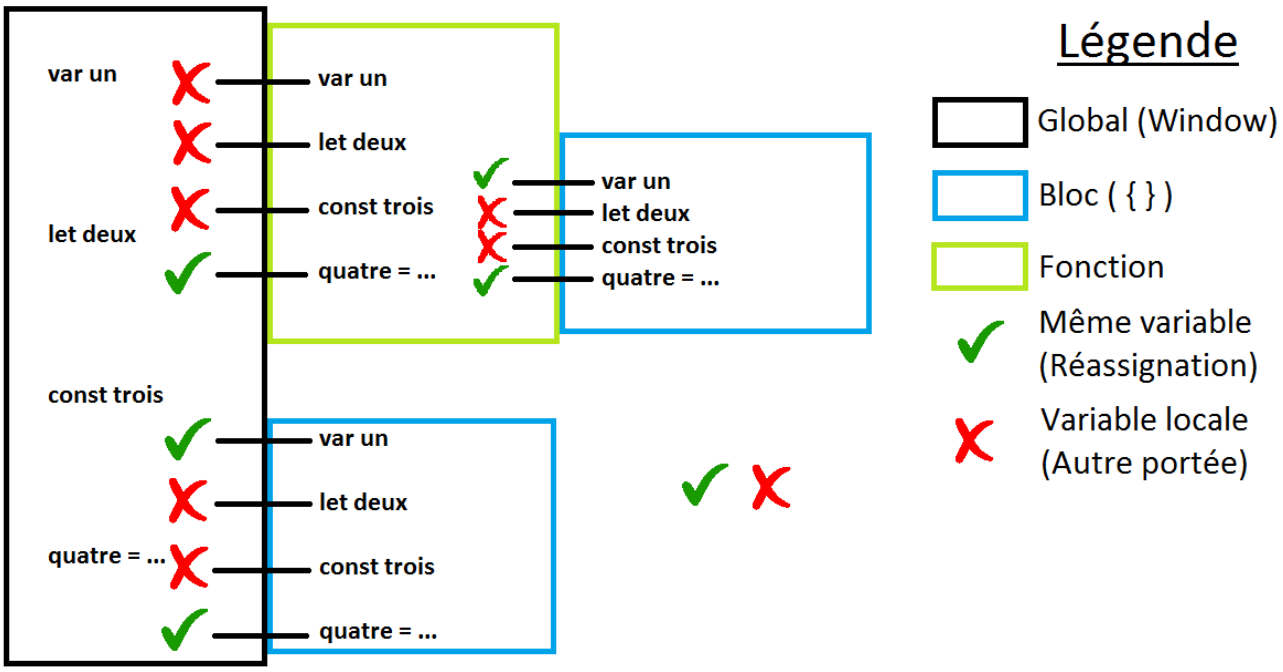
- var maVariable,
  "var" limite la portée de la variable à la fonction dans laquelle elle a été déclarée. Elle est donc inaccessible en dehors de cette fonction. Si déclarée en dehors d'une fonction est globale.

- let maVariable,
  "let" limite la portée au bloc dans lequel la variable à été déclarée. Un bloc est une paire de brackets ou accolades ("{}"). Si déclarée en dehors d'un bloc est globale.
  A noter que les constructions comme un "for" qui possèdent une partie juste avant les brackets sont égalements considérés comme faisant partie d'un bloc => for (let counter...) {}, la variable counter ne sera pas accessible après le "for" bien qu'elle ne fasse pas partie d'un bloc.
  (Les mots-clefs de type de variables ne sont pas valides à l'intérieur des parenthèses des "if", "while" et certaines autres constructions)

- const maVariable,
  "const" à le même comportement que "let" à l'exception qu'il ne permet pas de réassigner à cette variable ultérieurement. Cela signifie également qu'on est obligé de faire une initialisation (pas de déclaration sans donner de valeur parce que non-réassignable !).
  Attention à ne pas tomber dans le piège: on ne peut pas réassigner une valeur à la variable mais on peut modifier la valeur de celle-ci ! Par exemple avec une variable "const" qui contient un array => const monArray = [1, 3, 5], on peut utiliser la méthode push pour rajouter un élément ce qui donnera => monArray.push(7) // [1, 3, 5, 7]

- maVariable,
  Lorsque aucun mot mot-clef n'est utilisé, la variable est "globale" on peut donc y accéder depuis n'importe où. On ne peut pas déclarer une variable de façon globale sans lui donner une valeur (car avec "maVariable" Javascript

comprendrait qu'on veut récupérer la valeur de la variable et on aurait donc une erreur "ReferenceError").
Les autres types de variables agissent de la même façon lorsque déclarés en dehors de leur milieu => var maVariable en dehors de toute fonction est une variable globale.
Certaines différences sont également que sans mot-clef, une initialisation ou une assignation sont des expressions donc elles renvoient une valeur qui est dans tous les cas ce qui se trouve à droite du signe "=". Cela permet de les utiliser, bien que non recommandé, dans les endroits où une expression est obligatoire comme dans les parenthèses d'un "if" ou d'un "while" => if (maVar = true) {...} s'exécutera car le résultat de l'initialisation est true.



Pour déclarer ou initialiser plusieurs valeurs en même temps on peut simplement les séparer par des virgules après le mot-clef => var one, two = 2, three = 3 nous donnera bien 3 variables différentes.

Un dernier point important est qu'on peut placer des blocs ("brackets" / accolades) où l'on veut ! Ils ne sont pas limités aux endroits où la syntaxe le demande (les if, for, while, etc.), il suffit juste d'entourer le code d'accolades.

=> {

```
let variable1 = 5;
const variable2 = 3;
var variable3 = 8;
variable4 = 12;
```

}

Les variables **variable1** et **variable2** n'existent plus en dehors du bloc mais les variables **variable3** et **variable4** existent bien (car **var** utilisé à l'extérieur d'une fonction se comporte comme une variable globale).