

# 02 Les variables

## La déclaration de variables

Les variables contiennent des données qui peuvent être modifiées lors de l'exécution d'un programme.

Le nom de votre variable doit commencer par une lettre (alphabet ASCII) ou le signe `_` et se composer de lettres, de chiffres et des caractères `_` et `$` (à l'exclusion du blanc).

JavaScript est sensible à la casse.

Les variables se déclarent aujourd'hui principalement avec `let` (pour une valeur amenée à changer) ou `const` (pour une valeur qui ne doit pas être modifiée). L'instruction `var` existe encore, mais elle est considérée comme obsolète, car elle crée souvent des comportements inattendus liés à sa portée et au mécanisme de *hoisting*.

### Portée des variables :

- `let` et `const` ont une **portée de bloc** : une variable n'existe que dans les `{ }` où elle est déclarée.
- `var` a une **portée de fonction** et ignore les blocs simples.

### Exemple (let) :

```
// Variable modifiable
let num = 1;

// Variable contenant du texte
let prenom = "Jean";

alert(num);
alert(prenom);
```



Testez cet exemple.

### Exemple de portée (let) :

```
let x = 10;

if (true) {
    let x = 20; // cette variable n'existe que dans ce bloc
    alert("Dans le bloc : " + x);
}

alert("Hors du bloc : " + x); // on retrouve la valeur 10
```

Les deux variables `x` sont distinctes : la version "20" est limitée au bloc `{ }`.

Il existe une façon de déclarer des valeurs qui ne doivent pas changer : les **constantes**. Elles se déclarent avec le mot clé `const`. Une constante doit recevoir une valeur immédiatement et ne peut plus être réassignée.

### Exemple (const) :

```
const age = 37;
age = 27; // Erreur : impossible de modifier une constante

alert(age);
```



Testez cet exemple et vérifiez le message d'erreur dans la console.

### Exemple de portée (var) – à éviter :

```
var a = 5;

if (true) {
    var a = 99; // redéclare et écrase la variable du dessus !
}

alert(a); // affiche 99 (comportement surprenant)
```

Avec `var`, la variable `a` n'est pas limitée au bloc `{ }` : c'est pourquoi son usage est fortement déconseillé.

## Les types de données

JavaScript utilise 5 types de données :

- Nombre : Tout nombre entier ou avec virgule tel que `22` ou `3.1416`. Tout entier en octal ou hexadécimal tel que `0387`, `0xFFA8`.
- Chaîne : toute suite de caractères comprise entre guillemets ou côtes telle que "suite de caractères" ou 'suite de caractères'
- Booléen : les mots `true` pour vrai et `false` pour faux
- Objet : toute utilisation de variable par référence vers tout objet natif JavaScript (Array, Date, String ...) ou tout objet du DOM.
- `null` : mot réservé qui ne représente pas de valeur. La valeur `null` est affectée volontairement à une variable.
- `undefined` : mot réservé qui est renvoyé par une variable référencée qui n'a pas encore été définie (la variable existe mais n'a reçu aucune affectation de valeur).

Exemples :

```
var myVar;           // le type de la variable myVar est undefined
myVar = 324;         // type number
myVar = "Bonjour";   // type string
myVar = true;        // type boolean
myVar = new Array(); // type object
```

### Connaître le type d'une donnée

Une variable peut changer de type après une affectation. On peut vérifier le type en cours d'une variable avec l'instruction `typeof` :

```
console.log(typeof 42);          // Affiche "number"
console.log(typeof 'blubber');    // Affiche "string"
console.log(typeof true);         // Affiche "boolean"

var myVar;
console.log(typeof myVar);       // Affiche 'undefined'
```



Testez cet exemple.