Les variables en JavaScript

> Les **variables** servent à **stocker des données** réutilisables dans un programme.
>
> Elles sont essentielles pour **manipuler**, **afficher** ou **transmettre** de l'information en JavaScript.

1. Déclaration de variables

> La **déclaration** consiste à **définir une variable** et à indiquer à JavaScript que vous allez l'utiliser.
>
> On utilise principalement **trois** mots-clés pour déclarer des variables : `var`, `let`, et `const`.

Exemple :

```javascript
let nom = "Jean"; // Déclare une variable mutable
const age = 25;   // Déclare une constante immuable
var ville = "Paris"; // Ancien mot-clé, rarement utilisé aujourd'hui
```

1.1. Différences entre *`var`*, *`let`* et *`const`*

1.1.1. `var`

- **Ancien** mot-clé utilisé pour la déclaration.
- Portée **globale** ou limitée à la fonction où elle est déclarée.
- Sujet à des erreurs comme le `hoisting`.

```javascript
var score = 10;
console.log(score); // Affiche 10
```

1.1.2. `let`

- Introduit avec ES6 (JavaScript moderne).
- Portée **bloc**, ce qui signifie qu'elle est accessible uniquement dans l'étendue où elle a été déclarée.
- Préférée pour les variables qui changent.

```javascript
if (true) {
  let x = 5;
}
console.log(x); // Erreur, x n'est pas défini en dehors du bloc
```

1.1.3. `const`

- Déclaration pour des valeurs constants ou immuables.
- Nécessite une initialisation dès la déclaration.
- Egalement de portée bloc.

```javascript
const pi = 3.14;
pi = 3.15; // Erreur : on ne peut pas réassigner une constante
```

2. Initialisation des variables

> Une initialisation consiste à attribuer une valeur à une variable lors de sa déclaration.

Exemple :

```javascript
let couleur = "bleu"; // la variable couleur est initialisée avec "bleu"
let a = 2, b = 3, c = 4; // plusieurs variables initialisées en une seule ligne
```

2.1. Valeurs par défaut

- Une variable qui est déclarée mais sans valeur explicite porte la valeur spéciale `undefined` .

```javascript
let test;
console.log(test); // undefined
```

3. Types de valeurs stockées

> Une variable peut contenir différents types de valeurs.

3.1. Types primitifs

- String (chaîne de caractères) : `let nom = "Alice";`
- Number (nombre) : `const annee = 2023;`
- Boolean (booléen) : `var estActif = true;`
- Undefined : `var x;`
- Null : `var y = null;`

3.2. Type complexe : Objets

- Une variable peut aussi contenir un objet, un tableau ou une fonction.

```javascript
const personne = { nom: "Jean", age: 28 };
const liste = [1, 2, 3, 4];
```

4. Règles de nommage des variables

> Les noms de variables (ou identifier) doivent respecter des règles strictes :

- Ils ne peuvent pas commencer par un chiffre.
- Aucun espace ou caractère spécial (sauf `_` ou `$`).
- Ils respectent la casse (sensible aux majuscules/minuscules).

Exemple :

```javascript
let 1erTest = 10; // Erreur
const nomUtilisateur = "Alice"; // Correct
```

4.1. Bonnes pratiques

- Utiliser des noms clairs et compréhensibles.
- Préférer camelCase pour les noms de variables.

```
const temperatureDuMatin = 15; // Bon exemple de nom clair
```

5. Réaffectation de variables

`var` et `let` permettent de réassigner une nouvelle valeur à une variable, contrairement à `const`.

```javascript
let score = 10;
score = 20; // Pas de problème

const maxScore = 100;
maxScore = 150; // Erreur
```

6. À RETENIR

- Une variable est utilisée pour stocker des données.
- Mots-clés :
  - `var` (ancien, pour compatibilité).
  - `let` (préféré pour les variables réaffectables).
  - `const` (pour les constantes immuables).
- Règles principales :
  - `var` a une portée fonction ou globale.
  - `let` et `const` ont une portée bloc.
  - Les noms doivent être explicites et respecter les conventions.
- Types communs :
  - String, Number, Boolean, Undefined, Null, Objets/Collections.

Utiliser les bonnes pratiques dès le départ facilite un code clair, lisible et modulable. Créez de bonnes habitudes et rappelez-vous : choisissez des noms descriptifs et optez pour `const` ou `let` selon le besoin !