Par Robert DIASSÉ

Bases du JavaScript

Variables et constantes :

- **Variables :** En JavaScript, les variables sont déclarées à l'aide des mots-clés `var`, `let`. Elles sont utilisées pour stocker des valeurs qui peuvent être modifiées au cours de l'exécution du programme.
  Exemple :

```
var age = 25;
let nom = "Jean";
```

- **Constantes :** Les constantes sont déclarées avec le mot-clé `const` et ne peuvent pas être réaffectées après leur initialisation.
  Exemple :

```
const PI = 3.14;
```

Différences de portées entre les déclarations :

- `var` : La portée des variables déclarées avec `var` est fonctionnelle, ce qui signifie qu'elles peuvent être accessibles même en dehors de leur bloc de code parent.
- `let` : La portée des variables déclarées avec `let` est limitée au bloc de code dans lequel elles sont déclarées.
- `const` : Les constantes déclarées avec `const` ne peuvent pas être réaffectées après leur initialisation et ont une portée de bloc.

Types d'opérateurs :

- **Opérateurs arithmétiques :** `+`, `-`, `*`, `/`, `%` (modulo)
- **Opérateurs d'affectation :** `=`, `+=`, `-=`, `*=`, `/=`, etc.
- **Opérateurs de comparaison :** `==`, `===`, `!=`, `!==`, `>`, `<`, `>=`, `<=`
- **Opérateurs logiques :** `&&` (ET), `||` (OU), `!` (NON)

La Console JavaScript et Méthodes d'Affichage

La Console JavaScript :

La console JavaScript est un outil de débogage intégré dans les navigateurs web, qui permet aux développeurs de vérifier et de déboguer leur code JavaScript. Elle affiche des messages, des erreurs, des avertissements et d'autres informations utiles pendant l'exécution du code.

La console est un outil essentiel pour comprendre le comportement de votre code et détecter les erreurs. Voici quelques raisons pour lesquelles elle est importante :

- Affichage des valeurs des variables pendant l'exécution du programme.
- Détection des erreurs de syntaxe, d'exécution et de logique.
- Affichage des messages de débogage pour comprendre le flux d'exécution du programme.
- Mesure des performances du code en affichant le temps d'exécution des fonctions et des boucles.

Méthodes d'Affichage en JavaScript :

JavaScript offre plusieurs méthodes pour afficher des messages, des valeurs et d'autres informations dans la console. Voici les principales méthodes d'affichage :

1. **console.log() :** Utilisée pour afficher des messages simples dans la console.
   Exemple :

```
console.log("Hello, world!");
```

2. **console.error() :** Utilisée pour afficher des messages d'erreur dans la console.
   Exemple :

```
console.error("Une erreur s'est produite!");
```

3. **console.warn() :** Utilisée pour afficher des messages d'avertissement dans la console.
   Exemple :

```
console.warn("Attention : cette action est risquée!");
```

4. **console.info() :** Utilisée pour afficher des messages informatifs dans la console.
   Exemple :

```
console.info("Informations : ce site utilise des cookies.");
```

5. **console.debug() :** Utilisée pour afficher des messages de débogage dans la console. Cette méthode est similaire à `console.log()`, mais est principalement utilisée pour le débogage avancé.

Exemple :

```
console.debug("Valeur de la variable x :", x);
```

6. **console.table() :** Utilisée pour afficher des données tabulaires dans la console.
   Exemple :

```
const users = [
    { name: "Alice", age: 30 },
    { name: "Bob", age: 35 },
    { name: "Charlie", age: 25 }
];
console.table(users);
```

Ces méthodes vous permettent d'afficher différentes informations dans la console JavaScript, ce qui est utile pour le débogage, le suivi des performances et la compréhension du comportement de votre code.

Types de Données en JavaScript

En JavaScript, les variables peuvent stocker différents types de données. Voici les principaux types de données en JavaScript :

1. **Nombre (Number) :** Représente les nombres entiers ou décimaux.
   Exemples :

```
let age = 30; // entier
let prix = 19.99; // décimal
```

2. **Chaîne de caractères (String) :** Représente une séquence de caractères.
   Exemple :

```
let nom = "Jean";
```

3. **Booléen (Boolean) :** Représente une valeur vraie (true) ou fausse (false).
   Exemple :

```
let estMajeur = true;
```

4. **Null :** Représente une valeur nulle ou vide.
   Exemple :

```
let resultat = null;
```

5. **Indéfini (Undefined) :** Représente une variable qui n'a pas encore été affectée ou à laquelle aucune valeur n'a été attribuée.
   Exemple :

```
let prenom;
```

6. **Objet (Object) :** Représente une collection de données et/ou de fonctionnalités.
   Exemple :

```
let personne = { nom: "Alice", age: 25 };
```

7. **Tableau (Array) :** Représente une collection ordonnée d'éléments.
   Exemple :

```
let nombres = [1, 2, 3, 4, 5];
```

Vérification des Types de Données :

Pour vérifier le type de données d'une variable en JavaScript, vous pouvez utiliser l'opérateur `typeof`. Exemple :

```
let x = 10;
console.log(typeof x); // Affiche "number"
```

Conversion de Types :

JavaScript est un langage de typage dynamique, ce qui signifie que les types de données peuvent être convertis implicitement ou explicitement. Voici quelques exemples de conversion de types :

- Conversion explicite :

```
let age = "30";
console.log(typeof age); // Affiche "string"
age = parseInt(age); // Conversion en nombre entier
console.log(typeof age); // Affiche "number"
```

- Conversion implicite :

```
let total = 10 + "20"; // Concaténation de chaînes (conversion implicite)
console.log(total); // Affiche "1020"
```

La connaissance des types de données et de leurs comportements est essentielle pour travailler efficacement avec JavaScript.

Structures conditionnelles :

- **if...else :** Utilisée pour exécuter des blocs de code en fonction d'une condition.
  Exemple :

```
let age = 18;
if (age >= 18) {
    console.log("Vous êtes majeur.");
} else {
    console.log("Vous êtes mineur.");
}
```

- **Ternaire :** Une version abrégée de la structure if...else.
  Exemple :

```
let estMajeur = (age >= 18) ? "majeur" : "mineur";
console.log("Vous êtes " + estMajeur + ".");
```

Structures répétitives :

- **while :** Exécute un bloc de code tant qu'une condition spécifiée est vraie.
  Exemple :

```
let i = 0;
while (i < 5) {
    console.log(i);
    i++;
}
```

- **do...while :** Exécute un bloc de code une fois, puis répète l'exécution tant qu'une condition spécifiée est vraie.
  Exemple :

```
let i = 0;
do {
    console.log(i);
```

```
    i++;
} while (i < 5);
```

- **for :** Exécute un bloc de code un certain nombre de fois.
  Exemple :

```
for (let i = 0; i < 5; i++) {
    console.log(i);
}
```

Exercices Pratiques : Bases du JavaScript

Exercice 1 : Manipulation des Variables et Constantes

1. Déclarez une variable `age` et initialisez-la avec votre âge.
2. Déclarez une constante `PI` et initialisez-la avec la valeur de π (pi).
3. Affichez la valeur de `age` dans la console.
4. Essayez de réaffecter une nouvelle valeur à `PI` et observez ce qui se passe.

Exercice 2 : Utilisation des Opérateurs

1. Déclarez deux variables `x` et `y` et initialisez-les avec des valeurs numériques.
2. Utilisez les opérateurs arithmétiques pour effectuer les opérations suivantes :
   - Additionnez `x` et `y`.
   - Soustrayez `y` de `x`.
   - Multipliez `x` par `y`.
   - Divisez `x` par `y`.
   - Calculez le reste de la division de `x` par `y`.
3. Affichez les résultats de chaque opération dans la console.

Exercice 3 : Structures Conditionnelles

1. Demandez à l'utilisateur son âge à l'aide de la fonction `prompt()` et stockez la valeur dans une variable.
2. Utilisez une structure conditionnelle `if...else` pour afficher un message différent en fonction de l'âge de l'utilisateur :
   - Si l'âge est supérieur ou égal à 18, affichez "Vous êtes majeur."
   - Sinon, affichez "Vous êtes mineur."

Exercice 4 : Structures Répétitives

1. Utilisez une boucle `for` pour afficher les nombres de 1 à 10 dans la console.
2. Utilisez la boucle `for` pour afficher la table de multiplication d'un nombre recupéré avec `promt()`
3. Utilisez une boucle `while` pour afficher les nombres pairs de 2 à 10 dans la console.
4. Utilisez une boucle `do...while` pour demander à l'utilisateur de deviner un nombre entre 1 et 10. Continuez à lui demander tant qu'il ne devine pas le bon nombre.