Premiers pas en JavaScript

Rédaction de scripts JavaScript

Contrairement à PHP, il n'est pas nécessaire d'installer un serveur pour exécuter du code JavaScript. Il suffit d'un navigateur Web pour exécuter du code JavaScript. De plus, il n'y a pas de balise d'ouverture et de fermeture pour les scripts JavaScript. On écrit directement le code JavaScript dans un fichier avec l'extension `.js`.

Variables

Déclaration de variables

En JavaScript, on peut déclarer des variables avec les mots-clés `let`, `const` ou `var`. La déclaration de variables avec `let` et `const` a été introduite dans ECMAScript 2015 (ES6), tandis que la déclaration de variables avec `var` est plus ancienne.

```
let name = "Hassan";
const age = 30;
var city = "Montréal";
```

`let`

La déclaration de variables avec `let` permet de déclarer des variables qui peuvent être réassignées. C'est-à-dire que vous pouvez changer la valeur d'une variable déclarée avec `let`.

```
```

```
let name = "Hassan";
name = "Mike";
```

const

La déclaration de variables avec `const` permet de déclarer des variables qui ne peuvent pas être réassignées. C'est-à-dire que vous ne pouvez pas changer la valeur d'une variable déclarée avec `const`. Vous verrez une erreur si vous essayez de réassigner une variable déclarée avec `const`.

```
const age = 30;
age = 31; // TypeError: Assignment to constant variable.
```

var

La déclaration de variables avec `var` est plus ancienne et est moins utilisée. Les variables déclarées avec `var` peuvent être réassignées et leur portée est différente de celle des variables déclarées avec `let` et `const`. Nous n'entrerons pas dans les détails de la portée des variables dans ce cours. **Règle générale, il est préférable d'utiliser `let` et `const` pour déclarer des variables.**

Nommage des variables

En JavaScript, les noms de variables peuvent contenir des lettres, des chiffres, des caractères de soulignement (`_`) et des signes de dollar (`$`). Les noms de variables ne peuvent pas commencer par un chiffre. De plus, les noms de variables sont sensibles à la casse, c'est-à-dire que `name`, `Name` et `NAME` sont trois variables différentes.

La convention veut que les noms de variables soient écrits en camelCase, c'est-à-dire que la première lettre de chaque mot est en majuscule, sauf pour le premier mot.

```
let firstName = "Mehdi";
let lastName = "Laroussi";
```

Types de données

En JavaScript, les variables peuvent contenir différents types de données, comme des chaînes de caractères, des nombres, des booléens, des tableaux, des objets, etc.

```
let name = "Clara"; // chaîne de caractères
let age = 30; // nombre
let isStudent = true; // booléen
let fruits = ["apple", "banana", "cherry"]; // tableau
```

```
let person = { name: "Guillermo", age: 60 }; // objet
let x; // undefined
```

Le type de données d'une variable peut changer dynamiquement pendant l'exécution du programme. C'est-à-dire qu'une variable peut contenir une chaîne de caractères à un moment donné, puis un nombre à un autre moment.

Les chaînes de caractères

En JavaScript, les chaînes de caractères peuvent être déclarées avec des guillemets simples (`'`), des guillemets doubles (`"`) ou des backticks (`` ` ``). Cependant, si vous ouvrez une chaîne de caractères avec des guillemets simples, vous devez la fermer avec des guillemets simples, et si vous ouvrez une chaîne de caractères avec des guillemets doubles, vous devez la fermer avec des guillemets doubles.

```
let name1 = "Alice";
let name2 = "Bob";
let name3 = `Charlie`;
```

Littéraux de gabarits

Les littéraux de gabarits (template literals) sont des chaînes de caractères délimitées par des backticks comme les accents graves (`` ` ``). Ils permettent d'insérer des expressions JavaScript dans une chaîne de caractères en utilisant la syntaxe `${expression}`.

```
let name = "David";
let greeting = `Hello, ${name}!`;
```

Nous utiliserons principalement les littéraux de gabarits pour joindre des chaînes de caractères et des variables.

Sélectionner un élément HTML

Pour sélectionner un élément HTML avec JavaScript, on utilise la méthode `document.querySelector`. Cette méthode prend en paramètre un sélecteur CSS et retourne le premier élément HTML qui correspond au sélecteur. Tous les sélecteurs CSS sont valides en JavaScript. Si aucun élément ne correspond au sélecteur, la méthode retourne `null`.

```
let title = document.querySelector("h1");
let paragraph = document.querySelector(".content p");
let button = document.querySelector("#submit");
```

Modifier le texte d'un élément HTML

Pour modifier le texte d'un élément HTML avec JavaScript, on utilise la propriété `textContent` de l'élément. Cette propriété permet de lire et de modifier le texte d'un élément HTML.

On sélectionne d'abord l'élément HTML avec `document.querySelector`, on l'enregistre dans une variable puis on modifie la propriété `textContent` de l'élément. **Les propriétés sont séparées par un point (`.`) en JavaScript.**

```
let title = document.querySelector("h1");
title.textContent = "Hello, world!";
```

Chainage de méthodes

En JavaScript, on peut chaîner des méthodes pour exécuter plusieurs méthodes sur un objet. C'est-à-dire qu'on peut appeler une méthode sur le résultat d'une autre méthode.

```
document.querySelector("h1").textContent = "Hello, world!";
```

Cependant, le chainage de méthodes peut rendre le code difficile à lire. Il est préférable de stocker le résultat de la première méthode dans une variable, puis d'appeler la deuxième méthode sur cette variable.

Ajouter des commentaires

En JavaScript, on peut ajouter des commentaires sur une seule ligne avec `//` ou sur plusieurs lignes avec `/*` et `*/`.

```
// Ceci est un commentaire sur une seule ligne

/*
Ceci est un commentaire
sur plusieurs lignes
*/
```