ES6 (ECMAScript 2015) là phiên bản thứ 6 của chuẩn ECMAScript, và nó mang lại rất nhiều tính năng mới, cải thiện cách viết mã JavaScript. Dưới đây là tổng quan về một số tính năng quan trọng trong **ES6** mà bạn nên biết.

**1. Let và Const**

Trước ES6, var là cách duy nhất để khai báo biến, nhưng nó có một số vấn đề về phạm vi (scope) và khả năng gây lỗi không mong muốn. Với ES6, let và const giúp khắc phục những vấn đề này.

* **let**: Khai báo biến với phạm vi trong block (khối lệnh), thay vì phạm vi toàn bộ hàm như var.
* **const**: Khai báo một biến hằng số, nghĩa là không thể gán lại giá trị cho biến này sau khi đã khởi tạo.

javascript

Sao chépChỉnh sửa

let x = 10;

const y = 20;

x = 15; // hợp lệ

// y = 25; // lỗi: không thể gán lại giá trị cho biến hằng số

**2. Arrow Functions (Hàm mũi tên)**

Arrow functions giúp viết hàm ngắn gọn hơn và tự động bind this từ phạm vi bao ngoài.

javascript

Sao chépChỉnh sửa

// Hàm truyền thống

function sum(a, b) {

return a + b;

}

// Hàm mũi tên

const sum = (a, b) => a + b;

Arrow functions còn giúp giữ giá trị this từ phạm vi bên ngoài:

javascript

Sao chépChỉnh sửa

function Timer() {

this.seconds = 0;

setInterval(() => {

this.seconds++;

console.log(this.seconds); // `this` sẽ tham chiếu đến đối tượng Timer

}, 1000);

}

const timer = new Timer();

**3. Template Literals (Template chuỗi)**

Template literals cho phép bạn sử dụng chuỗi nhiều dòng và nhúng biểu thức vào trong chuỗi dễ dàng hơn.

javascript

Sao chépChỉnh sửa

const name = "John";

const age = 30;

const greeting = `Hello, my name is ${name} and I am ${age} years old.`;

console.log(greeting); // "Hello, my name is John and I am 30 years old."

**4. Destructuring**

Destructuring là một cách tiện lợi để trích xuất giá trị từ mảng hoặc đối tượng và gán vào các biến.

**Destructuring với mảng:**

javascript

Sao chépChỉnh sửa

const arr = [1, 2, 3];

const [a, b, c] = arr;

console.log(a, b, c); // 1 2 3

**Destructuring với đối tượng:**

javascript

Sao chépChỉnh sửa

const person = { name: "John", age: 30 };

const { name, age } = person;

console.log(name, age); // John 30

**5. Rest and Spread Operator**

* **Rest Operator (...)** giúp thu thập các phần tử còn lại thành mảng.
* **Spread Operator (...)** giúp phân tách một mảng hoặc đối tượng thành các phần tử riêng biệt.

**Rest Operator**:

javascript

Sao chépChỉnh sửa

const [first, ...rest] = [1, 2, 3, 4];

console.log(first); // 1

console.log(rest); // [2, 3, 4]

**Spread Operator**:

javascript

Sao chépChỉnh sửa

const arr1 = [1, 2, 3];

const arr2 = [...arr1, 4, 5];

console.log(arr2); // [1, 2, 3, 4, 5]

**6. Classes (Lớp đối tượng)**

ES6 giới thiệu cú pháp class để làm việc với đối tượng theo cách dễ hiểu hơn, tương tự như trong các ngôn ngữ lập trình hướng đối tượng khác.

javascript

Sao chépChỉnh sửa

class Person {

constructor(name, age) {

this.name = name;

this.age = age;

}

greet() {

console.log(`Hello, my name is ${this.name}`);

}

}

const person = new Person("John", 30);

person.greet(); // "Hello, my name is John"

**7. Modules (Mô-đun)**

ES6 hỗ trợ chia mã thành các mô-đun, giúp quản lý mã nguồn tốt hơn và dễ dàng tái sử dụng.

**Cách xuất khẩu (export) và nhập khẩu (import) trong ES6**:

javascript

Sao chépChỉnh sửa

// file person.js

export const person = { name: "John", age: 30 };

// file main.js

import { person } from './person.js';

console.log(person.name); // "John"

**8. Promises**

Promising (lời hứa) là cách xử lý bất đồng bộ trong ES6. Nó giúp tránh vấn đề "callback hell" và dễ dàng làm việc với các tác vụ bất đồng bộ.

javascript

Sao chépChỉnh sửa

const myPromise = new Promise((resolve, reject) => {

const success = true;

if (success) {

resolve("Operation was successful");

} else {

reject("Operation failed");

}

});

myPromise

.then(result => console.log(result)) // "Operation was successful"

.catch(error => console.log(error)); // "Operation failed"

**9. Default Parameters (Tham số mặc định)**

ES6 cho phép bạn đặt giá trị mặc định cho tham số hàm.

javascript

Sao chépChỉnh sửa

function greet(name = "Guest") {

console.log(`Hello, ${name}`);

}

greet(); // "Hello, Guest"

greet("John"); // "Hello, John"

**10. Iterators và Generators**

ES6 giới thiệu for...of để duyệt qua các phần tử của mảng hoặc đối tượng iterable. Generator là một loại hàm đặc biệt cho phép bạn tạm dừng và tiếp tục một hàm.

javascript

Sao chépChỉnh sửa

// Iterator

const arr = [1, 2, 3];

for (const num of arr) {

console.log(num); // 1, 2, 3

}

// Generator

function\* generator() {

yield 1;

yield 2;

yield 3;

}

const gen = generator();

console.log(gen.next().value); // 1

console.log(gen.next().value); // 2

console.log(gen.next().value); // 3

**Tóm tắt các tính năng chính của ES6:**

* let và const giúp quản lý phạm vi biến.
* Arrow functions giúp viết mã ngắn gọn và dễ hiểu.
* Template literals cho phép xử lý chuỗi dễ dàng hơn.
* Destructuring giúp trích xuất giá trị từ mảng và đối tượng.
* Rest và Spread operator giúp thao tác mảng và đối tượng dễ dàng hơn.
* Cú pháp class và module giúp quản lý mã nguồn tốt hơn.
* Promises giúp xử lý bất đồng bộ một cách trực quan hơn.

ES6 cải thiện rất nhiều khả năng và cách thức làm việc của JavaScript, khiến nó trở thành một ngôn ngữ hiện đại và dễ sử dụng hơn.

Nếu bạn cần giải thích thêm về bất kỳ tính năng nào trong số này, hãy cho mình biết nhé!

Top of Form

Bottom of Form