ES6+ (ECMAScript 2015 và các phiên bản sau) đã mang lại nhiều tính năng mạnh mẽ, giúp tăng cường khả năng làm việc với JavaScript. Dưới đây là một số tính năng nổi bật của ES6+ và các ví dụ minh họa:

**1. Let và Const**

* **Let**: Khai báo biến với phạm vi khối (block scope).
* **Const**: Dùng để khai báo biến không thể thay đổi giá trị sau khi đã được gán.

**Ví dụ**:

let x = 10;

x = 20; // Được phép thay đổi giá trị

const y = 30;

// y = 40; // Lỗi, không thể thay đổi giá trị của const

**2. Arrow Functions**

* Cung cấp cú pháp rút gọn cho các hàm, và duy trì ngữ cảnh của this.

**Ví dụ**:

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

console.log(sum(2, 3)); // Output: 5

**3. Template Literals**

* Cho phép xây dựng chuỗi dễ dàng với các biểu thức nhúng.

**Ví dụ**:

let name = 'Alice';

let age = 25;

console.log(`My name is ${name} and I am ${age} years old.`);

// Output: My name is Alice and I am 25 years old.

**4. Destructuring Assignment**

* Cho phép trích xuất các giá trị từ mảng hoặc đối tượng một cách dễ dàng.

**Với mảng**:

const arr = [1, 2, 3];

const [x, y] = arr;

console.log(x, y); // Output: 1 2

**Với đối tượng**:

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

const { name, age } = person;

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

**5. Default Parameters**

* Cho phép gán giá trị mặc định cho tham số trong hàm.

**Ví dụ**:

function greet(name = 'Guest') {

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

}

greet(); // Output: Hello, Guest

greet('Alice'); // Output: Hello, Alice

**6. Spread Operator**

* Giúp sao chép hoặc kết hợp các mảng và đối tượng một cách dễ dàng.

**Với mảng**:

const arr1 = [1, 2];

const arr2 = [3, 4];

const combined = [...arr1, ...arr2];

console.log(combined); // Output: [1, 2, 3, 4]

**Với đối tượng**:

const obj1 = { a: 1, b: 2 };

const obj2 = { c: 3, d: 4 };

const merged = { ...obj1, ...obj2 };

console.log(merged); // Output: { a: 1, b: 2, c: 3, d: 4 }

**7. Rest Parameters**

* Thu thập các tham số còn lại trong một mảng.

**Ví dụ**:

function sum(...numbers) {

return numbers.reduce((total, num) => total + num, 0);

}

console.log(sum(1, 2, 3, 4)); // Output: 10

**8. Classes**

* Cung cấp cú pháp để làm việc với các lớp trong JavaScript.

**Ví dụ**:

class Person {

constructor(name, age) {

this.name = name;

this.age = age;

}

greet() {

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

}

}

const person = new Person('Alice', 25);

person.greet(); // Output: Hello, my name is Alice

**9. Promises**

* Quản lý bất đồng bộ với cú pháp dễ hiểu hơn.

**Ví dụ**:

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

let success = true;

if (success) {

resolve('Operation was successful!');

} else {

reject('Operation failed.');

}

});

myPromise

.then((result) => console.log(result)) // Output: Operation was successful!

.catch((error) => console.log(error));

**10. Modules (import/export)**

* Cho phép tách mã thành các mô-đun (module), giúp tổ chức mã dễ dàng hơn.

**file1.js**:

export const greet = (name) => `Hello, ${name}!`;

**file2.js**:

import { greet } from './file1.js';

console.log(greet('Alice')); // Output: Hello, Alice!

**11. Async/Await (ES7+)**

* Cung cấp cách tiếp cận đồng bộ để làm việc với Promise, giúp mã dễ đọc hơn.

**Ví dụ**:

async function fetchData() {

let response = await fetch('https://api.example.com/data');

let data = await response.json();

console.log(data);

}

fetchData();

**12. Object.entries() và Object.values()**

* Object.entries() trả về mảng các cặp key-value.
* Object.values() trả về mảng các giá trị của đối tượng.

**Ví dụ**:

const obj = { a: 1, b: 2, c: 3 };

console.log(Object.entries(obj)); // Output: [['a', 1], ['b', 2], ['c', 3]]

console.log(Object.values(obj)); // Output: [1, 2, 3]

**13. Set và Map**

* **Set**: Lưu trữ các giá trị duy nhất.
* **Map**: Lưu trữ các cặp key-value, với khả năng sử dụng bất kỳ kiểu dữ liệu nào làm key.

**Set**:

const set = new Set([1, 2, 3, 3, 4]);

console.log(set); // Output: Set { 1, 2, 3, 4 }

**Map**:

const map = new Map();

map.set('name', 'Alice');

map.set('age', 25);

console.log(map.get('name')); // Output: Alice

**14. Enhanced Object Literals**

* Cung cấp cách viết ngắn gọn cho đối tượng, giúp mã ngắn gọn và dễ đọc hơn.

**Ví dụ**:

const name = 'Alice';

const person = {

name, // thay vì name: name

greet() {

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

}

};

person.greet(); // Output: Hello, Alice

**Kết luận**

ES6+ mang lại rất nhiều cải tiến đáng giá cho JavaScript, từ cú pháp ngắn gọn hơn, tính năng quản lý bất đồng bộ tốt hơn, cho đến khả năng tái sử dụng mã dễ dàng hơn với mô-đun và lớp. Các tính năng này giúp lập trình viên viết mã hiệu quả hơn, dễ bảo trì và mở rộng hơn.