**Tổng quan về ES6+**

ECMAScript (ES) là tiêu chuẩn của JavaScript. Phiên bản **ES6 (ECMAScript 2015)** đánh dấu một bước tiến lớn với nhiều tính năng mới, và các phiên bản sau như **ES7, ES8, ES9, ES10, ES11, ES12+** tiếp tục cải tiến.

**1. Biến & Phạm vi**

**let và const (Thay thế var)**

* let: Có phạm vi trong block ({}), có thể thay đổi giá trị.
* const: Có phạm vi trong block, không thể thay đổi giá trị.

javascript

Sao chépChỉnh sửa

let x = 10;

x = 20; // ✅ Có thể thay đổi

const y = 30;

y = 40; // ❌ Lỗi, không thể gán lại

**2. Template Literals (Chuỗi nội suy)**

Dùng **`` (backticks)** để chèn biến vào chuỗi dễ dàng hơn.

javascript

Sao chépChỉnh sửa

let name = "John";

let message = `Hello, ${name}!`;

console.log(message); // Hello, John!

**3. Arrow Function**

Cú pháp rút gọn của function.

javascript

Sao chépChỉnh sửa

// Function bình thường

function sayHello(name) {

return `Hello, ${name}!`;

}

// Arrow function

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

console.log(sayHello("Alice")); // Hello, Alice!

✅ Không có **this** riêng, phù hợp cho callback.

**4. Destructuring (Phá vỡ cấu trúc)**

**Array Destructuring**

javascript

Sao chépChỉnh sửa

const numbers = [1, 2, 3];

const [first, second] = numbers;

console.log(first, second); // 1 2

**Object Destructuring**

javascript

Sao chépChỉnh sửa

const person = { name: "Bob", age: 25 };

const { name, age } = person;

console.log(name, age); // Bob 25

**5. Default Parameters**

Giá trị mặc định cho tham số.

javascript

Sao chépChỉnh sửa

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

console.log(greet()); // Hello, Guest!

console.log(greet("Anna")); // Hello, Anna!

**6. Spread & Rest Operators (...)**

**Spread (...)**

Giúp sao chép hoặc nối array, object.

javascript

Sao chépChỉnh sửa

const arr1 = [1, 2, 3];

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

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

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

const obj2 = { ...obj1, c: 3 };

console.log(obj2); // { a: 1, b: 2, c: 3 }

**Rest (...)**

Dùng trong function để nhận nhiều tham số.

javascript

Sao chépChỉnh sửa

const sum = (...numbers) => numbers.reduce((a, b) => a + b, 0);

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

**7. Object Property Shorthand**

Viết gọn object khi biến có cùng tên với key.

javascript

Sao chépChỉnh sửa

const name = "Alice";

const age = 30;

const user = { name, age };

console.log(user); // { name: "Alice", age: 30 }

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

Tách code thành các module.

**Export**

javascript

Sao chépChỉnh sửa

// file math.js

export const add = (a, b) => a + b;

export const multiply = (a, b) => a \* b;

**Import**

javascript

Sao chépChỉnh sửa

// file main.js

import { add, multiply } from "./math.js";

console.log(add(2, 3)); // 5

**9. Optional Chaining (?.)**

Giúp tránh lỗi khi truy cập thuộc tính không tồn tại.

javascript

Sao chépChỉnh sửa

const user = { profile: { name: "Alice" } };

console.log(user.profile?.name); // Alice

console.log(user.address?.city); // undefined (không lỗi)

**10. Nullish Coalescing (??)**

Trả về giá trị bên phải nếu giá trị bên trái là null hoặc undefined.

javascript

Sao chépChỉnh sửa

const username = null;

console.log(username ?? "Guest"); // Guest

const count = 0;

console.log(count ?? 10); // 0 (vì 0 không phải null/undefined)

**11. Promises & Async/Await**

Dùng để xử lý bất đồng bộ.

**Promise**

javascript

Sao chépChỉnh sửa

const fetchData = () => {

return new Promise((resolve) => {

setTimeout(() => resolve("Data Loaded"), 2000);

});

};

fetchData().then((data) => console.log(data)); // Data Loaded (sau 2s)

**Async/Await**

javascript

Sao chépChỉnh sửa

const getData = async () => {

let data = await fetchData();

console.log(data);

};

getData(); // Data Loaded (sau 2s)

**12. Set & Map**

**Set (Tập hợp không trùng lặp)**

javascript

Sao chépChỉnh sửa

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

console.log(uniqueNumbers); // Set {1, 2, 3}

**Map (Lưu key-value, cho phép key là object)**

javascript

Sao chépChỉnh sửa

const userMap = new Map();

userMap.set("name", "Alice");

userMap.set("age", 25);

console.log(userMap.get("name")); // Alice

**13. Dynamic Import**

Chỉ import khi cần.

javascript

Sao chépChỉnh sửa

if (someCondition) {

import("./math.js").then((module) => {

console.log(module.add(2, 3));

});

}

**14. BigInt**

Dùng để lưu số rất lớn.

javascript

Sao chépChỉnh sửa

const bigNumber = 9007199254740991n;

console.log(bigNumber + 1n); // 9007199254740992n

**15. Class trong JavaScript**

ES6 giới thiệu cú pháp class.

javascript

Sao chépChỉnh sửa

class Person {

constructor(name, age) {

this.name = name;

this.age = age;

}

greet() {

return `Hello, my name is ${this.name}`;

}

}

const alice = new Person("Alice", 25);

console.log(alice.greet()); // Hello, my name is Alice

**Kết luận**

ES6+ giúp JavaScript trở nên mạnh mẽ và dễ dùng hơn với:

* Cú pháp gọn hơn (let, const, arrow functions).
* Hỗ trợ lập trình bất đồng bộ (Promise, async/await).
* Cải tiến object & array (destructuring, spread).
* Hỗ trợ modules (import/export).

Bạn cần làm rõ tính năng nào hoặc cần ví dụ chi tiết hơn không? 🚀