**Tổng Quan về ES6+ và Các Tính Năng Quan Trọng**

ES6+ (ECMAScript 2015 và các phiên bản mới hơn như ES7, ES8, ES9,...) đã giới thiệu nhiều cải tiến mạnh mẽ cho JavaScript, giúp code ngắn gọn hơn, dễ đọc hơn và hiệu suất tốt hơn.

Dưới đây là các tính năng quan trọng kèm theo ví dụ:

**1. let & const (Khai báo biến)**

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

🔹 **Ví dụ:**

js

Sao chépChỉnh sửa

let age = 25;

age = 26; // ✅ Được phép

const name = "Tuấn";

// name = "Hữu"; // ❌ Lỗi: Không thể gán lại giá trị

**2. Arrow Function (=>)**

* Viết ngắn gọn hơn so với function
* Không có this riêng (kế thừa this của ngữ cảnh bên ngoài)

🔹 **Ví dụ:**

js

Sao chépChỉnh sửa

const greet = (name) => `Xin chào, ${name}!`;

console.log(greet("Tuấn")); // "Xin chào, Tuấn!"

**3. Template Literals (Chuỗi template)**

* Dùng dấu **backticks (`)** để nội suy biến và viết nhiều dòng.

🔹 **Ví dụ:**

js

Sao chépChỉnh sửa

const username = "Tuấn";

const message = `Chào ${username},

chúc bạn một ngày tốt lành!`;

console.log(message);

**4. Destructuring (Phân rã)**

* Trích xuất dữ liệu từ **object** hoặc **array** một cách dễ dàng.

🔹 **Ví dụ với Object:**

js

Sao chépChỉnh sửa

const user = { name: "Tuấn", age: 25 };

const { name, age } = user;

console.log(name, age); // "Tuấn", 25

🔹 **Ví dụ với Array:**

js

Sao chépChỉnh sửa

const arr = [1, 2, 3];

const [a, b, c] = arr;

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

**5. Default Parameters (Giá trị mặc định)**

* Giúp đặt giá trị mặc định khi không truyền đối số.

🔹 **Ví dụ:**

js

Sao chépChỉnh sửa

const greet = (name = "bạn") => `Xin chào, ${name}!`;

console.log(greet()); // "Xin chào, bạn!"

console.log(greet("Tuấn")); // "Xin chào, Tuấn!"

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

* **Spread (...)**: Sao chép hoặc mở rộng mảng, object.
* **Rest (...)**: Gộp nhiều giá trị thành một mảng.

🔹 **Ví dụ với Spread:**

js

Sao chépChỉnh sửa

const numbers = [1, 2, 3];

const newNumbers = [...numbers, 4, 5];

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

const obj1 = { name: "Tuấn", age: 25 };

const obj2 = { ...obj1, city: "Hà Nội" };

console.log(obj2); // { name: "Tuấn", age: 25, city: "Hà Nội" }

🔹 **Ví dụ với Rest:**

js

Sao chépChỉnh sửa

const sum = (...nums) => nums.reduce((acc, num) => acc + num, 0);

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

**7. Enhanced Object Literals**

* Khi khai báo object, có thể viết gọn hơn.

🔹 **Ví dụ:**

js

Sao chépChỉnh sửa

const name = "Tuấn";

const age = 25;

const user = { name, age }; // Không cần viết name: name, age: age

console.log(user); // { name: "Tuấn", age: 25 }

**8. Promises & Async/Await**

* **Promise**: Giúp xử lý bất đồng bộ dễ hơn với .then()
* **Async/Await**: Viết code bất đồng bộ như đồng bộ.

🔹 **Ví dụ với Promise:**

js

Sao chépChỉnh sửa

const fetchData = () => {

return new Promise((resolve, reject) => {

setTimeout(() => resolve("Dữ liệu đã tải!"), 2000);

});

};

fetchData().then(data => console.log(data)); // "Dữ liệu đã tải!" (sau 2 giây)

🔹 **Ví dụ với Async/Await:**

js

Sao chépChỉnh sửa

const fetchData = async () => {

const data = await fetchData();

console.log(data);

};

fetchData();

**9. Modules (Import/Export)**

* Giúp chia code thành các file riêng biệt.

🔹 **export.js**

js

Sao chépChỉnh sửa

export const PI = 3.14;

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

🔹 **import.js**

js

Sao chépChỉnh sửa

import { PI, add } from './export.js';

console.log(PI); // 3.14

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

**10. Optional Chaining (?.)**

* Giúp truy cập thuộc tính sâu trong object mà không gây lỗi.

🔹 **Ví dụ:**

js

Sao chépChỉnh sửa

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

console.log(user?.profile?.name); // "Tuấn"

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

**11. Nullish Coalescing (??)**

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

🔹 **Ví dụ:**

js

Sao chépChỉnh sửa

const username = null;

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

const age = 0;

console.log(age ?? 18); // 0 (vì 0 không phải null/undefined)

**12. BigInt (Số lớn)**

* Dùng để lưu số nguyên cực lớn.

🔹 **Ví dụ:**

js

Sao chépChỉnh sửa

const bigNum = 9007199254740991n;

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

**Kết luận**

Các tính năng ES6+ giúp viết JavaScript hiện đại hơn, dễ đọc và bảo trì hơn. Bạn có thể sử dụng ES6+ trong React, Node.js hay bất kỳ ứng dụng JS nào để cải thiện hiệu suất và code quality. 🚀