📖 Giáo Trình Chuyên Sâu: Object trong JavaScript

I. Tổng Quan về Object

Giới thiệu Object trong JavaScript

- Tại sao Object quan trọng?
- So sánh Object với Array

Các cách tạo Object:

- Object Literal (`{}`)
- `new Object()`
- Factory Function
- Constructor Function
- Class (ES6+)

II. Thuộc Tính và Phương Thức của Object

Truy cập thuộc tính:

- Dấu chấm (`.`)
- Dấu ngoặc vuông (`[]`)

Thêm, sửa và xóa thuộc tính

Thuộc tính động (Computed Property)

Định nghĩa phương thức trong Object

`this` trong Object – Cách hoạt động và các vấn đề thường gặp

III. Duyệt và Xử Lý Object

Duyệt Object với `for...in`

Các phương thức làm việc với Object:

- `Object.keys()`
- `Object.values()`
- `Object.entries()`
- `Object.assign()`

Sao chép Object (Shallow Copy vs Deep Copy)

- `Object.assign()`
- `JSON.parse(JSON.stringify())`
- `structuredClone()`
- `_.cloneDeep()` (Lodash)

IV. Quản Lý Object - Object Freeze & Seal

Đóng băng Object với `Object.freeze()`

Niêm phong Object với `Object.seal()`

Khác biệt giữa `freeze()`, `seal()` và `preventExtensions()`

Kiểm tra trạng thái Object:

- `Object.isFrozen()`
- `Object.isSealed()`
- `Object.isExtensible()`

V. Prototype và Kế Thừa trong Object

Khái niệm Prototype trong JavaScript

Tạo và kế thừa Prototype

So sánh Prototype với Class

Các phương thức liên quan:

- `Object.create()`
- `Object.getPrototypeOf()`
- `Object.setPrototypeOf()`
- `hasOwnProperty()`

VI. Lập Trình Hướng Đối Tượng (OOP) với Object

Bốn nguyên tắc OOP trong JavaScript:

- Đóng gói (Encapsulation)
- Kế thừa (Inheritance)
- Đa hình (Polymorphism)
- Trừu tượng hóa (Abstraction)

Factory Function vs Constructor Function

Class và ES6 Enhancements

- `constructor()`
- `static methods`
- `getters/setters`
- `super()` và kế thừa
- Tính Private trong Object (`#property`)

VII. Quản Lý Bộ Nhớ và Garbage Collection

Cách JavaScript quản lý bộ nhớ

Cơ chế Garbage Collection (GC)

Thuật toán Mark-and-Sweep

Memory Leak trong Object

Cách tối ưu Object để tránh rò rỉ bộ nhớ

`WeakMap` và `WeakSet` – Giải pháp cho Garbage Collection

VIII. Những Tính Năng Mới & Best Practices

Optional Chaining (`?.`)

Nullish Coalescing (`??`)

Object Destructuring và Rest/Spread Operator

Best Practices khi làm việc với Object

- Tránh sử dụng `delete` thường xuyên
- Sử dụng `Object.freeze()` khi cần bảo vệ dữ liệu
- Khi nào nên dùng `class`, khi nào dùng `function`?