Mục tiêu:

1. Nắm được tổng quan các thao tác làm việc với Object ở cấp độ dễ.
2. Biết cách khởi tạo, truy cập, thêm, sửa, xóa thuộc tính trong Object.

Tổng quan về Object trong JavaScript

Trong **JavaScript**, **object** là một kiểu dữ liệu quan trọng và linh hoạt, dùng để lưu trữ các cặp **key-value** (thuộc tính và giá trị). Mỗi key trong object là một **string** hoặc **symbol**, và mỗi value có thể là bất kỳ kiểu dữ liệu nào, bao gồm cả function (được gọi là phương thức).

---

1. Khởi tạo Object

Có nhiều cách để tạo object trong JavaScript:

Object literal

Cách phổ biến và ngắn gọn nhất:

```
const person = {
    name: 'hoangnm',
    age: 33,
    greet: function() {
        console.log('Hello');
    }
};
```

Sử dụng hàm khởi tạo `new Object()`

```
const person = new Object();
person.name = 'Hoang';
person.age = 33;
```

Lưu ý: Các cách dùng class (ES6) hoặc dùng constructor function sẽ được học ở những bài học nâng cao

---

2. Thuộc tính và phương thức trong Object

Thuộc tính

Là các key-value dùng để lưu trữ thông tin.

```
const person = {
    name: 'ThayHoangJS',
    age: 33
};

console.log(person.name); // ThayHoangJS
console.log(person.age); // 33
```

Phương thức

Là các function được gán làm thuộc tính.

```
const person = {
    name: 'ThayHoangJS',
    greet() {
        console.log(`Hello, I am ${this.name}`);
    }
};
person.greet(); // Hello, I am ThayHoangJS
```

Với cách dùng trên, dù Object có thay đổi tên thì phương thức vẫn hoạt động đúng. Chúng ta đã sử dụng `this` để truy cập thuộc tính của Object hiện tại.

---

3. Truy cập thuộc tính

- **Dấu chấm** `.`: `object.property`.
- **Ngoặc vuông** `[]`: `object['property']`.

Ví dụ:

```
const obj = { key: 'value' };
console.log(obj.key);
console.log(obj['key']);
```

Lưu ý:

- Sử dụng dấu chấm `.` khi tên thuộc tính không chứa ký tự đặc biệt.
- Sử dụng ngoặc vuông `[]` khi tên thuộc tính chứa ký tự đặc biệt hoặc biến (giá trị động) - hay còn gọi là **Computed Property**.

---

4. Thêm, sửa và xóa thuộc tính

Thêm/Sửa

```
const obj = {};
obj.name = 'Hoang';
obj.age = 25;
obj.name = 'ThayHoangJS';
```

Xóa

```
delete obj.name;
console.log(obj.name);
```

---

5. Duyệt qua Object

`for...in`

Duyệt qua tất cả các key của object:

```
const person = { name: 'John', age: 30 };
for (let key in person) {
    console.log(key, person[key]);
}
```

Object Methods

- `Object.keys()`: Lấy danh sách các key.
- `Object.values()`: Lấy danh sách các value.
- `Object.entries()`: Lấy cặp key-value dưới dạng mảng.

Ví dụ:

```
const person = { name: 'ThayHoangJS', age: 33 };
console.log(Object.keys(person)); // ['name', 'age']
console.log(Object.values(person)); // ['ThayHoangJS', 33]
console.log(Object.entries(person)); // [['name', 'ThayHoangJS'], ['age',
33]]
```

---

6. Object Reference và Copy

Object trong JavaScript là kiểu tham chiếu (**reference type**):

Gán Object cho biến khác

Khi gán object cho biến khác, cả hai biến cùng tham chiếu đến một object (một địa chỉ ô nhớ):

```
const obj1 = { name: 'Hoang' };
const obj2 = obj1;
obj2.name = 'ThayHoangJS';
console.log(obj1.name); // ThayHoangJS (thay đổi cả 2 biến)
```

Sao chép Object

- **Sử dụng `Object.assign()`**:

```
const obj = { name: 'Hoang' };
const copy = Object.assign({}, obj);
copy.name = 'ThayHoangJS';
console.log(obj.name); // 'Hoang'
```

- `Object.assign()` là phương thức sao chép các thuộc tính từ một hoặc nhiều object nguồn vào object đích, trả về object đích bị thay đổi.
- `Object.assign()` chỉ sao chép các thuộc tính ở mức độ 1, nếu thuộc tính là object thì nó sẽ tham chiếu đến cùng một object. (Đây gọi là shallow copy).
- `Objects.assign()` cho phép gộp nhiều object thành một object đích (Object target).

Lưu ý: Khi muốn deep copy object, cần sử dụng các cách khác như **JSON.parse()** hoặc **lodash**. Hiện nay JS hỗ trợ deep copy thông qua structuredClone

```
const obj = { name: 'Hoang' };
const copy = JSON.parse(JSON.stringify(obj));
copy.name = 'ThayHoangJS';
console.log(obj.name); // 'Hoang'
```

```
import _ from 'lodash';

const obj = { name: 'Hoang' };
const copy = _.cloneDeep(obj);
copy.name = 'ThayHoangJS';
console.log(obj.name); // 'Hoang'
```

```
const obj = { name: 'Hoang' };
const copy = structuredClone(obj);
copy.name = 'ThayHoangJS';
console.log(obj.name); // 'Hoang'
```

Hiện nay ES6 hỗ trợ deep copy thông qua **structuredClone**:

```
const obj = { name: 'Hoang' };
const copy = structuredClone(obj);
copy.name = 'ThayHoangJS';

console.log(obj.name); // 'Hoang'
```

---

7. Một số tính năng nâng cao

Destructuring

Trích xuất thuộc tính từ object:

```
const person = { name: 'John', age: 30 };
const { name, age } = person;
console.log(name, age); // John 30
```

Rest Parameter

```
const person = { name: 'John', age: 30, country: 'USA' };
const { name, ...rest } = person;
console.log(rest); // { age: 30, country: 'USA' }
```

Spread Operator

```
const person = { name: 'John', age: 30 };
const newPerson = { ...person, country: 'USA' };
console.log(newPerson); // { name: 'John', age: 30, country: 'USA' }
```

Shorthand Property

```
const name = 'Alice';
const person = { name }; // { name: 'Alice' }
```

Computed Property

```
const key = 'age';
const person = { [key]: 30 };
console.log(person.age); // 30
```

Kiến thức cần nhớ

- **Object** là một kiểu dữ liệu quan trọng trong JavaScript, dùng để lưu trữ các cặp key-value.
- Có nhiều cách khởi tạo object: object literal, `new Object()`, class, constructor function.
- Truy cập thuộc tính bằng dấu chấm `.` hoặc ngoặc vuông `[]`. Sử dụng `[]` khi tên thuộc tính chứa ký tự đặc biệt hoặc biến (giá trị động).