Tổng quan

JavaScript sử dụng bộ nhớ máy tính thông qua heap (bộ nhớ động) và stack (bộ nhớ ngăn xếp) để lưu trữ và quản lý dữ liệu trong quá trình thực thi. Dưới đây là cách JavaScript quản lý bộ nhớ và một số cơ chế quan trọng liên quan:

1. Bộ nhớ stack và heap

Bộ nhớ stack (bộ nhớ ngăn xếp)

Dùng để lưu trữ các dữ liệu tĩnh, bao gồm:

- Các giá trị nguyên thuỷ (primitive values): number, string, boolean, null, undefined, symbol, và bigint.
- Tham chiếu tới các hàm hoặc đối tượng trong heap.
- Được tổ chức theo kiểu LIFO (Last In, First Out), tức là dữ liệu được đưa vào cuối và lấy ra đầu tiên.
- Kích thước của stack thường cố định và nhỏ hơn heap, vì nó được thiết kế để lưu trữ dữ liệu tạm thời, ngắn hạn.

```
function add(a, b) {
    const sum = a + b; // Giá trị của `sum` được lưu trong stack
    return sum;
}

add(5, 3); // Giá trị 5, 3, và 8 được lưu trên stack trong khi thực thi
```

Hãy tưởng tượng FILO CỦA STACK như một ngăn xếp đĩa, bạn đặt đĩa vào cuối và lấy ra đĩa đầu tiên.

```
function first() {
    console.log("First function");
    second();
}

function second() {
    console.log("Second function");
    third();
}

function third() {
    console.log("Third function");
}

first();
```

Bộ nhớ Stack xử lý như sau (FILO):

1. first() được gọi → Đưa vào Stack.
2. first() gọi second() → second() vào Stack trên first().
3. second() gọi third() → third() vào Stack trên second().
4. third() chạy xong → Lấy third() ra khỏi Stack.
5. second() chạy xong → Lấy second() ra khỏi Stack.
6. first() chạy xong → Lấy first() ra khỏi Stack.

Vì stack giống như một ngăn xếp đĩa, nên khi first() chạy xong, second() mới chạy, khi second() chạy xong thì third() mới chạy. Đó cùng là lý do nó không thể xử lý được những dữ liệu có thể thay đổi kích thước như array, object. Khi đó chúng ta cần đến heap.

Bộ nhớ heap (bộ nhớ động)

- Dùng để lưu trữ dữ liệu phức tạp (non-primitive values) như objects, arrays, và functions.
- Kích thước linh hoạt hơn stack và thường lớn hơn.
- Quản lý bộ nhớ trên heap phức tạp hơn vì nó không có thứ tự rõ ràng, và cần cơ chế dọn dẹp (garbage collection) để giải phóng bộ nhớ.

```
const user = {
    name: "Hoang",
    age: 33
};

// Biến `user` được lưu trong stack, nhưng dữ liệu thực tế của từng thuộc tính (name, age) được lưu trong heap.
```

Cơ chế tham chiếu trong bộ nhớ

Giá trị nguyên thuỷ

- Khi gán một giá trị nguyên thuỷ cho một biến, giá trị đó được lưu trực tiếp trong stack.

```
let a = 5; // Giá trị 5 được lưu trực tiếp trong stack
```

- Khi gán hoặc truyền vào hàm, JavaScript tạo bản sao giá trị (pass by value).

```
let x = 10;
let y = x; // Bản sao của `x` được gán cho `y`
y = 20;

console.log(x); // 10 (không bị ảnh hưởng bởi thay đổi của `y`)
console.log(y); // 20
```