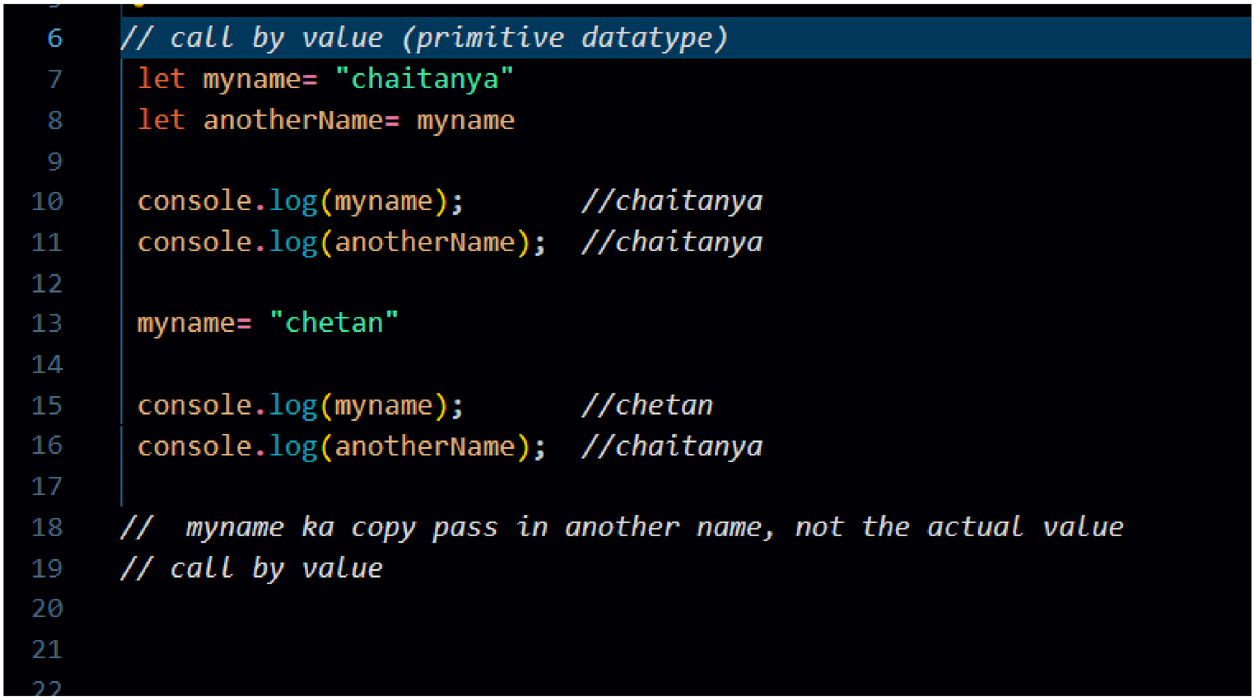
Memory

Memory in JavaScript



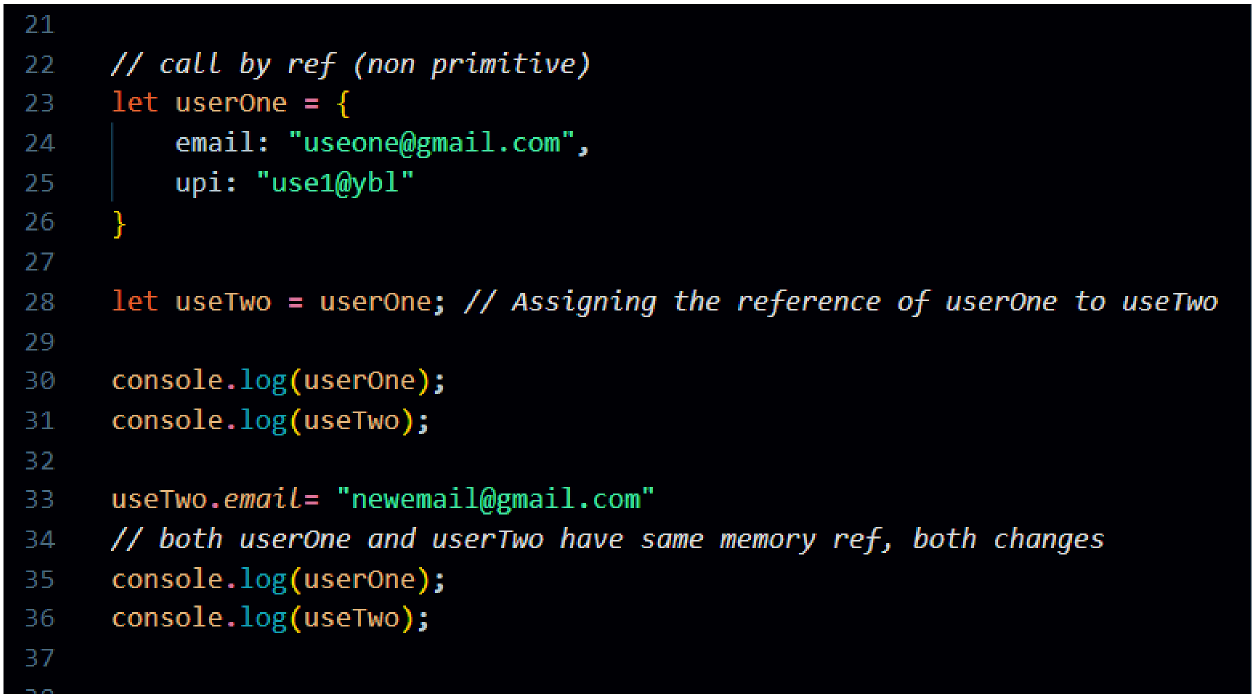
```
// call by value (primitive datatype)
 let myname= "chaitanya"
 let anotherName= myname

 console.log(myname);       //chaitanya
 console.log(anotherName);  //chaitanya

 myname= "chetan"

 console.log(myname);       //chetan
 console.log(anotherName);  //chaitanya

//  myname ka copy pass in another name, not the actual value
// call by value
```



```
// call by ref (non primitive)
let userOne = {
    email: "useone@gmail.com",
    upi: "use1@ybl"
}

let useTwo = userOne; // Assigning the reference of userOne to useTwo

console.log(userOne);
console.log(useTwo);

useTwo.email= "newemail@gmail.com"
// both userOne and userTwo have same memory ref, both changes
console.log(userOne);
console.log(useTwo);
```

Stack Memory

- Primitive types (such as `number, string, boolean, null, undefined, symbol, and bigint`) are stored in stack memory
- Execution is fast because stack memory has a fixed size.
- It follows the LIFO (Last In, First Out) principle.
- When a function executes, its local variables are stored in the stack, and once the function execution is complete, they are removed from the stack.

```
-----------------------------------------------
|               Stack Memory               |
-----------------------------------------------
| b → 20                              |
| a → 10                              |
-----------------------------------------------

let a = 10;  // Primitive type (Stack)
let b = a;   // `b` ko `a` ka copy milta hai
b = 20;      // `b` change hota hai, but `a` same rehta hai
```

```
console.log(a); // 10
console.log(b); // 20
```

Heap Memory

- Objects and reference(Non-Primitive) types (such as `arrays, functions, objects, heap, graph, tree`) are stored in heap memory.
- Heap memory has a dynamic size, which makes it slower compared to stack memory.
- When an object is created, memory is allocated in the heap, and its reference is stored in stack memory.
- **Garbage Collector (GC) unused objects ko clean** karta hai jo kisi bhi reference se accessible nahi hote in JS.

```
-------------------------------------------------
|              Stack Memory                 |
-------------------------------------------------
| obj1 → (Heap Ref: 0x1)                    |
| obj2 → (Heap Ref: 0x1)                    |
-------------------------------------------------

-------------------------------------------------
|              Heap Memory                  |
-------------------------------------------------
| (0x1) → { name: "Amit" }                  |
-------------------------------------------------

let obj1 = { name: "Rahul" };  // Heap me allocate hota hai
let obj2 = obj1;  // Stack me reference copy hota hai (Heap ka address)
obj2.name = "Amit";  // Heap me change hone se dono references pe effect

console.log(obj1.name); // "Amit"
console.log(obj2.name); // "Amit"
```