// function love(obj1){

//   console.log(obj1);
// }

// love(obj);
// will changes in obj1 be reflected in obj?

If you write:

```
function love(obj1){
    console.log(obj1);
}

love(obj);
```

and then modify the properties of `obj1` inside the function `love`, those changes WILL be reflected in the original `obj` outside the function.

Reason:
In JavaScript, when you pass an object to a function, what actually gets passed is a copy of the reference to that object. Both `obj` and `obj1` point to the same object in memory. So, if you change a property (like `obj1.name = "Rahul"`), the original `obj` will also reflect this change.[1] [2] [3]

> Changing properties of `obj1` inside the function will affect `obj` outside the function.

However, if you reassign `obj1` within the function (like `obj1 = {}`), the original `obj` will NOT be affected — only the local variable inside the function changes its reference.[1]

Example:

```
function love(obj1){
    obj1.amount = 1000; // This WILL change obj.amount outside the function
}

love(obj);
console.log(obj.amount); // 1000
```

But:

```
function love(obj1){
    obj1 = { name: 'Changed' }; // This does NOT change obj outside the function
}

love(obj);
console.log(obj.name); // "Rohit"
```

Summary:

- Mutating properties on `obj1` affects `obj`.
- Reassigning `obj1 = something` does NOT affect `obj`.

⁂

1. https://www.geeksforgeeks.org/javascript/pass-by-value-and-pass-by-reference-in-javascript/
2. https://flexiple.com/javascript/javascript-pass-by-reference-or-value
3. https://codedamn.com/news/javascript/pass-by-reference-in-javascript