In JavaScript, **shallow copy** and **deep copy** are two types of copying methods used when you want to duplicate an object or an array. They differ in how they handle nested objects or arrays inside the original object or array.

**1. Shallow Copy**

A shallow copy means that the top-level elements (the first level) are copied to the new object or array, but if there are nested objects or arrays inside, they are still references to the original ones. This means that if you modify a nested object in the copied structure, the original structure will also be affected.

**Example of Shallow Copy:**

let original = {

name: "John",

address: {

city: "New York",

zip: "10001"

}

};

// Creating a shallow copy using Object.assign

let shallowCopy = Object.assign({}, original);

// Modifying a nested object in the shallow copy

shallowCopy.address.city = "Los Angeles";

console.log(original.address.city); // "Los Angeles" (original is affected)

console.log(shallowCopy.address.city); // "Los Angeles"

**Explanation:**

* Object.assign() creates a shallow copy of the original object. It copies the top-level properties, but for nested objects like address, it only copies the reference to the original object.
* Changing shallowCopy.address.city also changes original.address.city because both point to the same nested object.

**2. Deep Copy**

A deep copy means that all objects or arrays are recursively copied, including any nested objects or arrays. This ensures that the original object and the copied one are completely independent of each other. Changes to nested properties in the deep copy will not affect the original object.

**Example of Deep Copy:**

let original = {

name: "John",

address: {

city: "New York",

zip: "10001"

}

};

// Creating a deep copy using JSON methods

let deepCopy = JSON.parse(JSON.stringify(original));

// Modifying a nested object in the deep copy

deepCopy.address.city = "Los Angeles";

console.log(original.address.city); // "New York" (original is NOT affected)

console.log(deepCopy.address.city); // "Los Angeles"

**Explanation:**

* JSON.parse(JSON.stringify(original)) creates a deep copy. It serializes the original object into a JSON string and then deserializes it back into a new object. This results in a completely independent copy, including any nested objects.
* Changes to deepCopy.address.city do not affect original.address.city because all nested objects are independently copied.