**Shallow Vs Deep**

**Shallow Copy :**

A shallow copy occurs when you copy the reference of an object to a new variable. In this process, only the top-level properties are copied, while nested objects or arrays still reference the original memory location. This means that if you change the nested properties in one object, those changes will reflect in the other because they share the same memory reference.

When you assign one object to another using the assignment operator (=), a shallow copy is created

This is known as a shallow copy. The newly created object has the same memory address as the old one.

Hence, any change made to either of them changes the attributes for both. To overcome this problem, a deep copy is used. If one of them is removed from memory, the other one ceases to exist. In a way the two objects are interdependent.

## ****Deep Copy****

A deep copy, on the other hand, creates a completely independent copy of the object, including all nested objects or arrays. This ensures that changes made to one object do not affect the other. Each object is stored in a separate memory location, making them entirely independent.

Now to create a deep copy of an object in JavaScript we use JSON.parse() and JSON.stringify() methods.

the new object is created using the JSON.parse() and JSON.stringify() methods of JavaScript. JSON.stringify() takes a JavaScript object as an argument and then transforms it into a JSON string. This JSON string is passed to the JSON.parse() method which then transforms it into a JavaScript object.

This method is useful when the object is small and has serializable properties. But if the object is very large and contains certain non-serializable properties then there is a risk of data loss. Especially if an object contains methods then JSON.stringify() will fail as methods are non-serializable. There are better ways to a deep clone of which one is Lodash which allows cloning methods as well.