Copy by Value vs Copy by reference

- In JavaScript, there are two ways to copy viz. 1. Copy by value 2. Copy by reference

- The process of copying in primitive data types (var, let, const) happens by the 1st
- method i: e Copy by value whereas the process of copying in composite data types (array, object) happens by the 2nd method.

E: g (1)

```
var a = 1;
var b = a;

a=5;

console.log(a);
console.log(b);
```

Output:
5
1

- Above given is an example of copy by value. At the 2nd line (var b=a), the value of 'variable a' is assigned to 'variable b'.
- Hence, after changing the value of a to 5 (3rd line), it doesn't affect the value of 'variable b'.

E: g (2)

```
var a = [1,2,3,4];
var b = a;

a.push(5);

console.log(a);
console.log(b);
```

Output:
[1,2,3,4,5]
[1,2,3,4,5]

- Above given is an example of copy by reference. At the 2nd line (var b = a), the address of array-a is assigned to array-b.
- Which means both the variable arrays point at the same address/location. Hence, the changes made in the 1st array(a.push(5)) are reflected in the 2nd array also.

How to copy by value a composite datatype (array + objects)

1. Using spread operator

E: g (1)

```
var a = [1,2,3,4];
var b = […a];
a.push(5);

console.log(a);
console.log(b);
```

Output:
[1,2,3,4,5]
[1,2,3,4]

- As shown in above example, at 2nd line (var b = […a]) 'array b' is created and all the elements from 'array a' are copied in it. But both the arrays are totally separate and point at different memory locations.
- Hence, if 'array a' is altered, the changes will not be reflected in 'array b'.

2. Using Object.assign()

E: g (2)

```
obj1 = {a:1,b:2};
obj2 = Object.assign({},obj1);
obj1.c = 3;
console.log(obj1);
console.log(obj2);
```

- The Object.assign() method copies all enumerable own properties from one or more source objects to a target object. It returns the target object.

3. Copying values one by one

E: g (3)

```
var a = [1,2,3,4];
var b = [];

for(var i=0; i<a.length;i++)
{
        b[i]=a[i];
}
a.push(5);
```

Output:

[1,2,3,4,5]
[1,2,3,4]

- As shown in above example, one by one, all the elements of ‘array a’ are copied into ‘array b’ inside for loop(b[i]=a[i]).
- Hence, after pushing an element into ‘array a’ doesn’t affect the ‘array b’.