2. **Copy by value v/s copy by reference:**

- Copy by value:
  - It applies to primitive data types i.e. Numbers, Boolean and Strings.
  - Example:

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

  - In the above example we have assigned value 10 to variable 'a' and 'a' to variable 'b'.
  - Since both of these variables are primitive data types, both variables points towards different memory location i.e. it has a copy of its own.
  - It means change in the value of variable 'b' doesn't affect the value of variable 'a'.
- Copy by reference:
  - It applies to composite data types i.e. Arrays, Objects and Functions.
  - Example:

```
let arr1 = [1,2,3];
let arr2 = arr1;
console.log(arr1); //[1,2,3]
arr2.push(4);
console.log(arr1); //[1,2,3,4]
console.log(arr2); //[1,2,3,4]
```

  - In the above example we have assigned value [1,2,3] to array 'arr1' and 'arr1' to array 'arr2'.
  - Here change in the value of array arr2 affects array arr1 since both points towards same memory location.
  - Hence both the arrays print same result.

3. **Copy by value a composite data type:**

- Using Spread operator:
  - We can use spread operator to implement this.
  - Example:

```
let arr1 = [1,2,3];
let arr2 = [...arr1];
console.log(arr1); //[1,2,3]
arr2.push(4);
```

```
console.log(arr1); //[1,2,3]
console.log(arr2); //[1,2,3,4]
```

- Here array 'arr2' makes the copy of its own in the form of [...arr1].
- [...arr1] takes all the elements of an array 'arr1' and appends those elements in the array 'arr2'. That means both arr1 and arr2 are pointing towards different memory location.
- So the arr1 doesn't get altered even after we make changes in arr2.

- Using Object.assign():
  - The Object.assign() method copies all enumarable own properties from one or more source objects to a target object.
  - This will be a shallow caopy.
  - In the Object.assign method 1st argument must be empty [ ]or{ }.
  - Example:

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

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