**Difference between copy by value and copy by reference:**

|  |  |
| --- | --- |
| **copy by value** | **copy by reference** |
| For primitive datatypes: boolean, null, undefined, string and number | For composite datatypes: Array and Objects |
| let a = 10;  let b=a;  a points to memory where 10 is stored  b points to a different memory where the value of a is copied into | let a = [ 1, 2, 3];  let b = a;  a points to memory where the array object( [ 1, 2, 3] ) is stored  b points to a (which points to [ 1, 2, 3] ) |
| **=** is used | **=** is used |
| **=**  copies the value on the right side expression to the left side variable | **=**  copies the address of the right side expression to the left side variable |
| **===** checks the value and its type for primitive types | **===** checks whether the memory address is same else returns false for reference types |

**copy by value a composite datatype (array+objects):**

There are basically 3 options

**Spread operator(...)**

let a = [1, 2, 3];

let b = [...a];

**Object.assign()**

let a = [1, 2, 3];

let b = Object.assign([ ], a);

**JSON.parse() and JSON.stringify()**

let a = [1, 2, 3];

let b = JSON.parse(JSON.stringify(a));

Spread and object.assign performs **shallow copy** while JSON.parse(JSON.stringify( )) performs **deep copy**.

Shallow arrays and objects are those which have only one level of values:

let a = [ 1, “string”, null];

let b = { ‘name’ : xyz , ‘age’ : 55 }

Deep arrays and objects are those which have more than one level of values:

let a = [ 1, “string”, null, [ ‘abc’ , ‘bcd’ , 88] ];

let b = { ‘name’ : xyz , ‘age’ : 55, [ ‘abc’ , ‘bcd’ , 88] , { ‘lop’ : 678 , ‘days’ : [14, 15, 29] } };