So , basically in JavaScript we have 3 datatypes, trivial, primitive and composite. For primitive data types(number , string) we do apply copy by value(deep copy) to pass the value of one variable into another. But in the case of composite datatype(arrays, objects) we do copy by reference(shallow copy).

Copy by reference has its own advantages in regard to processing time and memory space when dealing with huge amount of data. But how do we apply copy by value over these composite data.

*Arrays***:**

Coming to this data type we can copy the value by various methods.

So the first method is to use slice() operation which will clone the array and then return a new reference to the second array.

*var arr1 = [1,2,3];  
var arr2 = arr1.slice();  
arr2[0] = 8;  
  
console.log(arr1); //1 2 3  
console.log(arr2);//8 2 3*

Second method will be to use spreads (…) to copy one array into another.

*var arr1 = [1,2,3];  
var arr2 = […arr1];  
arr2[0] = 8;  
  
console.log(arr1);  
console.log(arr2);*

Objects:

So to copy by value for a JavaScript object there are three ways to do so.

First method would be to use the spreads (…).

*var obj1 = {  
left : { “car” : 4, “bus” : 2, “bike” : 3 },  
right : null,  
data : 234  
};  
  
var obj2 = { …obj1 };  
obj2.center = null;  
  
console.log(obj1);  
console.log(obj2);*

Second method would be to use *Object.assign()*method.

*var obj1 = {  
left : { “car” : 4, “bus” : 2, “bike” : 3 },  
right : null,  
data : 234  
};  
  
var obj2 = Object.assign({}, obj1);  
obj2.center = null;  
  
console.log(obj1);  
console.log(obj2);*

Third method would be to use JSON.stringify() and JSON.parse() methods.

*var obj1 = {  
left : { “car” : 4, “bus” : 2, “bike” : 3 },  
right : null,  
data : 234  
};  
  
var obj2 = JSON.parse(JSON.stringify(obj1));  
obj2.center = 999;  
  
console.log(obj1);  
console.log(obj2);*

In JavaScript, you use variables to store values that can be primitive or references. When you make a copy of a value stored in a variable, you create a new variable with the same value. For a primitive value, you just simply use a simple assignment. And when you change the value of the copied variable, the value of the original remains the same. If you use the assignment operator for a reference value, it will not copy the value. Instead, both variables will reference the same object in the memory. And when access the object via the new variable and change the value of its property , you change the value of the property of the object.