1. Difference between copy by value and copy by reference.

Copy by Value:

All the primitive types such as Boolean, null, undefined, string and number are passed by value.

Example:

var x = 1;

var y = ‘hi’;

var z = null;

var a = x;

var b = y;

var c = z;

console.log(x); //1

console.log(y); //hi

console.log(z); //null

console.log(a); //1

console.log(b); //hi

console.log(c); //null

In this example the values of variables x, y, z are copied to the variables a, b, c. The value of x, y, z are not affected or changed when we copy the primitive data types. It means these variables do not have relationship with one another.

Copy by Reference:

All the non-primitive values such as array, function and object are copied by reference. Generally these non-primitive values are collectively called as objects. When an object is passed by reference, the address of the object in the memory is copied to the new variable and referenced. Hence, any changes made to the referenced object will affect the original object.

Example:

var x = {dog:’Poodle’}

var y = x;

y.size = ‘large’;

console.log(x); // {dog:’Poodle’,size:’large’}

console.log(y); // {dog:’Poodle’,size:’large’}

In this example, we have referenced variable y to x and added a new property to the variable y. So, any changes made to y will also be reflected in x.

1. How do you copy by value a composite data type in JavaScript?

There are three ways to copy by value for a composite datatype:

* + Spread(…) operator
  + Object.assign() method
  + JSON.stringify() and JSON.parse()