**TASK 2**

1. **Write a write up on copy by value and copy by reference**

There are two kinds of data types in Javascript, Primitive and Composite. String, Number, Boolean, Undefined come under primitive data types. The variables defined with this data type hold the values directly. And when a variable having a value is assigned to another variable, the value is copied to it. For example,

Location2

Location1

let a = 10;

**10**

**10**

let b = a;

console.log(a, b) //output: 10, 10 **a**  **b**

Here two variables are stored in two different locations of memory. In other words, copy of variable ‘a’ is created. When a is assigned to b, the value 10 will be copied and stored in b. Both have same value. But the variables are in two different locations, so, if now we change the value of one variable the other one is not affected. For example,

b = 50;

console.log(a, b) // Output: 10, 50

The data types of arrays and objects come under composite data types, which means that the variables defined of these type do not store the value directly but they hold a reference to the object value. They point to the location or address of memory where the object value resides.

When a variable of this data type is assigned to another variable, the reference is assigned to the variable and not the value itself. So both the variables wirll be pointing to the same address location in memory.

**X**

Rose, lily, jasmine

For eample,

**arr2**

Location **X**

Location **X**

let arr1 = [“rose”, “lily”, “jasmine”];

**arr1**

let arr2 = arr1;

console.log(arr1, arr2); // output: [‘rose’, ‘lily’, ‘jasmine’], [‘rose’, ‘lily’, ‘jasmine’]

Here, when we pass arrr1 and arr2 as parameters to the console log function, javascript engine goes to the address location which is pointed by both these variables and fetch the value from there. Since both point to the same reference or location, changing the value of one array affects the other. If I add an element to arr2, it will be reflected even in arr1.

arr2[3] = “marigold”;

console.log(arr1); //output: [‘rose’, ‘lili’, ‘jasmine’, ‘marigold’]

This is call by reference.

1. **How to copy by value a composite data type(array+objects)?**

In composite data types when a variable is assigned to another they are copied by reference and not value. To copy by value, spread operator **(…**) can be used. It creates a copy of the first variable and assigns its values to the second one.

It can be done as,

let arr1 = [1, 2, 3];

let arr2 = **[…**arr1];

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

This operator creates a copy of arr1 and hence changing one of the arrays does not affect another. Like, if arr2[0] = [4];

Console.log(arra1, arr2); //output: [1, 2, 3], [4, 2, 3]

The same apllies to even object data types.

For example,

let obj1 = {name:”abc”, age”:10”, hobby:”painting”};

let obj2 = {**…**obj1};

console.log(obj1, obj2);

//output: {name:”abc”, age”:10”, hobby:”painting”},

{name:”abc”, age”:10”, hobby:”painting”}

Now, if I changethe age property of second object, it will not affect the first objectas the variables are copied by value using spread operator.

obj2.age = 20;

console.log(obj1, obj2);

//output: {name:”abc”, age:10, hobby:”painting”},

{name:”abc”, age:20, hobby:”painting”}