Task No.2

Q.1) Load the rest countries data using your html and script.js file and run a for loop on the data and print all the country name in the console.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Guvi Class</title>

</head>

<body>

    <script src="script1.js"></script>

</body>

</html>

let request = new XMLHttpRequest();

request.open('GET','https://restcountries.eu/rest/v2/all',true);

request.send();

request.onload = function()

{

    let data = JSON.parse(this.response);

    for(let i in data)

    {

        console.log(data[i].name);

    }

}

Q.2) Give a write up on Difference between copy by value and copy by reference.

**1.Copy by Value**

In a primitive data-type when a variable is assigned a value we can imagine that a box is created in the memory. This box has a sticker attached to it i.e. the variable name. Inside the box the value assigned to the variable is stored.

For e.g.

var x = 17;

var y = ‘xyz’;

var z = null;

In above code, ‘x’ contains value 17 and ‘y’ contains ‘xyz’

var x = 17;

var y = ‘xyz’;

var z = null;

var a = x;

var b = y;

console.log(x, y, a, b); // -> 17, ‘xyz’, 17, ‘xyz’

In above code; the**values** in the boxes ‘x’ and ‘y’ are copied into the variables ‘a’ and ‘b’.

At this point of time both ‘x’ and ‘a’ contains the value 17. Both ‘y’ and ‘b’ contains the value ‘xyz’. However, an important thing to understand here is that even though ‘x’ and ‘a’ as well as ‘y’ and ‘b’ contains the same value they are not connected to each other. It is so because the values are directly copied into the new variables.

Changes taking place in one does not affect the other.

var x = 17;

var y = ‘xyz’;

var z = null;

var a = x;

var b = y;

x = 5;

y = ‘abc’;

console.log(x, y, a, b) // -> 5, ‘abc’, 17, ‘xyz

**Copy By Reference**

In case of a non-primitive data-type the values are not directly copied. When a non-primitive data-type is assigned a value, a box is created with a sticker of the name of the data-type. However, the values it is assigned is not stored directly in the box. The language itself assigns a different memory location to store the data. The address of this memory location is stored in the box created.

Let user = { name : ‘ajay’ };

Let admin = user;

admin.name = ‘vijay’; // value changed

alert(user.name) // name changed to shyam

In above code, when the value of admin is changed it automatically changes the value of user as well.

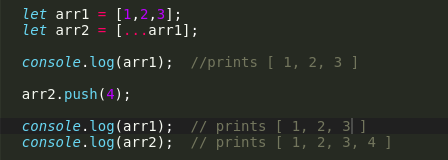
This happens because both ‘user’ and ‘admin’ are storing the address of the memory location. And when one changes the values in the allocated memory it is reflected in the other as well.

We can further elaborate it we can say that; copy by reference is like having two keys of the same room shared between ‘admin’ and ‘user’. If one of them alters the arrangement of the room the other would experience it as well.

Q.3) How to copy by value a composite datatype (array+objects).

We can use the concept of **spread operator (…)**

To have a better understanding lets look at the below example



So, arr2 makes a copy of its own in the form of […arr1].  
What […arr1] does here is, it takes in an array arr1 and expands it into individual elements, and again it takes the form of array and gets assigned to arr2.

So, the arr1 doesn’t get altered even after we make changes to array arr2, after the usage of spread operator.