Task 2

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.
2. Give a write up on Difference between copy by value and copy by reference.
3. How to copy by value a composite datatype (array + objects).

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.

Index.html

```
<!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>Document</title>
</head>
<body>
  <script src="script.js"></script>
</body>
</html>
```

```
script.js
//create a request instance
var req= new XMLHttpRequest();
//initiate a connection
req.open('GET','https://restcountries.eu/rest/v2/all',true);
//sending the request
req.send();
//load the function this function will trigger only when data received successfully
req.onload=function()
{
var data=JSON.parse(this.response);
//console.log(data);
for (var key in data) {
 console.log(data[key].name);
   }
}
```

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

Javascript has 5 data types that are passed by ***value***: Boolean, null, undefined, String, and Number. We'll call these **primitive types**.

Javascript has 3 data types that are passed by ***reference***: Array, Function, and Object. These are all technically Objects, so we'll refer to them collectively as **Objects**.

Primitives

If a primitive type is assigned to a variable, we can think of that variable as containing the primitive value.

var x = 10;

var y = 'abc';

var z = null;

When we assign these variables to other variables using =, we copy the value to the new variable. They are **copied by value.**

```
var x = 10;
var y = 'abc';var a = x;
var b = y;console.log(x, y, a, b); // -> 10, 'abc', 10, 'abc'
```

Changing one does not change the other. Think of the variables as having no relationship to each other.

```
var x = 10;
var y = 'abc';var a = x;
var b = y;a = 5;
b = 'def';console.log(x, y, a, b); // -> 10, 'abc', 5, 'def'
```

Objects

Variables that are assigned a non-primitive value are given a reference to that value. That reference points to the object's location in memory. The variables don't actually contain the value.

Objects are created at some location in your computer's memory. When we write arr = [], we've created an array in memory. What the variable arr receives is the address, the location, of that array.

Let's pretend that address is a new data type that is passed by value, just like number or string.

An address points to the location, in memory, of a value that is passed by reference. Just like a string is denoted by quotation marks ('' or ""), an address will be denoted by arrow brackets, <>.

When we assign and use a reference-type variable, what we write and see is:

```
1) var arr = [];
2) arr.push(1);
```

the value, the address, contained by the variable arr is static. The array in memory is what changes. When we use arr to do something, such as pushing a value, the Javascript engine goes to the location of arr in memory and works with the information stored there.

Assigning by Reference

When a reference type value, an object, is copied to another variable using =, the address of that value is what's actually copied over as if it were a primitive. Objects are copied by reference instead of by value.

```
var reference = [1];
var refCopy = reference;
```

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

Arrays, objects, functions are all of object type which comes under composite data types. As we know variable holds data in case of composite data type it holds reference that is address of that particular value in memory.

var a= 10; // here a holds the value 10.

var b= [10,20] // b holds some address like 8023 etc.. not 10 and 20.

Therefore we can't clone data in composite data types. To do that spread operator is used, that is three dots (…), it spreads the elements of that particular array or object and its values can be used to assign to some other variable.

Example :

```
var a= [1,2,3,4,5];
var b= [...a];
var c=a;
a[0]=99;
Console.log(a);
```

Console.log(b);

Console.log(c);

Output :

99,2,3,4,5

1,2,3,4,5

99,2,3,4,5