**B22 WE - Task 2**

**Q1: Load the rest countries data using your HTML and script.js files and run a loop on the data and print all the countryname in a console.**

**Ans:**

**index.html:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<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:**

// step 1. create a request variable

let requestURL = 'https://restcountries.eu/rest/v2/all';

let request = new XMLHttpRequest();

// step 2.create a new http connection

request.open('GET', requestURL, true);

// step 3. set expected response type and send request

request.responseType = 'json';

request.send();

// step 4. load response and print country name

request.onload = function () {

const restCountries = this.response;

printCountryNames(restCountries);

}

// function to loop through rest countries and print their names

function printCountryNames(restCountries) {

for (let i in restCountries) {

console.log(restCountries[i]['name']);

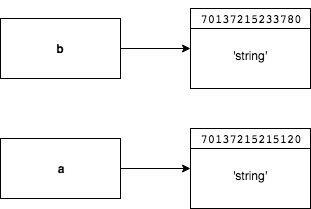
}

}

**Q2: Write a write up on difference between copy by value and copy by reference.**

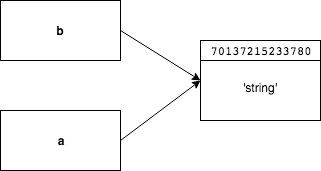
**Ans:**

**Copy by value:** constitutes copying of data, where changes to the copied value are not reflected in the original value. Each copy points to different memory address



b = a, a and b are pointing to different memory addresses

**Copy by Reference** consitutes aliasing of data, where changes to the aliased value are reflected in the original value. Each copy points to the same memory location.



b = a, a and b are pointing to the same memory address

in JavaScript, Primitive data types (Boolean, null, undefined, String, and Number) are copied by value, whereas Objects (Array, Function, and Object)are copied by reference value.

**Q3: How to copy by value a composite datatype (array + object)**

**Ans:**

Composite datatype can be copied by value using one of the methods mentioned below:

1. Using the spread (...) operator
2. Using the Object.assign() method
3. Using the JSON.stringify() and JSON.parse() methods

1. Using the spread (...) operator:

**//copy by value using spread operator**

let a = [1,2,3];

let b = a;

let c = [...a];

console.log(a,b,c); //[ 1, 2, 3 ] [ 1, 2, 3 ] [ 1, 2, 3 ]

a[1] = 100;

console.log(a,b,c); // [ 1, 100, 3 ] [ 1, 100, 3 ] [ 1, 2, 3 ]

2. Using Object.assign() method:

**//copy by value using Object.assign() method**

let a = [2,3,5,7,9];

let b = Object.assign([],a);

console.log(a,b); // [ 2, 3, 5, 7, 9 ] [ 2, 3, 5, 7, 9 ]

a[3] = 10;

console.log(a,b); // [ 2, 3, 5, 10, 9 ] [ 2, 3, 5, 7, 9 ]

3. Using the JSON.stringify() and JSON.parse() methods:

**//copy by value using JSON.stringify() and JSON.parse() method**

let a = [1,1,2,3,5,8];

let b = JSON.parse(JSON.stringify(a));

console.log(a,b); // [ 1, 1, 2, 3, 5, 8 ] [ 1, 1, 2, 3, 5, 8 ]

a[0] = 10;

console.log(a,b); // [ 10, 1, 2, 3, 5, 8 ] [ 1, 1, 2, 3, 5, 8 ]

Both spread (...) and Object.assign() perform a **shallow copy** while the copying using JSON methods carry a **deep copy**. Shallow copy copies only reference in case of nested objects. Also, JSON method doesn’t work in case of copying object with a method.

**//copy by value using JSON.stringify() & JSON.parse() method deep copy ex.**

let a = [1,2,[3],[5,8],[13,21,34]];

let b = JSON.parse(JSON.stringify(a));

console.log(a); // [ 1, 2, [ 3 ], [ 5, 8 ], [ 13, 21, 34 ] ]

console.log(b); // [ 1, 2, [ 3 ], [ 5, 8 ], [ 13, 21, 34 ] ]

a[0] = 10;

a[4][0] = 9;

console.log(a); // [ **10**, 2, [ 3 ], [ 5, 8 ], [ **9**, 21, 34 ] ]

console.log(b); // [ 1, 2, [ 3 ], [ 5, 8 ], [ 13, 21, 34 ] ]

**//copy by value using Object.assign method – shallow copy ex.**

let a = [1,2,[3],[5,8],[13,21,34]];

let b = Object.assign([],a);

console.log(a); // [ 1, 2, [ 3 ], [ 5, 8 ], [ 13, 21, 34 ] ]

console.log(b); // [ 1, 2, [ 3 ], [ 5, 8 ], [ 13, 21, 34 ] ]

a[0] = 10;

a[4][0] = 9;

console.log(a); // [ **10**, 2, [ 3 ], [ 5, 8 ], [ **9**, 21, 34 ] ]

console.log(b); // [ 1, 2, [ 3 ], [ 5, 8 ], [ **9**, 21, 34 ] ]