**Task - 2 Assignment**

**Question 1 : html and script.js file and run a for loop on the data and print all the country names in the console.**

**Index.html file :**

**<!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 variable for XMLHttpRequest**

**var request = new XMLHttpRequest();**

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

**request.send();**

**request.onload = function(){**

**var countryData = JSON.parse(this.response);**

**for(var i in countryData){**

**console.log(countryData[i].name);**

**}**

**}**

**Question 2 : Write a up on Difference Between Copy By Value and Copy By Reference**

**Copy By Value**

1.In copy by value separated memory allocated to each variable,It is primarily used for primitive data-types such as Boolean, Null, Undefined, String, Number when a variable is assigned to a value, for example

1. var X =17

var Y = ‘abc’

‘X’ contains value 17, ‘Y’ contains ‘abc’.

**Next example,**

var X =17

var Y = ‘abc’

var a = X

var b = Y

X = 10

Y = fgh

console.log(X,Y,a,b)

**// OUTPUT :- 10,fgh,17, abc**

**Explaination:-**

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 ‘abc’. However,‘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. variables are independent of each other.

**Copy By Reference**

1.In copy by reference supports a non-primitive data-types such as Array, Function, Objects 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.

For Example,

let user = {name: 'abc'};

let admin = user;

admin.name = 'xyz';

console.log(user.name);

**//OUTPUT :- xyz**

**Explanation:-**

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.

**Question 3: How do you copy by value a composite data type(array+objects)**

There are 3 ways to copy by value for composite data types.

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

The spread operator is used three dots (…), the elements of that particular array or object and its values can be used to assign to some other variable. It is mostly used in the variable array where there is more than 1 values are expected. It allows us the privilege to obtain a list of parameters from an array. Using spread will clone your object.

For 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]**

1. **Using Object.assign() method**

The **Object.assign()** method copies all enumerable own properties from one or more source objects to a target object. It returns the target object.

For Example,

var a = [1,2,3];

var b = object.assign([],a);

console.log(a,b);

**// OUTPUT :- [1,2,3] [1,2,3]**

b[2] = 100;

console.log(a,b);

**// OUTPUT :- [1,2,3] [1,2,100]**

1. **Using the JSON.stringify() and JSON.parse() methods**

The JSON object, available in all modern browsers, has two useful methods to deal with JSON-formatted content: parse and stringify. JSON.parse() takes a JSON string and transforms it into a JavaScript object. JSON.stringify() takes a JavaScript object and transforms it into a JSON string. Using JSON.parse() and JSON.stringify() for copy performs deep copy .

For Example,

var a = [1,2,3];

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

console.log(a,b);

**//OUTPUT : - [1,2,3] [1,2,3]**

var b[2] = 99

Console.log(a,b);

**//OUTPUT:- [1,2,3] [1,2,99]**