JAVASCRIPT - TASK 2

1. HTML and script.js file and run for a loop on the data and print all the country names in the console

- CODE : “Did in Visual code editor and copied from there”
- index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>sample api data</title>
</head>
<body>
<script src = "script.js"></script>
</body>
</html>
```

- script.js

```
var request = new XMLHttpRequest();
request.open('GET','https://restcountries.eu/rest/v2/all', true);
request.send();
request.onload = function (){
    var data = JSON.parse(this.response);
    for(var i in data)
    { console.log(data[i].name); }
}
```

OUTPUT:

- Afghanistan
- script.js:16 Åland Islands
- script.js:16 Albania
- script.js:16 Algeria
- script.js:16 American Samoa
- script.js:16 Andorra
- script.js:16 Angola
- script.js:16 Anguilla
- script.js:16 Antarctica
- script.js:16 Antigua and Barbuda
- script.js:16 Argentina
- script.js:16 Armenia
- script.js:16 Aruba
- script.js:16 Australia
- script.js:16 Austria

- script.js:16 Azerbaijan
- script.js:16 Bahamas
- script.js:16 Bahrain
- script.js:16 Bangladesh
- script.js:16 Barbados
- script.js:16 Belarus
- script.js:16 Belgium
- script.js:16 Belize
- script.js:16 Benin
- script.js:16 Bermuda
- script.js:16 Bhutan
- script.js:16 Bolivia (Plurinational State of)
- script.js:16 Bonaire, Sint Eustatius and Saba
- script.js:16 Bosnia and Herzegovina
- script.js:16 Botswana
- script.js:16 Bouvet Island
- script.js:16 Brazil
- script.js:16 British Indian Ocean Territory
- script.js:16 United States Minor Outlying Islands
- script.js:16 Virgin Islands (British)
- script.js:16 Virgin Islands (U.S.)
- script.js:16 Brunei Darussalam
- script.js:16 Bulgaria
- script.js:16 Burkina Faso
- script.js:16 Burundi
- script.js:16 Cambodia
- script.js:16 Cameroon
- script.js:16 Canada
- script.js:16 Cabo Verde
- script.js:16 Cayman Islands
- script.js:16 Central African Republic
- script.js:16 Chad
- script.js:16 Chile
- script.js:16 China
- script.js:16 Christmas Island
- script.js:16 Cocos (Keeling) Islands
- script.js:16 Colombia
- script.js:16 Comoros
- script.js:16 Congo
- script.js:16 Congo (Democratic Republic of the)
- script.js:16 Cook Islands
- script.js:16 Costa Rica
- script.js:16 Croatia
- script.js:16 Cuba
- script.js:16 Curaçao
- script.js:16 Cyprus
- script.js:16 Czech Republic
- script.js:16 Denmark
- script.js:16 Djibouti

- **script.js:16 Dominica**
- **script.js:16 Dominican Republic**
- **script.js:16 Ecuador**
- **script.js:16 Egypt**
- **script.js:16 El Salvador**
- **script.js:16 Equatorial Guinea**
- **script.js:16 Eritrea**
- **script.js:16 Estonia**
- **script.js:16 Ethiopia**
- **script.js:16 Falkland Islands (Malvinas)**
- **script.js:16 Faroe Islands**
- **script.js:16 Fiji**
- **script.js:16 Finland**
- **script.js:16 France**
- **script.js:16 French Guiana**
- **script.js:16 French Polynesia**
- **script.js:16 French Southern Territories**
- **script.js:16 Gabon**
- **script.js:16 Gambia**
- **script.js:16 Georgia**
- **script.js:16 Germany**
- **script.js:16 Ghana**
- **script.js:16 Gibraltar**
- **script.js:16 Greece**
- **script.js:16 Greenland**
- **script.js:16 Grenada**
- **script.js:16 Guadeloupe**
- **script.js:16 Guam**
- **script.js:16 Guatemala**
- **script.js:16 Guernsey**
- **script.js:16 Guinea**
- **script.js:16 Guinea-Bissau**
- **script.js:16 Guyana**
- **script.js:16 Haiti**
- **script.js:16 Heard Island and McDonald Islands**
- **script.js:16 Holy See**
- **script.js:16 Honduras**
- **script.js:16 Hong Kong**
- **script.js:16 Hungary**
- **script.js:16 Iceland**
- **script.js:16 India**
- **script.js:16 Indonesia**
- **script.js:16 Côte d'Ivoire**
- **script.js:16 Iran (Islamic Republic of)**
- **script.js:16 Iraq**
- **script.js:16 Ireland**
- **script.js:16 Isle of Man**
- **script.js:16 Israel**
- **script.js:16 Italy**

- **script.js:16 Jamaica**
- **script.js:16 Japan**
- **script.js:16 Jersey**
- **script.js:16 Jordan**
- **script.js:16 Kazakhstan**
- **script.js:16 Kenya**
- **script.js:16 Kiribati**
- **script.js:16 Kuwait**
- **script.js:16 Kyrgyzstan**
- **script.js:16 Lao People's Democratic Republic**
- **script.js:16 Latvia**
- **script.js:16 Lebanon**
- **script.js:16 Lesotho**
- **script.js:16 Liberia**
- **script.js:16 Libya**
- **script.js:16 Liechtenstein**
- **script.js:16 Lithuania**
- **script.js:16 Luxembourg**
- **script.js:16 Macao**
- **script.js:16 Macedonia (the former Yugoslav Republic of)**
- **script.js:16 Madagascar**
- **script.js:16 Malawi**
- **script.js:16 Malaysia**
- **script.js:16 Maldives**
- **script.js:16 Mali**
- **script.js:16 Malta**
- **script.js:16 Marshall Islands**
- **script.js:16 Martinique**
- **script.js:16 Mauritania**
- **script.js:16 Mauritius**
- **script.js:16 Mayotte**
- **script.js:16 Mexico**
- **script.js:16 Micronesia (Federated States of)**
- **script.js:16 Moldova (Republic of)**
- **script.js:16 Monaco**
- **script.js:16 Mongolia**
- **script.js:16 Montenegro**
- **script.js:16 Montserrat**
- **script.js:16 Morocco**
- **script.js:16 Mozambique**
- **script.js:16 Myanmar**
- **script.js:16 Namibia**
- **script.js:16 Nauru**
- **script.js:16 Nepal**
- **script.js:16 Netherlands**
- **script.js:16 New Caledonia**
- **script.js:16 New Zealand**
- **script.js:16 Nicaragua**
- **script.js:16 Niger**

- **script.js:16 Nigeria**
- **script.js:16 Niue**
- **script.js:16 Norfolk Island**
- **script.js:16 Korea (Democratic People's Republic of)**
- **script.js:16 Northern Mariana Islands**
- **script.js:16 Norway**
- **script.js:16 Oman**
- **script.js:16 Pakistan**
- **script.js:16 Palau**
- **script.js:16 Palestine, State of**
- **script.js:16 Panama**
- **script.js:16 Papua New Guinea**
- **script.js:16 Paraguay**
- **script.js:16 Peru**
- **script.js:16 Philippines**
- **script.js:16 Pitcairn**
- **script.js:16 Poland**
- **script.js:16 Portugal**
- **script.js:16 Puerto Rico**
- **script.js:16 Qatar**
- **script.js:16 Republic of Kosovo**
- **script.js:16 Réunion**
- **script.js:16 Romania**
- **script.js:16 Russian Federation**
- **script.js:16 Rwanda**
- **script.js:16 Saint Barthélemy**
- **script.js:16 Saint Helena, Ascension and Tristan da Cunha**
- **script.js:16 Saint Kitts and Nevis**
- **script.js:16 Saint Lucia**
- **script.js:16 Saint Martin (French part)**
- **script.js:16 Saint Pierre and Miquelon**
- **script.js:16 Saint Vincent and the Grenadines**
- **script.js:16 Samoa**
- **script.js:16 San Marino**
- **script.js:16 Sao Tome and Principe**
- **script.js:16 Saudi Arabia**
- **script.js:16 Senegal**
- **script.js:16 Serbia**
- **script.js:16 Seychelles**
- **script.js:16 Sierra Leone**
- **script.js:16 Singapore**
- **script.js:16 Sint Maarten (Dutch part)**
- **script.js:16 Slovakia**
- **script.js:16 Slovenia**
- **script.js:16 Solomon Islands**
- **script.js:16 Somalia**
- **script.js:16 South Africa**
- **script.js:16 South Georgia and the South Sandwich Islands**
- **script.js:16 Korea (Republic of)**

- **script.js:16 South Sudan**
- **script.js:16 Spain**
- **script.js:16 Sri Lanka**
- **script.js:16 Sudan**
- **script.js:16 Suriname**
- **script.js:16 Svalbard and Jan Mayen**
- **script.js:16 Swaziland**
- **script.js:16 Sweden**
- **script.js:16 Switzerland**
- **script.js:16 Syrian Arab Republic**
- **script.js:16 Taiwan**
- **script.js:16 Tajikistan**
- **script.js:16 Tanzania, United Republic of**
- **script.js:16 Thailand**
- **script.js:16 Timor-Leste**
- **script.js:16 Togo**
- **script.js:16 Tokelau**
- **script.js:16 Tonga**
- **script.js:16 Trinidad and Tobago**
- **script.js:16 Tunisia**
- **script.js:16 Turkey**
- **script.js:16 Turkmenistan**
- **script.js:16 Turks and Caicos Islands**
- **script.js:16 Tuvalu**
- **script.js:16 Uganda**
- **script.js:16 Ukraine**
- **script.js:16 United Arab Emirates**
- **script.js:16 United Kingdom of Great Britain and Northern Ireland**
- **script.js:16 United States of America**
- **script.js:16 Uruguay**
- **script.js:16 Uzbekistan**
- **script.js:16 Vanuatu**
- **script.js:16 Venezuela (Bolivarian Republic of)**
- **script.js:16 Viet Nam**
- **script.js:16 Wallis and Futuna**
- **script.js:16 Western Sahara**
- **script.js:16 Yemen**
- **script.js:16 Zambia**
- **script.js:16 Zimbabwe**

2. Write a writeup on the difference between copy by value and copy by reference?

A. Copy by value:

- **In the 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 variable name attached to it.**

- Inside the box, the value assigned to the variable is stored.
- Eg: var a = 5;
  var b = 'abc';
  var c = null;
  var x = a;
  var y = b;
  conole.log(a,b,x,y);

  Output:
  5, 'abc', 5, 'abc'
  - Here in this example, the values 'a' and b are copied to variable 'x' and 'y'.
  - At this point of time, both 'a' and 'x' contains the value 5.
  - Both 'b' and 'y' contains the value 'abc'.
  - However, an important thing to understand here is that even though 'a' and 'x', as well as 'b' and 'y', contains the same value they are not connected to each other.
  - It is so because the values are directly copied into the variables.
  - Changes taking place in one does not affect the other.

Copy by reference:

- In the case of non-primitive data type, the values are not directly copied.
- When a non-primitive data type is assigned a value 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.
- Eg: let user = { name: 'Rahul' };
  let admin = user;
  admin.name = 'Harsha'; // value changed
  alert(user.name); // name changed to 'Harsha'
  - Here in this example, when the value admin is changed it automatically changes the value of the user as well.
  - This happens because both user and admin are storing the address of the memory location.
  - And when one changes the value in the allocated memory ist is reflected in the other as well.

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

A. JavaScript can define arrays and objects as a composite data sort. Composite data types are referenced in non-primitive data types, whereas primitive data types such as integer, string are given value.

PASS BY REFERENCE FOR COMPOSITE DATATYPE:

```
var arr = [1,2,3];
var arr1 = arr;
console.log(arr);    // prints[1,2,3]
arr1[0] = 0;
console.log(arr);    // prints[0,2,3]
console.log(arr1);   // prints[0,2,3]
```

In the above example, we have allocated 'arr' array to 'arr1' for arrays and we try to modify 'arr1' array, 'arr' array also effects because when assigning (arr1 = arr), we don't actually make a copy of 'arr' array, 'arr1' array points to memory where 'arr' array points, so if we change 'arr1' array, it affects 'arr' array. We only have one memory spot where 'arr' and 'arr1' are pointing.

We can overcome this problem with the help of the three methods

- **spread operator(...) in javascript.**

```
Eg:     var arr = [1,2,3];
        var arr1 = [...arr];
        console.log(arr);     //prints[1,2,3]
        arr1[0] = 0;
        console.log(arr);     //prints[1,2,3]
        console.log(arr1);    //prints[0,2,3]
```

We used the spread operator for 'arr' array in the above example, because it generates a copy of 'arr' array and 'arr1' points to that copy, so if we try to alter 'arr1' array, it doesn't affect 'arr' array, since we have two separate array copies now.

- **We can also use "Object.assign() method" ; "JSON.stringify()" and JSON parse() methods**
- **For Eg: const person = { firstname = 'Rahul',**
  lastname = 'Wesley' };

```
// using Object.assign() method;
 let N = Object.assign{(), person};

// using JSON
let M = JSON.parse(JSON.stringify(person));
```

4. JSON task

Problem 0 : Part A (15 mins):

Playing with JSON object's Values:

Fluffy sorry, Fluffyy is my fav cat and it has 2 catFriends

Write a code to get the below details of Fluffyy so that

I can take him to vet.

```
var cat = {
name: 'Fluffy',
activities: ['play', 'eat cat food'],
catFriends: [
{
```

```
name: 'bar',
activities: ['be grumpy', 'eat bread omblet'],
weight: 8,
furcolor: 'white'},
{
name: 'foo',
activities: ['sleep', 'pre-sleep naps'],
weight: 3
}
]
}
console.log(cat);
```

Basic Tasks to play with JSON

1. **Add height and weight to Fluffy**

```
cat.weight = 7;
cat.height = 3;
```

2. **Fluffy name is spelt wrongly. Update it to Fluffyy**

```
cat.name = "Fluffyy";
```

3. **List all the activities of Fluffyy's catFriends.**

```
console.log(cat.activities);
```

4. **Print the catFriends names.**

```
console.log(cat.catFriends.name)
```

5. **Print the total weight of catFriends**

```
var sum=0;
for(var i in cat.catFriends)
{  sum = sum+cat.catFriends[i].weight;  }
console.log(sum);
```

6. **Print the total activities of all cats (op:6)**

```
for (var i in cat.catFrineds)

{ console.log(cat.catFriends[i].activities); }
```

7. **Add 2 more activities to bar & foo cats**

```
cat.catFriends[0].activities[2]="jumping";
cat.catFriends[0].activities[3]="rolling";
cat.catFriends[1].activities[2]="laughing";
cat.catFriends[1].activities[3]="eat junk food";
console.log(cat.catFriends[0].activities);
console.log(cat.catFriends[1].activities);
```

8. **Update the fur color of bar**

```
cat.catFriends[0].furcolor = “black”;

console.log(cat.catFriends[0].furcolor);
```

Problem 0 : Part B (15 mins):

Iterating with JSON object’s Values

Above is some information about my car. As you can see, I am not the best driver.

I have caused a few accidents.

Please update this driving record so that I can feel better about my driving skills.

```
var myCar = {
make: ‘Bugatti’,
model: ‘Bugatti La Voiture Noire’,
year: 2019,
accidents: [
{
date: ‘3/15/2019’,
damage_points: ‘5000’,
atFaultForAccident: true
```

```
},
{
date: '7/4/2022',
damage_points: '2200',
atFaultForAccident: true
},
{
date: '6/22/2021',
damage_points: '7900',
atFaultForAccident: true
}
]
}
```

1. Loop over the accidents array. Change atFaultForAccident from true to false.

```
for(var i in myCar.accidents)
{
        mycar.accidents[i].atFaultForAccident =  false;  }
```

2. Print the date of my accidents

```
for(var i in myCar.accidents)
  {
  console.log(myCar.accidents[i].date)
  }
```

Problem 1 (5 mins):
Parsing an JSON object's Values:

Write a function called "printAllValues" which returns an newArray of all the input object's values.

Input (Object):
var object = {name: "RajiniKanth", age: 33, hasPets : false};

Output:
["RajiniKanth", 33, false]

Sample Function proto:

```
var obj = {name : “RajiniKanth”, age : 33, hasPets : false};
function printAllValues(obj) {
  console.log(Object.values(obj));
}
printAllValues();
```

Problem 2(5 mins) :
Parsing an JSON object’s Keys:

Write a function called “printAllKeys” which returns an newArray of all the input object’s keys.

Example Input:
{name : ‘RajiniKanth’, age : 25, hasPets : true}

Example Output:
[‘name’, ‘age’, ‘hasPets’]

Sample Function proto:

```
function printAllKeys(obj) {
   console.log(Object.keys(obj));
 }
printAllKeys();
```

Problem 3(7–9 mins):

Parsing an JSON object and convert it to a list:

Write a function called “convertObjectToList” which converts an object literal into an array of arrays.

Input (Object):
var object = {name: “ISRO”, age: 35, role: “Scientist”};

Output:
[[“name”, “ISRO”], [“age”, 35], [“role”, “Scientist”]]

Sample Function proto:

```
var obj = {name: “ISRO”, age: 35, role: “Scientist”};
function convertListToObject(obj) {
    console.log(Object.entries(object));
}
convertListToObject();
```

Problem 4(5 mins):

Parsing a list and transform the first and last elements of it:
Write a function ‘transformFirstAndLast’ that takes in an array, and returns an object with:
1) the first element of the array as the object’s key, and
2) the last element of the array as that key’s value.

Input (Array):
var array = [“GUVI”, “I”, “am”, “Geek”];

Output:
var object = {
GUVI : “Geek”
}

Sample Function proto:

```
var arr = [“GUVI”, “I”, “am”, “a geek”];
function transformFirstAndLast(arr) {
let newObject = {};
    let arrLength = arr.length;
    newObject[arr[0]] = arr[arrLength-1]

return newObject;
```

```
}
console.log(transformFirstAndLast(array));
```

Problem 5 (7 -9 mins):
Parsing a list of lists and convert into a JSON object:

Write a function “fromListToObject” which takes in an array of arrays, and returns an object with each pair of elements in the array as a key-value pair.

Input (Array):
var array = [[“make”, “Ford”], [“model”, “Mustang”], [“year”, 1964]];

Output:
var object = {
make : “Ford”
model : “Mustang”,
year : 1964
}

Sample Function proto:

```
var arr = [[“make”, “Ford”], [“model”, “Mustang”], [“year”, 1964]];
function fromListToObject(arr) {
var newObject = {};
for(let i in arr){
      newObject[arr[i][0]] = arr[i][1];
   }

return newObject;
}
```

Problem 6 (10 mins):
Parsing a list of lists and convert into a JSON object:

Write a function called “transformGeekData” that transforms some set of data from one format to another.

Input (Array):

var array = [[[“firstName”, “Vasanth”], [“lastName”, “Raja”], [“age”, 24], [“role”, “JSWizard”]], [[“firstName”, “Sri”], [“lastName”, “Devi”], [“age”, 28], [“role”, “Coder”]]];

Output:

[
{firstName: “Vasanth”, lastName: “Raja”, age: 24, role: “JSWizard”},
{firstName: “Sri”, lastName: “Devi”, age: 28, role: “Coder”}
]

Sample Function proto:

```
var arr= [[[“firstName”, “Vasanth”], [“lastName”, “Raja”], [“age”, 24], [“role”, “JSWizard”]], [[“firstName”, “Sri”], [“lastName”, “Devi”], [“age”, 28], [“role”, “Coder”]]];
function transformEmployeeData(arr) {
var tranformEmployeeList = [];

//Your code
for(let i=0; i<arr.length; i++)
{
        transformEmployeeList[i] = {};
       for(let j=o; j<arr[i].length; j++)
        transformEmployeeList[i][arr[i][j][0]] = arr[i][j][1];
}

return tranformEmployeeList;
}
console.log(transformEmployeeData(arr));
```

Problem 7 (10 — 20 mins):

Parsing two JSON objects and Compare:

Write an “assertObjectsEqual” function from scratch.

Assume that the objects in question contain only scalar values (i.e., simple values like strings or numbers).

It is OK to use JSON.stringify().

Note: The examples below represent different use cases for the same test. In practice, you should never have multiple tests with the same name.

Success Case:

Input:

```
var expected = {foo: 5, bar: 6};
var actual = {foo: 5, bar: 6}
assertObjectsEqual(actual, expected, 'detects that two objects are equal');
```

Output:

Passed

Failure Case:

Input:

```
var expected = {foo: 6, bar: 5};
var actual = {foo: 5, bar: 6}
assertObjectsEqual(actual, expected, 'detects that two objects are equal');
```

Output:

FAILED [my test] Expected {"foo":6,"bar":5}, but got {"foo":5,"bar":6}

```
var expected = {foo: 5, bar: 6};
var actual = {foo: 5, bar: 6}

function assertObjectsEqual(actual, expected, testName){
// your code here
let actualString = JSON.stringify(actual);
let expectedString = JSON.stringify(expected);
if(actualString != expectedString) {
        console.log("Failed [" +testName+ "] Expected" + expectedString + "but
got" + actualString);
}
else
    testName = console.log("Passed");
return testName;
}
assertObjectsEqual(actual,expected, 'detects that two object are equal');
```

Problem 8(10 mins):

Parsing JSON objects and Compare:

I have a mock data of security Questions and Answers. You function should take the object and a pair of strings and should return if the quest is present and if its valid answer

```
var securityQuestions = [
{
question: “What was your first pet’s name?”,
expectedAnswer: “FlufferNutter”
},
{
question: “What was the model year of your first car?”,
expectedAnswer: “1985”
},
{
question: “What city were you born in?”,
expectedAnswer: “NYC”
}
]
function chksecurityQuestions(securityQuestions,question,ans) {

// your code here
let output = false;
for(let i=0; i<securityQuestions.Length; i++) {
    if (securityQuestions[i].questions === question) {
        if (securityQuestions[i].expectedAnswer === ans) {
            output = true;
}
}
}
return output;
}


//Test case1:
```

```
var ques = “What was your first pet’s name?”;
var ans  =  “FlufferNutter”;
var status = chksecurityQuestions(securityQuestions, ques, ans);
console.log(status); // true
//Test case2:
var ques = “What was your first pet’s name?”;
var ans  =  “DufferNutter”;
var status = chksecurityQuestions(securityQuestions, ques, ans);
console.log(status); // flase
```

Problem 9(20 mins):

Parsing JSON objects and Compare:

Write a function to return the list of characters below 20 age

```
var students = [
{
name: “Siddharth Abhimanyu”, age: 21}, { name: “Malar”, age: 25},
{name: “Maari”,age: 18},{name: “Bhallala Deva”,age: 17},
{name: “Baahubali”,age: 16},{name: “AAK chandran”,age: 23},   {name:“Gabbar Singh”,age: 33},{name: “Mogambo”,age: 53},
{name: “Munnabhai”,age: 40},{name: “Sher Khan”,age: 20},
{name: “Chulbul Pandey”,age: 19},{name: “Anthony”,age: 28},
{name: “Devdas”,age: 56}
];
function returnMinors(arr)
{
        for(let i in students)
        {
        if(students[i].age<=20)
        {
        console.log(students[i].name);
        }
        }
}
console.log(returnMinors(students));
```

5. Try the rest countries api. Extract and print the total population of all the countries in the console. use the html template.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>sample api data</title>
</head>
<body>
<script src = "script.js"></script>
</body>
</html>
```

Script.js

```
var request = new XMLHttpRequest();
request.open('GET','https://restcountries.eu/rest/v2/all',
true);
request.send();

request.onload = function (){
    var data = JSON.parse(this.response);
    var totalPopulation = data.reduce((acc, item) => {
        return acc + item.population;
    }, 0)
    console.log(totalPopulation);
}
```

OUTPUT: 7349137231