TASK-21ST MARCH

A) .Generate an API key with given link below https://openweathermap.org/guide .Print the current weather data in console- By lat lang

Index.html file

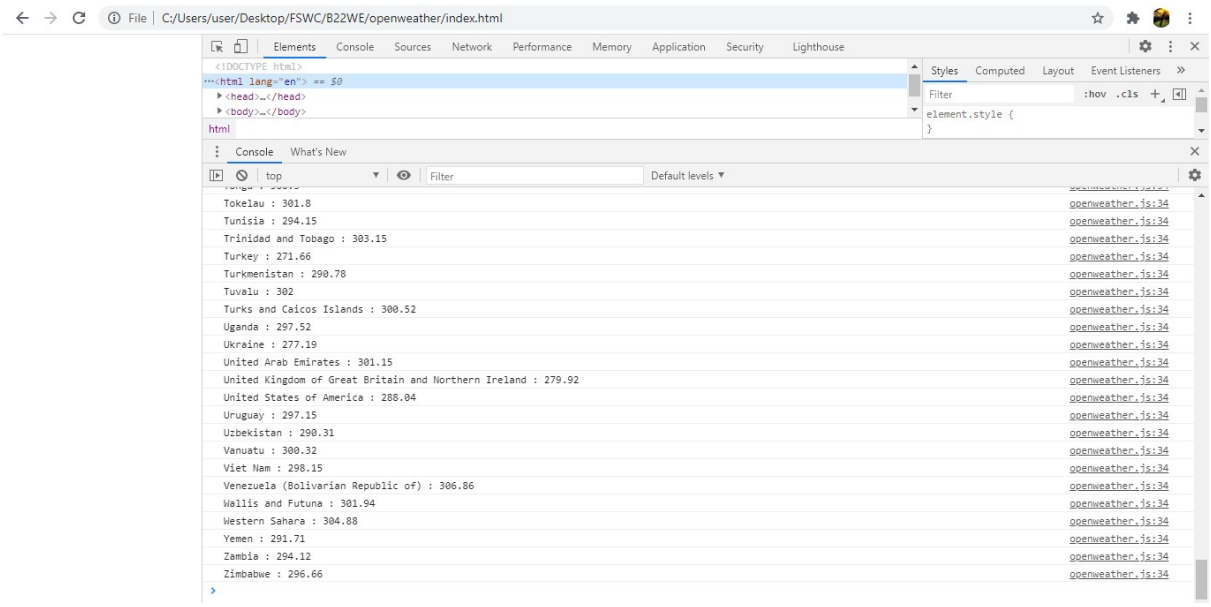
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script src="openweather.js"></script>
</body>
</html>
```

Openweather.js file

```
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 (i in countryData) {
        try {
            var countryName = countryData[i].name;
            var latLong = countryData[i].latlng;
            if (latLong.length === 0) throw new Error('Lat Long not fou
nd');

            weatherData(countryName, ...latLong);
        } catch (e) {
            console.log('Invalid co-
ordinate data for country: ' + countryName + ' ' + e.message);
        }
    }
};
var weatherData = function (name, lat, lng) {
    var apiKey = 'bf5888409bd78bafc10a05ae206d012a';
```

```
    var URL = `https://api.openweathermap.org/data/2.5/weather?lat=${la
t}&lon=${lng}&appid=${apiKey}`;

    var requestWeatherData = new XMLHttpRequest();

    requestWeatherData.open('GET', URL, true);

    requestWeatherData.send();

    requestWeatherData.onload = function () {
                try {
                    var WeatherData = JSON.parse(this.response);
                    console.log(`${name} : ${WeatherData.main.temp}`);
                } catch (e) {
                    console.log('Invalid response from API for ' + name
);
                }
            };
        };
```

Output :



B). **Do the below programs in anonymous function and IIFE**

IIFE:

1. To print odd numbers in an array using IIFE (immediately invoked function expression)

```
const readline = require('readline');
const inp = readline.createInterface({
  input: process.stdin
});
const userInput = [];
inp.on("line", (data) => {
  userInput.push(data);});
inp.on("close", () =>{
 //start-here
 //To print odd numbers in an array using IIFE (immediately invoked function expression)
(function() {
var a=userInput[0]
var arr=a.split(" ")
for(i=0;i<arr.length;i++)
 if(arr[i]%2!==0)
console.log(arr[i])
})();
//end here
});
```

2. **Convert all the strings to title caps in a string array IIFE (immediately invoked function expression)**

```
const readline = require('readline');

const inp = readline.createInterface({

  input: process.stdin

});

const userInput = [];

inp.on("line", (data) => {

  userInput.push(data);});

inp.on("close", () =>{

 //start-here

 //Convert all the strings to title caps in a string array  using IIFE (immediately invoked function expression)

(function () {

str=userInput[0]

str = str.toLowerCase().split(' ');

for (var i = 0; i < str.length; i++) {

        str[i] = str[i].charAt(0).toUpperCase() + str[i].slice(1);

}

y= str.join(' ');

console.log(y)

})()

//end here

});
```

3. **Sum of all numbers in an array using IIFE (immediately invoked function expression)**

```
const readline = require('readline');
const inp = readline.createInterface({
  input: process.stdin
});
const userInput = [];
inp.on("line", (data) => {
  userInput.push(data);});
inp.on("close", () =>{
 //start-here
 //Sum of all numbers in an array  using IIFE (immediately invoked function expression)
(function () {
a=userInput[0]
arr=a.split(" ")
sum=0
for(i=0;i<arr.length;i++)
{
sum=sum+parseInt(arr[i])
}
console.log(sum)
})()
//end here
});
```

4. to return all the prime numbers in an array using IIFE

```
const readline = require('readline');
const inp = readline.createInterface({
  input: process.stdin
});
const userInput = [];
inp.on("line", (data) => {
  userInput.push(data);});
inp.on("close", () =>{
  //start-here
  //to return all the prime numbers in an array using IIFE
(function(){
var numArray = [2, 3, 4, 5, 6, 7, 8, 9, 10,11]
numArray = numArray.filter((number) => {
  for (var i = 2; i <= Math.sqrt(number); i++)
  {
    if (number % i === 0)
    return false;
  }
  return true;
});
console.log(numArray);
})();
});
```

5**. to return all the palindromes in an array using IIFE**

```
const readline = require('readline');
const inp = readline.createInterface({
  input: process.stdin
});
const userInput = [];
inp.on("line", (data) => {
  userInput.push(data);});
inp.on("close", () =>{
  //start-here//to return all the palindromes  in an array using IIFE
(function(){
const arr = ['carecar', 1344, 12321, 'did', 'cannot'];
const isPalindrome = el => {
   const str = String(el);
   let i = 0;
   let j = str.length - 1;
   while(i < j) {
      if(str[i] === str[j]) {
         i++;
         j--;
      }
      else {
         return false;
      }
   }
   return true;
};
```

```
const findPalindrome = arr => {
  return arr.filter(el => isPalindrome(el));
};
console.log(findPalindrome(arr));
})();
});
```

6. **Return median of two sorted arrays of same size using IIFE**

https://www.youtube.com/watch?v=H4gYNnS8kfE

```
const readline = require('readline');
const inp = readline.createInterface({
 input: process.stdin
});
const userInput = [];
inp.on("line", (data) => {
 userInput.push(data);});
inp.on("close", () =>{
 //start-here//Return median of two sorted arrays of same size  using IIFE
(function(){
//Input: nums1 = [1,3], nums2 = [2]
//Output: 2.00000
//Questions:
// what is a median ?
// best time complexity ?
//test cases
 // nums1 = [1,3,34,90], nums2 = []
 // nums1 = []. nums2 = []
function bruteForce(nums1 = [], nums2 = []) {
```

```
  const mergedNums = [...nums1, ...nums2].sort((a, b) => a - b);
  const midPoint = Math.floor(mergedNums.length / 2);
  return mergedNums.length % 2 !== 0
    ? mergedNums[midPoint]
    : (mergedNums[midPoint - 1] + mergedNums[midPoint]) / 2;
}
// merge sort
// complexity O(n + m)
function findMedianSortedArrays(nums1 = [], nums2 = []) {
  let i1 = 0;
  let i2 = 0;
  const len1 = nums1.length;
  const len2 = nums2.length;
  const len = len1 + len2;
  if (len === 0) {
    return null;
  }
  const merged = [];
  while (i1 < len1 && i2 < len2) {
    if (nums1[i1] <= nums2[i2]) {
      merged.push(nums1[i1++]);
    } else {
      merged.push(nums2[i2++]);
    }
  }
  while (i1 < len1) {
    merged.push(nums1[i1++]);
```

```
  }
  while (i2 < len2) {
    merged.push(nums2[i2++]);
  }
  const isOdd = len % 2;
  if (isOdd) {
    return merged[(len - 1) / 2];
  } else {
    return (merged[merged.length / 2] + merged[merged.length / 2 - 1]) / 2;
  }
}
const nums1 = [1,3];
const nums2 = [2];
console.log(findMedianSortedArrays(nums1, nums2))
})();
});
```

7. To Remove duplicates from an array using IIFE

```
const readline = require('readline');
const inp = readline.createInterface({
  input: process.stdin
});
const userInput = [];
inp.on("line", (data) => {
  userInput.push(data);});
inp.on("close", () =>{
  //start-here//to Remove duplicates from an array using IIFE
(function(){
```

```
function getUnique(arr){
   let uniqueArr = [];
   // loop through array
   for(let i of arr) {
      if(uniqueArr.indexOf(i) === -1) {
         uniqueArr.push(i);
      }
   }
   console.log(uniqueArr);
}
const array = [1, 2, 3, 2, 3];
// calling the function
// passing array argument
getUnique(array);
})();
});
```

8. Rotate an array by k times and return the rotated array using IIFE.

```
 const readline = require('readline');
const inp = readline.createInterface({
  input: process.stdin
});
const userInput = [];
inp.on("line", (data) => {
  userInput.push(data);});
inp.on("close", () =>{
  //start-here
(function(){
```

```
function rotLeft(arr, rotations) {
  const rotatedArray = arr.concat();
  for (let i = 0; i < rotations; i++) {
    const frontItem = rotatedArray.shift();
    rotatedArray.push(frontItem);
  }
  return rotatedArray;
}
const numRotation = 4;
const sampleArray = [1, 2, 3, 4, 5];
console.log(rotLeft(sampleArray, numRotation));
})();
});
```

Anonymous:

1. Print odd numbers in an array

```
const readline = require('readline');
const inp = readline.createInterface({
  input: process.stdin
});
const userInput = [];
inp.on("line", (data) => {
  userInput.push(data);});
inp.on("close", () =>{
  //start-here
var anon = function() {
  var a=userInput[0]
var arr=a.split(" ")
```

```
for(i=0;i<arr.length;i++)
 if(arr[i]%2!==0)
console.log(arr[i])
}
anon();
});
```

2. Convert all the strings to title caps in a string array

```
const readline = require('readline');
const inp = readline.createInterface({
  input: process.stdin
});
const userInput = [];
inp.on("line", (data) => {
  userInput.push(data);});
inp.on("close", () =>{
  //start-here// Anonnymous
var anon = function() {
 str=userInput[0]
str = str.toLowerCase().split(' ');
for (var i = 0; i < str.length; i++) {
        str[i] = str[i].charAt(0).toUpperCase() + str[i].slice(1);
}
y= str.join(' ');
console.log(y)
}
anon();
});
```

3. Sum of all numbers in an array

```
const readline = require('readline');
const inp = readline.createInterface({
  input: process.stdin
});
const userInput = [];
inp.on("line", (data) => {
  userInput.push(data);});
inp.on("close", () =>{
  //start-here// Anonnymous
var anon = function() {
a=userInput[0]
arr=a.split(" ")
sum=0
for(i=0;i<arr.length;i++)
{
sum=sum+parseInt(arr[i])
}
console.log(sum)
}
anon();
});
```

4. Return all the prime numbers in an array

```
const readline = require('readline');
const inp = readline.createInterface({
  input: process.stdin
});
```

```
const userInput = [];
inp.on("line", (data) => {
  userInput.push(data);});
inp.on("close", () =>{
  //start-here// Anonnymous
var anon = function() {
var numArray = [2, 3, 4, 5, 6, 7, 8, 9, 10,11]
numArray = numArray.filter((number) => {
  for (var i = 2; i <= Math.sqrt(number); i++)
  {
    if (number % i === 0)
    return false;
  }
  return true;
});
console.log(numArray);
}
anon();
});
```

5. Return all the palindromes in an array

```
const readline = require('readline');
const inp = readline.createInterface({
  input: process.stdin
});
const userInput = [];
inp.on("line", (data) => {
  userInput.push(data);});
```

```
inp.on("close", () =>{
  //start-here// Anonnymous
var anon = function() {
const arr = ['carecar', 1344, 12321, 'did', 'cannot'];
const isPalindrome = el => {
  const str = String(el);
  let i = 0;
  let j = str.length - 1;
  while(i < j) {
    if(str[i] === str[j]) {
      i++;
      j--;
    }
    else {
      return false;
    }
  }
  return true;
};
const findPalindrome = arr => {
  return arr.filter(el => isPalindrome(el));
};
console.log(findPalindrome(arr));
}
anon();
});
```

6. Return median of two sorted arrays of same size

```
const readline = require('readline');
const inp = readline.createInterface({
  input: process.stdin
});
const userInput = [];
inp.on("line", (data) => {
  userInput.push(data);});
inp.on("close", () =>{
  //start-here// Anonnymous
var anon = function() {
function bruteForce(nums1 = [], nums2 = []) {
  const mergedNums = [...nums1, ...nums2].sort((a, b) => a - b);
  const midPoint = Math.floor(mergedNums.length / 2);
  return mergedNums.length % 2 !== 0
    ? mergedNums[midPoint]
    : (mergedNums[midPoint - 1] + mergedNums[midPoint]) / 2;
}
// merge sort
// complexity O(n + m)
function findMedianSortedArrays(nums1 = [], nums2 = []) {
  let i1 = 0;
  let i2 = 0;
  const len1 = nums1.length;
  const len2 = nums2.length;
  const len = len1 + len2;
  if (len === 0) {
```

```
    return null;
  }
  const merged = [];
  while (i1 < len1 && i2 < len2) {
    if (nums1[i1] <= nums2[i2]) {
      merged.push(nums1[i1++]);
    } else {
      merged.push(nums2[i2++]);
    }
  }
  while (i1 < len1) {
    merged.push(nums1[i1++]);
  }
  while (i2 < len2) {
    merged.push(nums2[i2++]);
  }
  const isOdd = len % 2;
  if (isOdd) {
    return merged[(len - 1) / 2];
  } else {
    return (merged[merged.length / 2] + merged[merged.length / 2 - 1]) / 2;
  }
}
const nums1 = [1,3];
const nums2 = [2];
console.log(findMedianSortedArrays(nums1, nums2))
}
```

```
anon();
});
```

7. Remove duplicates from an array

```
const readline = require('readline');
const inp = readline.createInterface({
  input: process.stdin
});
const userInput = [];
inp.on("line", (data) => {
  userInput.push(data);});
inp.on("close", () =>{
  //start-here// Anonnymous
var anon = function() {function getUnique(arr){
    let uniqueArr = [];
    // loop through array
    for(let i of arr) {
        if(uniqueArr.indexOf(i) === -1) {
            uniqueArr.push(i);
        }
    }
    console.log(uniqueArr);
}
const array = [1, 2, 3, 2, 3];
// calling the function
// passing array argument
getUnique(array);
}
```

```
anon();
});
```

8. Rotate an array by k times and return the rotated array.

```
const readline = require('readline');
const inp = readline.createInterface({
  input: process.stdin
});
const userInput = [];
inp.on("line", (data) => {
  userInput.push(data);});
inp.on("close", () =>{
  //start-here
var anon = function() {
    function rotLeft(arr, rotations) {
  const rotatedArray = arr.concat();
  for (let i = 0; i < rotations; i++) {
    const frontItem = rotatedArray.shift();
    rotatedArray.push(frontItem);
  }
  return rotatedArray;
}
const numRotation = 4;
const sampleArray = [1, 2, 3, 4, 5];
console.log(rotLeft(sampleArray, numRotation));
}
anon();
});
```