**Question-1**Write a function which combines an array of objects, grouped by a key you provide (this key  
will correspond to a key found in the objects. The function will index the new object with the  
value of those keys.

**Answer :**

const users = [

    {

        id: 1,

        name: 'brain',

    },

    {

        id: 2,

        name: 'john',

    },

    {

        id: 3,

        name: 'brain',

        age: 30

    },

    {

        id: 4,

        age: 40

    }

]

function addObject(dictObj, key, val)

{

    if(dictObj[key] == null) {

        dictObj[key] = [];

    }

    dictObj[key].push(val);

}

function groupByKey(objects) {

    var groupedObj = new Object();

    for (var obj of objects) {

        var objName = obj.name || 'no\_name';

        addObject(groupedObj, objName, obj);

    }

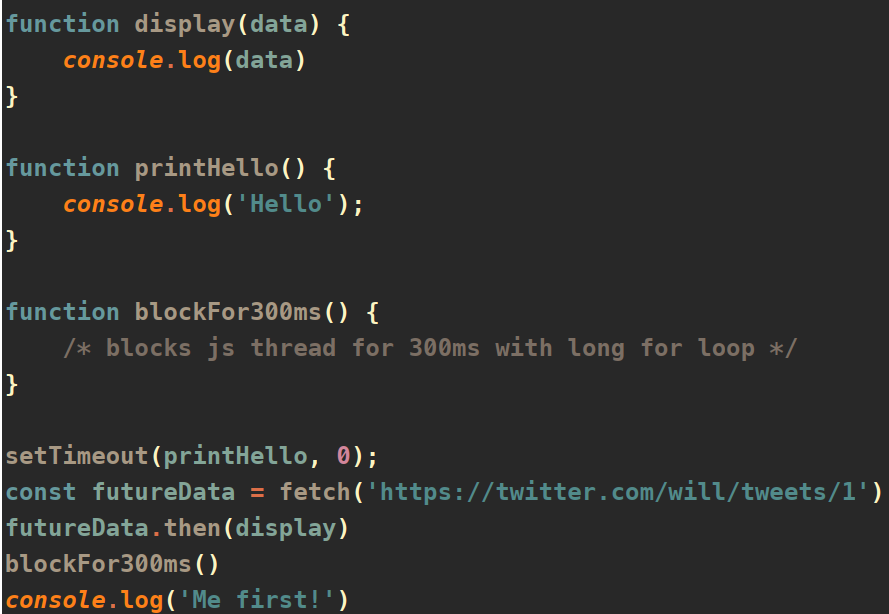
    return groupedObj;

}

var retVal = groupByKey(users);

console.log(retVal);

**Question-2**



è Which will run first? Write displays in correct order of execution

**Answer :**

blockFor300ms() will run first as it is an asynchronous function and it will block the execution of other worker threads until completion.

Execution time: 300ms

Me first!

Hello

Response {size: 0, timeout: 0, Symbol(Body internals): Object, Symbol(Response internals): Object}

**Question-3**: Write a function that takes any string argument and returns it anagrams.

**Answer:**

// Reference : Heap's Algorithm

// https://en.wikipedia.org/wiki/Heap%27s\_algorithm

function generateAnagram(str) {

    var anagramList = [];

    function swap(strArr, i, j) {

        var temp = strArr[i];

        strArr[i] = strArr[j];

        strArr[j] = temp;

    }

    function generate(strArr, k = strArr.length) {

        if(k == 1) {

            // console.log(strArr + " | ");

            anagramList.push(strArr.join(''));

        }

        else {

            // Recursively call once for each k

            for(var i = 0; i < k; i += 1) {

                generate(strArr, k-1);

                // avoid swap when i==k-1

                if (i < k - 1) {

                    // swap choice dependent on parity of k

                    swap(strArr, (k%2 == 0) ? i:0, k-1);

                }

            }

        }

    }

    generate(str.split(''));

    return [...new Set(anagramList)];

}

console.log(generateAnagram('abc'));

**Question-4:** Modify the Number Object in JS, so that following output is possible.  
***console*.log(...8) // 0 1 2 3 4 5 6 7 8**

**Answer:**

Number.prototype[Symbol.iterator] = function\* () {

    const num = this;

    if( num < 0) {

        // if number is -ve yield 0

        yield 0;

    }

    else {

        // +ve number

        for (let i = 0; i <= num; i++) {

            yield i;

        }

    }

};

console.log(...8);

console.log(...0);

console.log(...-8);

**Question-5:** How does the “this” keyword work? Provide some code examples

**Answer:**

'this' keyword refers to the object it belongs to. Within a method, 'this' refers to the owner object.

var helloWorld = {

    greetings: 'Hello',

    print : function(name) {

        console.log(this.greetings + ' ' + name + '!!!');

    }

};

helloWorld.print('Question\_5');

**Question-6:** Write a function or functions so following can be achieved  
***console*.log(addAll([add5(1), multiply4(2), divide2(3)]))  
result: 15.5**

**Answer:**

let add5 = new Function('a', 'return a+5');

let multiply4 = new Function('a', 'return a\*4');

let divide2 = new Function('a', 'return a/2');

function addAll(fnArr) {

    let sum = 0

    for( let fn of fnArr) {

        sum += fn;

    }

    return sum;

}

console.log(addAll([add5(1), multiply4(2), divide2(3)]));

**Question-7:** Write a recursive function that performs a binary search

**Answer:**

function binarySearch(arr, item, lt=0, rt=arr.length) {

    // check if the left indexed element == item

    if(lt==rt && arr[lt] == item) {

        return lt;

    }

    if(lt<rt) {

        mid = Math.ceil((lt+rt)/2);

        // Check if the middle element == item

        if(arr[mid] == item) {

            return mid;

        }

        if(arr[mid] > item) {

            // item is smaller than middle element, parse left subarray

            rt = mid-1;

        } else {

            // item is greater than middle element, parse right subarray

            lt = mid;

        }

        return binarySearch(arr, item, lt, rt);

    }

    return -1;

}

var arr = [2, 3, 4, 10, 40];

console.log(binarySearch(arr,2));

console.log(binarySearch(arr,3));

console.log(binarySearch(arr,4));

console.log(binarySearch(arr,10));

console.log(binarySearch(arr,40));

console.log(binarySearch(arr,50));

**Question-8:** What would be the unique triplets in an array, that all add up to zero, where  
the array is of integers? Your answer should not include any triplets. What would be the big  
o of the function?

**Answer:**

**O(n) =**

function findTriplets(arr)

{

    var tempArr = arr.sort();

    var resultSet = {};

    var arrLen = arr.length;

    for (var i=0; i<arrLen-1; i+=1 ) {

        // Set to hold the third number which can form the triplet

        var thirdNumSet = new Set();

        for (var j=i+1; j<arrLen; j++) {

            // add two numbers

            var sumOfTwoNum = (tempArr[i] + tempArr[j]) \* -1;

            // Check if the sum of two numbers is present in the third number set

            if (thirdNumSet.has(sumOfTwoNum)) {

                // triplet

                let triplet = [tempArr[i],sumOfTwoNum,tempArr[j]];

                // Add triplet if not exists already in the dictionary

                if( resultSet[triplet.toString()] == null ) {

                    resultSet[triplet.toString()] = triplet;

                }

            }

            else {

                // add the second number to third number set for future reference.

                thirdNumSet.add(tempArr[j]);

            }

        }

    }

    return Object.values(resultSet);

}

// 'i' loop traverses from 0 to n-2.

// 'j' loop traverses from i to n-1

// total number of executions in the loop = n\*(n-1)

// Hence O(n) = n^2

arr = [-1, 0, 1, 1, -1, 0]

console.log(JSON.stringify(findTriplets(arr)));

**Question-9:** Explain this JS class implementation, what new keyword automate behind the  
scene?



**Answer:**

‘new’ keyword creates a new User object and invokes the constructor with values ‘Eva’ and ‘9’.

The values passed to the constructor is assigned to the Object members ‘name’ and ‘score.

**Question-10:** Provide a high-level design to build a web application for a cloud-based media workflow management system, where user can encode an mp4 file to stream to an  
RTMP destination.

Workflow diagram is the preferred solution.  
**Answer:**

A close up of a map

Description automatically generated