JS

Coding Basics

This page contains basic problems solved with JavaScript code.

For each problem solutions are given in the folders with test scripts.

Install Unit Test Package

Jest is used by Facebook to test all JavaScript code including React applications.

To know more about jest refer this link Jest Webpage.

```
yarn global add jest
or
npm install --save-dev jest
```

Run Jest

Command to execute the test file.

```
jest test.js
```

Example Output :

```
PASS  reversestring/test.js
  ✓ Reverse function exists (3ms)
  ✓ Reverse reverses a string (1ms)
  ✓ Reverse reverses a string
```

```
Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        2.469s
```

Debugger

File Name : index.js

```
function reverse(str){
     debugger;
    return str.split('').reduce((rev,char)=>
        char+rev , '');
}
reverse('qwsde');
module.exports = reverse;
}
```

To debug the above program, go to terminal and run the below command.

```
    node inspect index.js
    Press c
    repl
```

Type variable name to check the value of a variable or type a statement to run.

```
    return str.split('').reduce((rev,char)=>
        char+rev , '');
```

Reverse a string program

To test this concept we wrote a string reverse function, first get the string to be reversed, we convert the string into array, reverse it and convert back into string.

```
function reverse(str) {

    const arr = str.split(''); // convert the string
into array.
    arr.reverse(); // Reverse the string.
    return arr.join(''); // Join the string and
return the string.

}

function reverse(str) {
    return str
         .split('')
         .reverse()
         .join('');
}

function reverse(str) {
 let reversed = '';
    for(let characters of str)
    {
        reversed = characters + reversed;
    }

    return reversed;
}
function reverse(str) {
  return str.split('').reduce((rev,char)=>
        char+rev,'');
}
```

Check the given string is a Palindrome

```
function palindrome(str) {

   return str ===
str.split('').reduce((rev,char)=>char+rev,'');

}
```

Using every prototype

```
function palindrome(str) {
      return str.split('').every( (char,i) => {
        return char === str[str.length-1-i];
    });
}
```

Reverse an Integer

```
function reverseInt(n) {

    return
Math.sign(n)*parseInt(n.toString().split('').reduce((
num,rev) => rev+num,''));

}
```

Find maximum repeated characted in a string

```
function maxChar(str) {

    chars = {};
    max = 0;

    for(let char of str){
        chars[char] = chars[char] + 1 || 1;
    }

    for(let char in chars)
    {
        if(chars[char] > max)
        {
            max = char;
        }
    }
    return max;
}
```

FizzBuzz program

```
function fizzBuzz(n) {

    for (let i = 1; i <= n ; i++){
        if(i%3 == 0 && i%5 !=0 )
        {
            console.log('fizz');
        }
        else if (i % 3 != 0 && i % 5 == 0 )
        {
            console.log('buzz');
        }
        else if (i % 3 == 0 && i % 5 == 0) {
            console.log('fizzbuzz');
        }
        else{
            console.log(i);
        }
    }

}
```

Split an array into chunks

```
function chunk(array, size) {

    var chunk = [];
    for(let i= 0 ; array.length > 0;i++)
    {
        chunk[i] =  array.splice(0,size);
    }
    return chunk;
}
```

Anagrams

Check to see if two provided strings are anagrams of eachother.

```
function anagrams(stringA, stringB) {

    if (stringA.replace(/[^\w]/g, "
").toLowerCase().split('').sort().join('') ===
stringB.replace(/[^\w]/g, "
").toLowerCase().split('').sort().join(''))
    {
        return true;
    }
    return false;

}
```

Capitalize

Function to capitalize the first letter of each word in the string

```
function capitalize(str) {
    let result = str[0].toUpperCase();

    for(let i = 1 ; i < str.length ; i++ )
    {
        if(str[i-1] === ' ')
        {
            result += str[i].toUpperCase();
        }
        else
        {
            result += str[i];
        }
    }

    return result;
}
```

Steps

The function should console log a step shape with N levels using the # character

Example : steps(4) '# ' '## ' '### ' '####'

Recursive function

```
function steps(n, row = 0, stairs = ''){

    if(n === row)
    {
        return;
    }
    if(n === stairs.length)
    {
        console.log(stairs);
        steps(n, row + 1);
        return;
    }

    if(stairs.length <= row){
            stairs += '#';
    }
    else{
            stairs += ' ';
        }
    steps(n,row,stairs);
}
```

```
function steps(n) {

    for(let row = 0 ; row < n ; row++ )
    {
        let stairs = '';
        for (let column = 0; column < n; column++)
        {
            if (column <= row)
            {
               stairs += '#';
            }
            else
            {
                stairs += ' ';
            }
        }
        console.log(stairs);
    }
}
```

Pyramid

Print a pyramid using recursive function

```
function pyramid(n, row = 0, block = '') {
let base = (n * 2) - 1;
let middle = Math.floor(base/2);
if(n === row)
{
    return;
}
if(base === block.length)
{
    console.log(block);
    pyramid(n,row + 1);
    return;
}
if (block.length >= middle - row && block.length <=
middle + row)
{
    block += '#';
}
else{
    block += ' ';
}
    pyramid(n,row,block);
}
```

Vowels

Find Number of vowels in a given string

Iterative Solution

```
function vowels(str) {

    let result = 0;
    const arr = str.toLowerCase().split('');
    for (let char of arr) {
        if (char === 'a' || char === 'e' || char ===
'i' || char === 'o' || char === 'u') {
            result++;
        }
    }
    return result;
}
```

Iterative Solution with includes Function

```
function vowels(str){

    const vowels = 'aeiou';
    let result = 0;
    for (let char of str.toLowerCase()) {
       if(vowels.includes(char))
        {
            result++;
        }
    }
    return result;

}
```

RegEx solution

```
function vowels(str){
    const matches = str.match(/[aeiou]/gi);
    return matches ? matches.length : 0 ;
}
```

Spiral Matrix

Function that accepts an integer N and returns a NxN spiral matrix.

```
function matrix(n) {

    const results = [];
    let counter = 0;
    let sr = sc = 0;
    let er = ec = n-1;
    for(let i = 0; i < n ; i++)
    {
        results.push([]);
    }

    while(sc <= ec && sr <= er)
    {
        for(let i = sr ; i <= er ; i++)
        {
            results[sr][i] = ++counter;
        }
        sr++;
        for(let i = sr ; i <= er ; i++)
        {
            results[i][ec] = ++counter;
        }
        ec--;
        for(let i =  ec; i >= sc ; i-- )
        {
            results[er][i] = ++counter;
        }
        er--;
        for(let i = er ; i >= sr ; i--)
        {
            results[i][sc] = ++counter;
        }
        sc++;
    }
    return results;
}
```

Fibonacci Series

Print out the n-th entry in the fibonacci series.

Iterative Solution

```
function fib(n) {
     const result = [0, 1];
     for (let i = 2; i <= n; i++) {
         result.push(result[i - 2] + result[i - 1]);
     }
     return result[n];
 }
```

Recursive Function using Memoize method

```
function memoize(fn){
    const cache = {};
    return function(...args){
        if(cache[args]){
            return cache[args];
        }
        const result = fn.apply(this, args);
        cache[args] = result;
        return result;
    }
}
function recfib(n) {
    if(n < 2)
    {
        return n;
    }
    return fib(n-1) + fib(n-2);
}
const fib = memoize(recfib);
```

Data Structures

Queue

```
class Queue {
    constructor(){
        this.data = [];
    }
    add(record){
        this.data.unshift(record);
    }
    remove(){
       return this.data.pop();
    }
}
```