Website development

1.install vscode

2.install extensions live server,html,javascript,python from extensions add emmet

3.install node externally

Open live servers for launching in browser

First step in learning :

console.log("Hello world");

creating arrays in js:

|  |
| --- |
| let arr=[]; |
|  | let arr1 =[]; |
|  | let st = "hello" |
|  | st1= st.split('') |
|  | arr.push(st1) |
|  | //console.log(arr); |
|  |  |
|  | for(let i=arr.lenght -1; i>=0; i--){ |
|  | arr1.push(arr[i]); |
|  | } |
|  | console.log(arr1); |

Exercise 1

Function for creating factorial

function factorial(n){

    var fact = 1

    if(n<=0){

        console.log(0)

    }

    else{

        for(var i=1; i<=n;i++){

               var fact =fact\*i

              }}

    return fact

let,const,var,function

function readLine(){}

we can access any functions by declaring abovementioned syntax

function is a set of statements which are assigned to do specific tasks

let is used to assign a variable it can’t be accessed outside of local block

var is also used to assign a variable it can be accessed outside local block or it can be called as global variable

const is used for assign a constant value or unchangeable value to the characters/strings/variable

exercise 2

 const Pi = Math.PI

    let r = readLine()

    let area = Pi\*r\*r

    let perimeter = 2\*Pi\*r

    console.log(area)

    console.log(perimeter)

the above code is used for finding perimeter and area here all the above mentioned functioned are used

if else statement and for loop:

if (a==0 && a>10){

statement

exec

}

Else {

Statement

Exec

}

For{let i=1;i<=n;i++){

statement exec}

exercise if else:

function getGrade(score) {

    let grade;

    // Write your code here

    if(score>25 && score<=30){

        return 'A';

    }

    else if(score>20 && score<=25){

        return 'B';

    }

    else if(score>15 && score<=20){

        return 'C';

    }

    else if(score>10 && score<=15){

        return 'D';

    }

    else if(score>5 && score<=10){

        return 'E';

    }

    else if(score>=0 && score<=5){

        return 'F';

    }

    return grade;

\*\*remember to add inverted comma’s to character or else you will get characted undefined error

Input: 22

Output:B

Switch statements

Similar to if else statements but it comes handy at complex situations like for below code :

function getLetter(s) {

    let letter;

    // Write your code here

    letter = s.charAt(0);

    switch(letter){

        case 'a','e','i','o','u':

            return 'A';

            break;

        case 'b':

            return 'B';

            break;

        case 'h':

            return 'C';

            break;

        case 'n','z':

            return 'D';

            break;

    }

    return letter;

input: asdkh

output: A

input:bsvhjdg

output: B

check if the give string 3 first letter is in above string print associated character

Includes()

A string function which is used to return Boolean value ,

If a character or element is inside the array or an variable we can use this function if it is there inside or not,

Ex:

Const Sr = ‘arbax’

If(Sr.includes(a)){

Console.log(‘A’)

}

Ex2:

function vowelsAndConsonants(s) {

const vowels = 'aeiou';

var consonants = '';

for(var i = 0; i < s.length-1; i++) {

if (vowels.includes(s[i])) {

console.log(s[i]);

}

else {

consonants += s[i] + '\n';

}

}

console.log(consonants.trim());

}

Arrays in js:

Array are collections of characters,strings,number that can be created by declaring ar =[]

for (let i = 0; i < nums.length; i++) {

if (nums[i] > first) {

second = first;

first = nums[i]

}

if (nums[i] > second && nums[i] < first) {

second = nums[i];

}

}

return second;

join()

join() returns an array as a string:

fruits = [arr,arr1,arr2]

tet= fruits.join()

op: arr arr1 arr2

fruits = [arr,arr1,arr2]

tet= fruits.join(‘a’)

op: arraarr1aarr2a

The **split()** method splits a String object into an array of strings by separating the string into substrings.

The **splice()** method changes the content of an array by removing existing elements and/or adding new elements.

The **join()** method joins all elements of an array into a string

var a = "asdasd|dasd|rttewrtert";var b = a.split('|');  
// ["asdasd", "dasd", "rttewrtert"]

var c = b.splice(1);  
// ["dasd", "rttewrtert"]var d = c.join('');  
// dasdrttewrtert

try catch throw

function myFunction() {

const message = document.getElementById("p01");

message.innerHTML = "";

let x = document.getElementById("demo").value;

try {

if(x == "") throw "empty";

if(isNaN(x)) throw "not a number";

x = Number(x);

if(x < 5) throw "too low";

if(x > 10) throw "too high";

}

catch(err) {

message.innerHTML = "Input is " + err;

}

}

Reverse()

Reversing a string ,returns the string

function reverseString(s) {

    var rarr

    try{

    console.log(s.split("").reverse().join(''));

    }

     catch(e) {

        console.log(e.message);

        console.log(s)

    }

}

Throw: function which throws an string of error when code blows up

function isPositive(a) {

    if(a == '0'){

        throw new Error('Zero Error')

    }

    else if(a < '0'){

        throw new Error('Negative Error')

    }

    return 'YES'

syntax: throw new Error

function main() {

    const n = +(readLine());

    for (let i = 0; i < n; i++) {

        const a = +(readLine());

        try {

            console.log(isPositive(a));

        } catch (e) {

            console.log(e.message);

        }

    }

}

Objects:

In programming to represent real time objects we use objects and classes and sub classes to define that .

Objects can be an real time entity/objects can be a method which has different properties inside it.

Laptop is an object its properties are processor,ram,storage,brand.

In js we represent the objects in below syntax

Const/let/var lap = new laptop(a,b);

Underlined is passing value as parameters for the object

Example:

function Rectangle(a, b) {

    return{

        length:a,

        width:b,

        perimeter:2\*(a+b),

        area:a\*b

    }

}

function main() {

    const a = +(readLine());

    const b = +(readLine());

    const rec = new Rectangle(a, b);

    console.log(rec.length);

    console.log(rec.width);

    console.log(rec.perimeter);

    console.log(rec.area);

}

Input: 2

3

Output:2

3

10

6

Looping inside object

function getCount(objects) {

    let n = 0;

    for(let o of objects) {

        n += (o.x == o.y);

    }

    return n;

}

Another aspect

return objects.filter(function(o){return o.x==o.y}).length

}

Input:

5

1 1

2 3

3 3

3 4

4 5

Output:

2

Classes

Class are defined as set of functions,methods,objects or constructors which is used for reduce ,reuse,flexibility,polymorphism,encapsulate

The code which is used by other function

class Polygon{

    constructor(sides){

        this.sides = sides

    }

    perimeter(){

        let sum = 0

        for(let i=0; i<this.sides.length; i++){

            sum += this.sides[i]

        } return sum

    }

}

const rectangle = new Polygon([10, 20, 10, 20]);

const square = new Polygon([10, 10, 10, 10]);

const pentagon = new Polygon([10, 20, 30, 40, 43]);

console.log(rectangle.perimeter());

console.log(square.perimeter());

console.log(pentagon.perimeter());

the above is rectangle,square,penta using class polygon to calculate the perimeter for all three functions [classes are also called as blueprint]

input is defined as rectangle,square,pentagon

output:

* **60**
* **40**
* **143**

Example of inheritance

class Rectangle {

    constructor(w, h) {

        this.w = w;

        this.h = h;

    }

}

/\*

 \*  Write code that adds an 'area' method to the Rectangle class' prototype

 \*/

Rectangle.prototype.area =function (){

    return this.w \* this.h

}

/\*

 \* Create a Square class that inherits from Rectangle and implement its class constructor

 \*/

class Square extends Rectangle{

    constructor(w){

        super(w,w)

    }

}

if (JSON.stringify(Object.getOwnPropertyNames(Square.prototype)) === JSON.stringify([ 'constructor' ])) {

    const rec = new Rectangle(3, 4); // inputs

    const sqr = new Square(3); // inputs

    console.log(rec.area());

    console.log(sqr.area());

} else {

    console.log(-1);

    console.log(-1);

}

Output:

Expected Output

* **12**
* **9**

Arrow function =>

Arrow functions can convert any anonymous function to working function

Arrow functions were introduced in ES6.

Arrow functions allow us to write shorter function syntax:

Ex:let myFunction = (a, b) => a \* b;

Before:

hello = function() {  
  return "Hello World!";  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_arrow_function1)

With Arrow Function:

hello = () => {  
  return "Hello World!";  
}

example hackerrank:

function modifyArray(nums) {

    return nums.map(ele => ele %2==0 ? ele\*2 : ele\*3 )

}

function main() {

    const n = +(readLine());

    const a = readLine().split(' ').map(Number);

    console.log(modifyArray(a).toString().split(',').join(' '));

}

Input (stdin)

* **5**
* **1 2 3 4 5**

Expected Output

* **3 4 9 8 15**

Regular Expressions

A regular expression is a **pattern** of characters.

The pattern is used to do pattern-matching **"search-and-replace"** functions on text.

In JavaScript, a **RegExp Object** is a pattern with **Properties** and **Methods**.

function regexVar() {

    /\*

     \* Declare a RegExp object variable named 're'

     \* It must match ALL occurrences of numbers in a string.

     \*/

    const re = /\d+/g

    /\*

     \* Do not remove the return statement

     \*/

    return re;

}

Input (stdin)

* **102, 1948948 and 1.3 and 4.5**

Expected Output

* **102**
* **1948948**
* **1**
* **3**
* **4**
* **5**

function regexVar() {

    /\*

     \* Declare a RegExp object variable named 're'

     \* It must match a string that starts and ends with the same vowel (i.e., {a, e, i, o, u})

     \*/

    const re = /^([aeiou]).\*\1$/

    /\*

     \* Do not remove the return statement

     \*/

    return re;

}

Input (stdin)

* **bcd**

Your Output (stdout)

* **false**

Expected Output

* **false**

HTML JS DOM

document.getElementById("demo").innerHTML = "hello";

//html

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" href="css/button.css" type="text/css">

</head>

<body>

<script src="js/button.js" type="text/javascript"></script>

</body>

</html>

//html

//css

.center{

display: flex;

justify-content: center;

align-items: center;

height: 200px;

}

#btn{

padding: 20px 50px;

}

//css

js////

var btn = document.createElement("button");

btn.innerHTML = 0;

btn.id = "btn";

btn.className = "button";

document.body.appendChild(btn);

btn.onclick = function() {

btn.innerHTML = Number(btn.innerHTML) +1;

}

Js///

Javascript executes in execution context

Global execution context creates variable environment(variables) and code(functions) context will be created .

Text

Description automatically generated

Whiteboard

Description automatically generated

Hoisting:

Hoisting as a core concept relies on the way how Execution Context is created. In the first phase i.e. the Memory Allocation Phase all the variables and functions are allocated memory, even before any code is executed. All the variables are assigned undefined at this point in time in the local memory.

1. In JS, before the code is executed, the variables get initialized to undefined.

2. Arrow functions enact as variables and get "undefined" during the memory creation phase while functions actually get run.

3. Hoisting: Mechanism in JS where the variable declarations are moved to the top of the scope before execution. Therefore it is possible to call a function before initializing it.

4. Whenever a JS program is run, a global execution block is created, which comprises of 2: Memory creation and Code execution.

2 Golden Rules:

1. Variable declarations are scanned and are made undefined

2. Function declarations are scanned and are made available

FUNCTIONS AND VARIABLE ENVIRONMENT:

A person sitting in front of a whiteboard with writing on it

Description automatically generated with low confidence

Whenever a function is invoked in JavaScript a functional Execution Context is created and memory is allocated. Once the memory is allocated to the variables and functions, then the code is executed synchronously, one line at a time.

1. We learnt how functions work in JS.

2. At first a global execution context is created, which consists of Memory and code and has 2 phases: Memory allocation phase and code execution phase.

3. In the first phase, the variables are assigned "undefined" while functions have their own code.

4. Whenever there is a function declaration in the code, a separate local execution context gets created having its own phases and is pushed into the call stack.

5. Once the function ends, the EC is removed from the call stack.

6. When the program ends, even the global EC is pulled out of the call stack.

SHORTEST CODE IN JAVASCRIPT

Empty code file is an shortest program in javascript

Global space is an outside space other than function space

JavaScript Engine creates a global object whenever you run any JS code. In the case of browsers, this global object is known as `window`, check out

1. Shortest Program in JS: Empty file. Still, browsers make global EC and global space along with Window object.

2. Global Space: Anything that is not in a function, is in the global space.

3. Variables present in a global space can be accessed by a "window" object. (like window.a)

4. In global space, (this === window) object.

If u try to access or console log variable inside from function , that variable throws an error as undefined .

Text, application

Description automatically generated

**NOT DEFINED AND UNDEFINED**

Not defined – variable that is not defined or declared

Undefined – memory space is reserved but value not assigned

1. Undefined is like a placeholder till a variable is not assigned a value.

2. undefined !== not defined

3. JS- weakly typed language since it doesn't depend on data type declarations.

**LEXICAL-SCOPE**

L;Diagram

Description automatically generated

Lexical environment- from above image , c() sits lexically inside b() where a() sits lexically in global.

Lexical environment – kind of interlinking of function through functions

Lexical environment – local memory +reference to lexical environment to parent, similarly it is created while global execution context.

Whole chain of lexical environment is called scope chain.

Scope chain is exhausted then when variable isn’t found inside the functions.

Let and const declarations are hoisted.