**JavaScript**

........objects.............

let person={

name : 'vishal',

age : 23

}

person.name ='vikas';

person['name'] ='ajith';

person['age']=26;

person.area = 'Narsipatnam'

console.log(person);

...........Arrays..................

let selectedColors = ['red','blue'];

selectedColors[2] = 'green';

console.log(selectedColors);

// Function (Methods) (Performing a task)"Example 1"

function great(firstname,lastname){

let s = firstname+lastname;

console.log("Hello"+" "+s);

}

great('vishal','palla');

// Functions Example 2 (Calculating a value)

function square(num){

return num\*num;

}

console.log(square(2));

.........."OPERATORS".............

........ 'Arthamatic Operators.............

let x =10;

let y = 12;

console.log(x+y);

console.log(x\*y);

console.log(x/y);

console.log(x-y);

console.log(x%y);

console.log(x\*\*y);

console.log(x++);

console.log(--x);

...........comparision Operators..............

let x =0;

console.log(x<1);

console.log(x>=1);

console.log(x!=1);

.........Equality Operators.................

Strict Equality Operators (Type + value)

console.log(1===1);

console.log('1'===1);

Loss Equality Operators

.......... Ternary Operators..............

let points = 110;

let type = points > 100 ? 'Gold' : "Silver";

console .log(type);

..........Logical And (&&).............

Returns True if both Operands are TRUE

console.log(true && true); 3441035368

let highIncome = true;

let goodGrade = true;

let eligible = highIncome && goodGrade ;

console.log(eligible);

........ Logical OR || operator............

Returns the TRUE even one operands are True

let highIncome = true;

let goodGrade = true;

let eligible = highIncome || goodGrade ;

console.log(eligible);

let color = 'red';

let color1 = undefined;

let defaultValue = 'blue';

let givenColor = color || defaultValue;

let givenColor1 = color1 || defaultValue;

console.log(givenColor);

console.log(givenColor1);

..............Bit Wise Operators...................

1 = 00000001

2 = 00000010

R = 00000011

R = 00000000

Read, Write, Execute

00000100

00000110

00000111

console.log(1|2); // Bit Wise OR

console.log(1&2); // Bit Wise AND

............ Swapping Exercise..............

let a = 'red';

let b = 'blue';

let temp = b;

b = a;

a = temp;

console.log(a);

console.log(b);

............ "For in loop"................

const person ={

name: 'vishal',

age:23

}

for (let key in person)

console.log(key,person[key])

const employee = {

name:"vishal palla",

age:"25",

area:"Narsipatnam",

company:"cbre"

}

for(let key in employee){

console.log(key +" :"+ employee[key]);

}

......... "FIZZBUZZ".......................

Divisible by 3 => Fizz

Divisible by 5 => Buzz

Not divisible by both 3 & 5 => FizzBuzz

Not a number => "Not a number"

const input = fizzBuzz(3)

console.log(input);

function fizzBuzz(input){

const a = fizzBuzz(input)

console.log(a);

}

fizzBuzz(3);

fizzBuzz(5);

fizzBuzz(15);

fizzBuzz(11);

fizzBuzz("vishal");

.......... "ExerCise" ...............

function checkSpeed(speed){

if(speed<=74){

console.log("ok")

}

if(speed===180){

console.log("Licence is suspended")

}

if(speed!==180 ){

if(speed>75){

let i = speed-75;

let j = Math.ceil(i/5)

console.log(j)

}

}

}

checkSpeed(71)

........... " Exercise "...........

const movie = {

title : "Arjun Reddy",

duration : 180,

rating : 10,

director : "Sandeep Reddy Vanga"

}

showProperties(movie)

function showProperties(obj){

for (ele in obj){

if(typeof obj[ele]==="string"){

console.log(ele+":"+obj[ele])

}

}

}

............. "Array"................

const colors = ['red', 'blue', 'yellow']

const employees = [{name:"vishal", age:25}, {name:"vikas", age:26}]

for(index in colors){

console.log(index, colors[index])

}

for(let index in colors){

console.log(index, colors[index]);

}

for(index in employees){

console.log(employees[index]);

}

........."NEW ..." "FOR OF LOOP" mostly used for Arrays............................

const colors = ['red','blue','green','yellow'];

const employees = [{name:"vishal", age:25}, {name:"vikas", age:26}]

for (color of colors){

console.log(color)

}

for(employee of employees){

console.log(employee);

}

WE use "FOR IN" loop to iterate over the properties of an "object"............

WE use "FOR OF " loop to iterate over the elements or items in "ARRAY" .............

........"Object Oreinted Programming ".............

const cirle = {

radius :1,

location :{

x:1,

y:2

},

isVisible : true,

draw : function(){

console.log("draw");

}

};

cirle.draw(); // method

............"Factory Functions"................

Factory Function uses camel naming convintion....

function createCircle(radius){

return{

radius,

draw(){

console.log("draw")

}

};

}

const Circle1 = createCircle(1);

console.log(Circle1)

const Circle2 = createCircle(2);

console.log(Circle2)

........"Constructor Function"........

Constructor function uses pascal naming convintion

function Circle(radius){

this.radius = radius;

this.draw = function(){

console.log('draw')

}

}

const circle = new Circle(2);

console.log(circle)

................" Dynamic nature of Objects "................................

const circle = {

radius :1

};

circle.color = 'yellow';

circle.draw = function(){}

console.log(circle)

delete circle.color;

delete circle.draw;

console.log(circle)

const person = {

name:"vishal"

}

person.age = 24;

person.iq = function(){}

console.log(person);

.........."Functions are Actually Objects".......................

In java Script Functions are Objects......

function Circle(radius){

this.radius = radius;

this.draw = function(){

console.log('draw')

}

}

const Circle1 = new Function('radius',`this.radius = radius;

this.draw = function(){

console.log('draw')

}

`);

Circle.call({}, 1)

Circle.apply({}, [1,2,3])

const another = new Circle(1);

........"Two types are there in java Script...they are"..................

ValueType : ReferenceType:

Number Object

String Functions

Boolean Array

Symbol

undefined

null

primitive data types are independent......

Values are Primitives types are copied by values

Reference are objects which are copied by Reference..........

...... "Cloning a "Object" "...........

const circle = {

radius : 1,

draw() {

console.log('draw');

}

};

const another = Object.assign({}, circle);

const another = { ...circle};

const another = { ...circle,border:'black'};

console.log(another);

..........................Garbage Collection...........................

Memory allocation and deallocation happens automatically behind the scene and you have

no control over it..This job is done by Garbage Collector in javaScript , it takes takes

care about the variables which are not used so long and it will delete it automatically

by using complex algorithms written behind the scenes

.......................String.............................

String primitive

const message = "iam vishal";

String object

const another = new String("hi");

.......................Literals........................

Object {}

Boolean true, false

String '' , ""

Template ``

..............Template Literals in Java Script.........................

we use these teemplate literals to write a string as we like and to eliminate the

concatination and "/n" signs to get a new line

const name = "vishal";

const another =

`Hi ${name} ${2+3}

Thank you for joining my mailing list.

Regards,

Mosh`;

........................DATE OBJECT.................

const now = new Date();

const date1 = new Date(2018,4,11,9);

const date2 = new Date("May 11 2018 09:00");

date1.getFullYear;

.........Exercise...................

const address = {

street: "Peenarpalam",

city: "visakhapatnam",

ZipCode:531116

};

showAdress(address);

function showAdress(address){

for (key in address){

console.log(key,address[key])

}

}

........................Exercise................................

const address = FactorycreateAddress('a','b','c')

// Factory function

function FactorycreateAddress(street,city,zip){

return{

street,

city,

zip

};

}

// Constructor Function

const address1 = new ConstructorAddress('a','b','c')

function ConstructorAddress(street,city,zip){

this.street = street,

this.city = city,

this.zip = zip

}

............................."Arrays".................................................

add elements at "End"

const a =[3, 4, 5]

a.push(5,6)

// Beginning

a.unshift(1, 2)

// Middle

a.splice(2,0,"a","b")

console.log(a)

...................Iterating a ARRAY..............

const arr = [1,2,3,4,5]

for (value of arr){

console.log(value)

}

arr.forEach(function(number){

console.log(number)

})

arr.forEach(function(number,index){

console.log(index,number)

})

arr.forEach( number=> console.log(number))

arr.forEach( (number,index)=> console.log(index,number))

..............Joining Arrays.............................

the elements in the array can be joined using join method

const message ="This is vishal"

const part = message.split(" ")

console.log(part)

const joined=part.join("-")

console.log(joined)

...............Sorting the Arrays.................

sorting the arrays which contains values

const num =[2,3,1]

num.sort();

console.log(num)

num.reverse()

console.log(num)

sorting the arrays which contains objects

const courses = [

{ id:1, name :'Node.js'},

{ id:2, name :'JavaScript'},

];

courses.sort(function(a, b){

if(a.name<b.name) return -1;

if(a.name > b.name) return 1;

return 0;

});

console.log(courses)

sorting the arrays which contains objects irrespective of UpperCase and LowerCase

const courses = [

{ id:1, name :'Node.js'},

{ id:2, name :'javaScript'},

];

courses.sort(function(a, b){

const name1 = a.name.toUpperCase();

const name2 = b.name.toUpperCase();

if(name1<name2) return -1;

if(name1 > name2) return 1;

return 0;

});

console.log(courses)

..........Testing the Elements of an Array................

ANY NEGATIVE

const numbers =[1,-1,3,8];

const allPositive = numbers.every(function(value){

return value >=0;

});

console.log(allPositive)

ANY POSITIVE

const numbers =[1,-1,3,8];

const atleastOnePositive = numbers.some(function(value){

return value >=0;

});

console.log(atleastOnePositive)

..................Filtering an Array.............

It removes any negative values present in the array

const numbers = [1,-1,2,3];

const filtered = numbers.filter(function(value){

return value >=0;

});

console.log(filtered)

// Another way writing the same function

const filtered1 = numbers.filter(value=> value >=0

);

console.log(filtered1)

..................Mapping a Array...............

It means putting a array elements to another array

const num = [2, 3, 4, 5]

const num1 = num.map(function (value){

return "<li>" +value+"</li>"

})

console.log(num1)

const num2 = num.map(value=>"<li>" +value+"</li>")

const strings = num2.join('');

console.log(strings)

....................Redcung an Array.............................

sum of the values in a array

const arr= [2, 8, 7]

const sum = arr.reduce((accumilation,currentvalue) =>{

return accumilation + currentvalue;

},0)

console.log(sum)

.......................... Finding the Elements in a Array for primitives.......................

const arr = [1,8,9,1,6,5]

console.log(arr.indexOf(8))

console.log(arr.lastIndexOf(1))

console.log(arr.includes(6))

console.log(arr.indexOf(6,1))

.......................... Finding the Elements in a Array for references.......................

const courses = [

{id:1, name:"vishal"},

{id:2, name:"vikas"}

];

courses have two objects in it, in "courses.find() "" we pass argument(call back function) that is a function called function(course), here in function course is a parameter which holds the object in a array and we search course.name==""...if it present it will return a boolean true........

const a = courses.find(function(course){

return course.name==="vishal";

})

console.log(a)

.............. Arrow Function......................

when pass a function as argument or as a call back function with only single parameter we can use a arrow function..we remove the 'function' word and remove the braces for parameter and to separate the parameter from the body we use '=>' symbol and remove the curly braces and if we have only one return statement we can remove that one also.....

const arr = [

{id :1, name:'vishal'},

{id:2 , name:'vikas'}

]

const a = arr.find(course => course.name==="vikas"

)

console.log(a)

............Removing a element in a Array.............

const numbers = [1,9, 8,7]

..... To remove a element at the 'end' we use pop() method..

const last = numbers.pop();

console.log(numbers)

....To remove a element at the "beginning" we use shift....

const first = numbers.shift();

console.log(numbers)

....To remove a element at the "middle" we use shift....

numbers.splice(2,2);

console.log(numbers)

................Emptying an Array..................

let numbers = [1,2,3,4]

solution 1

numbers =[]

solution 2

numbers.length = 0;

solution 3

numbers.splice(0, numbers.length)

solution 4

while (numbers.length > 0)

numbers.pop();

.............Combining & Slicing Arrays.................

const first =[1,2,3]

const last =[4,5,6]

const final = first.concat(last)

console.log(final)

const obj =[

{id:1 , name:"vishal"},

{id:2 , name:"vikas"}

]

const a ={id:3, name:"ajith"}

const obj1 = obj.concat(a)

console.log(obj1)

.................The Spread Operator...............

It is used to combine the elements in a arrays and also to copy

const first =[1, 2, 3]

const second =[4, 5, 6]

const combined = [...first,...second]

console.log(combined)

const copy = [...combined]

console.log(copy)

const b = prompt("what is your name")

console.log(b)

.........................Finding maximum......................

let max =0;

const first =[1, 2, 3, 4 ,5]

for (value of first){

if (value>max){

max = value;

}

}

console.log(max)

Function Declaration vs Function Expression

Function Declaration

function walk(){

console.log("walk")

}

// Anonymous Function Expression

let run = function(){

console.log("run")

};

run()

........................Hoisting.................

"Hoisting" is the process of moving function declaration top of the file by the JavaScript Engine.

walk()

function walk(){

console.log("walk")

}

Function Expression

In the anonomus function expression we cannot call the function before or top of the function..

const run = function(){

console.log("run")

};

...............Arguments & NaN..................

function sum(a,b){

return a+b // b is undefined

// 1 + undefined = "NaN"

}

console.log(sum(1))

function sum(){

let total =0;

let a =0;

for (a of arguments){

total+=a;

}

return total

}

console.log(sum(1,2,3,4,5))

...........Rest Operator......................

Rest operator looks like spread operator but its not....rest operations actually used with parameter in functions to get all the arguments assigned to the function and store as a array

function sum(...args){

//...args here it does put all arguments in array

console.log(args)

let total =0;

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

total+=args[i]

}

return total

}

console.log(sum(1,10,3,4,5));

let vs var

var is function scoped and let, const is block scoped

when we uses "var" it will create a global variable and attaches the global variable to window object.............