**JAVASCRIPT**

1. **What is JavaScript?**

JavaScript is a high-level, interpreted programming language primarily used for adding interactivity and functionality to web pages. It can be executed in a web browser, making it a crucial part of web development.

1. **How to include JavaScript in HTML?**

<!DOCTYPE html>

<html>

<head>

<title>My JavaScript Page</title>

</head>

<body>

<h1>Hello, JavaScript!</h1>

<script>

// JavaScript code goes here

alert('Hello, world!');

</script>

</body>

</html>

**3. Variables:**

Variables are used to store data in JavaScript. You can declare variables using var, let, or const.

var name = "John"; // Declaring a variable and assigning a string value

let age = 30; // Declaring a variable with let

const pi = 3.14; // Declaring a constant variable

1. **Data Types:**

JavaScript has various data types, including strings, numbers, booleans, objects, arrays, and more.

var name = "John"; // String

var age = 30; // Number

var isStudent = true; // Boolean

var person = { // Object

firstName: "John",

lastName: "Doe"

};

var fruits = ["apple", "banana", "cherry"]; // Array

1. **Functions:**

Functions in JavaScript allow you to group code into reusable blocks. Here's how you can define and call a function:

function greet(name) {

console.log("Hello, " + name + "!");

}

greet("John"); // Calling the function

1. **Conditional Statements:**

You can use if, else if, and else statements for conditional logic in JavaScript:

var age = 18;

if (age < 18) {

console.log("You are a minor.");

} else if (age >= 18 && age < 65) {

console.log("You are an adult.");

} else {

console.log("You are a senior citizen.");

}

1. **Loops:**

JavaScript offers for, while, and do-while loops for repetitive tasks:

for (var i = 0; i < 5; i++) {

console.log("Iteration " + i);

}

var count = 0;

while (count < 5) {

console.log("Count is " + count);

count++;

}

1. **Events:**

JavaScript is often used to handle events in web pages, such as button clicks or form submissions. You can attach event listeners to HTML elements:

var button = document.getElementById("myButton");

button.addEventListener("click", function() {

alert("Button clicked!");

});

These are the foundational concepts of JavaScript. As you progress, you can explore more advanced topics like DOM manipulation, AJAX, and frameworks like React or Angular for building interactive web applications.

What is JavaScript? What can you do with it?

Where does JavaScript code run? And what is the difference between JavaScript and ECMA Script? So let's start with the first question. What is JavaScript?

JavaScript is one of the most popular and widely used programming languages in the world right now. It's growing faster than any other programming languages, and big companies like Netflix Walmart, and PayPal, build entire applications around JavaScript. And here's the average salary of a JavaScript developer, in the united states. That is $72,000 a year. So it's a great opportunity to get a great job learning JavaScript.

What can you do with JavaScript?

For a long time, JavaScript was only used in browsers to build interactive web pages. Some developers referred to JavaScript as a toy language. But those days are gone because of huge community support and investments like large companies like Facebook and Google. These days you can build full blown web or mobile apps as well as real-time networking apps like chats and video streaming services, command-line tools, or even games.

Where does JavaScript code run? JavaScript was originally designed to run only in browsers. So every browser has what we call a JavaScript Engine that can execute JavaScript code. For example, the JavaScript engines in Firefox and Chrome are SpiderMonkey and V8. In 2009, a very clever engineer called Brian Doll took the open source JavaScript engine in Chrome, and embedded it inside a C++ program. He called that program Node. So Node is a C++ program, that includes Google's V8 JavaScript engine. Now with this we can

run JavaScript code outside of a browser, so we can pass our JavaScript code to Node for execution. And this means the with JavaScript we can build the back-end for our web and mobile applications. So, in a nut shell, JavaScript code can be run inside of a browser, or in Node. Browsers and Node provide a runtime environment for our JavaScript code.

what is the difference between JavaScript and ECMA Script?

Well, ECMA Script is just a specification, JavaScript is a programming language that confirms to this specification. So, we have this organization called ECMA, which is responsible for defining standards, they take care of this ECMA Script specification. The first version of ECMA Script was released in 1997. Then, starting in 2015, ECMA has been working on annual releases of a new specification. So in 2015. ECMA Script 2015 which is also called ECMA Script version 6, or ES6 for short. This specification defined many new features for JavaScript. Alright, enough theory, let's see JavaScript in action. So every browser has a JavaScript engine, and we can easily write JavaScript code here without any additional tools. Of course, this is not how we build real world applications, but this is just for a quick demo. So open up Chrome, right click on an empty area and go to inspect. Now this opens up Chrome developer tools. Here, select the console tab, this is our JavaScript console, and we can write any valid JavaScript code here. So, type this, console.log put a single code here, and then Hello World Another single code to terminate, close the parenthesis, and add a semi colon at the end.

// Command line

<Script > ===this tag maybe placed in two places mostly at the end

<script>

console.log('kokila');

</script>////**Structure to print the tag line**

**CODE FOR SOLUTION VARIABLES AND CONSTANTS**

///This is my first javascript code

// let for changable code and const for default code

let person ={

name:'kokila',

age:90

}

//Dot notation

person.age=30;

//Bracket notation

let selection='name';

person[selection]='mary';

console.log(person.name);

**ARRAYS**

We use arrays to store the list of objects

// arrays

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

console.log(selectedColors[0]);

//print all the objects in the selected orders

// arrays

let selectedColors =['red', 'purple','white'];

console.log(selectedColors);

///Showing up an extra color which is not mentioned inside the array list

// arrays

let selectedColors =['red', 'purple','white'];

selectedColors[3]= 'green';

console.log(selectedColors);

//To find the length of the array

// arrays

let selectedColors =['red', 'purple','white'];

selectedColors[3]= 'green';

console.log(selectedColors.length);

**Functions**

**///funcion() is a method to call and the parsing parameter is name**

function greet(name){

console.log("hello"+ name );

}

greet('john');

greet('marry');

//if the function is parsed morethan one parameter it is necessary to call them with the method , if we didn’t do this it will return the undefined valued for the next paramter

Input :

function greet(name,lastName){

console.log("hello"+ name + '' + lastName);

}

greet('john');

greet('marry');

Output:

hellojohnundefined

hellomarryundefined

///if the parameter if defined and its value is mentioned

Input:

function greet(name,lastName){

console.log("hello"+ name + '' + lastName);

}

greet('john','python');

greet('marry','java');

Output:

hellojohnpython

hellomarryjava

**Types Of Functions**

//Performing a Task

function greet(name,lastName){

console.log("hello"+ name + '' + lastName);

}

// greet('john','python');

// instead of calling the greet function call square function

//Function calaculates a value

function square(number){

return number\*number;

}

let number =square(4);

console.log(number);

**Without assigning variables and direct printing**

//Performing a Task

function greet(name,lastName){

console.log("hello"+ name + '' + lastName);

}

// greet('john','python');

// instead of calling the greet function call square function

//Function calaculates a value

function square(number){

return number\*number;

}

console.log(square(2));

**Operators in JavaScript**

**Variables Expressions Algorithms (Logic’s)**

1. Arithmetic Operators

let x=4;

let y=2;

console.log(x+y);//6

console.log(x-y);//2

console.log(x\*y);//8

console.log(x%y);//0

console.log(x/y);//2

console.log(x\*\*y);//16 power of

//Increment operator

console.log(x++);//4

console.log(x);//5

console.log(++x);//6

//Decrement Operator

console.log(y--);//2

console.log(y);//1

console.log(--y);//0

1. Assignment Operators

**Input:**

let x=11;

x+=5;

console.log(x);

let y=10;

y-=1;

console.log(y);

let z=9;

z\*=7;

console.log(z);

let m=4;

m%=9;

console.log(m);

let c=33;

c/=20;

console.log(c);

// x++; Both are Equal

// x= x+1;

// x= x+5;

// if we need to add more than 1 value means we need this assignment operator

**Output:**

assignmentOperators.js:4 16

assignmentOperators.js:7 9

assignmentOperators.js:10 63

assignmentOperators.js:13 4

assignmentOperators.js:16 1.65

1. Comparison Operators

//Relational Operators

let x=1;

console.log(x > 0);

console.log(x>=1);

let y=9;

console.log(y<2);

console.log(y<=10);

//Equality Operators

let z=8;

console.log(z===1);

console.log(z!==1);

**Output:**

true

true

false

true

false

True

**Equality Operators**

//Strict Equality

console.log(1===1);//accepts it as true

console.log('1'===1);//not accept it as true because its checks value the power of === sign

console.log(true===1);//not accepted false

//Lose Equality

console.log(1==1);//true

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

console.log(true==1);//true

// in lose equality it accepts everything as true so js always goes with Strict Equality

**Ternary Operators**

//if a customer has more than 100points they are gold customer otherwise they are a silver customer

let points =100;

let type= points>90 ? 'gold':'silver';// if the statement is true its rerturns value gold or else its return false

console.log(type);//prints gold

let type2= points<90 ? 'gold':'silver';

console.log(type2);// prints silver

1. **Logical Operators (With Booleans)**

There are 3 operators

**AND**

//Used this operators to make decisions on multiple condition

//We have three type of operators logical and, logical or & logical not

//logical AND(&&) returns true , if both the operands are true

console.log(true && true);//prints true

console.log(false && false);//prints false

console.log(true && false);//prints false

console.log(false && true);//prints false

let highIncome= true;

let goodCreditScore = true;

let eligibleForLoan = highIncome && goodCreditScore;

console.log(eligibleForLoan);//its shows true

let highIncome1= true;

let goodCreditScore1 = false;

let eligibleForLoan1 = highIncome1 && goodCreditScore1;

console.log(eligibleForLoan1);//its shows false  
**OR**

//Used this operators to make decisions on multiple condition

//We have three type of operators logical and, logical or & logical not

//logical AND(&&) returns true , if both the operands are true

console.log(true && true);//prints true

console.log(false && false);//prints false

console.log(true && false);//prints false

console.log(false && true);//prints false

let highIncome= true;

let goodCreditScore = true;

let eligibleForLoan = highIncome && goodCreditScore;

console.log(eligibleForLoan);//its shows true

let highIncome1= true;

let goodCreditScore1 = false;

let eligibleForLoan1 = highIncome1 && goodCreditScore1;

console.log(eligibleForLoan1);//its shows false

//OR OPERATOR

//Logical OR(||)

// it returns true if one of the operand returns true

let highIncome2= true;

let goodCreditScore2 = false;

let eligibleForLoan2 = highIncome1 || goodCreditScore1;

console.log(eligibleForLoan2);//its shows true

**NOT**

//NOT Operator (!)

//its always shows if its true shows false and if the condition is false its shows true

let highIncome3 = false;

let goodCreditScore3= false;

let eligibleForLoan3= highIncome3|| goodCreditScore3;

console.log("Eligible",eligibleForLoan3); //it shows Eligible false

let applicationRefused =!eligibleForLoan3;

console.log(" Application Refused", applicationRefused); //it prints Application Refused true

**Logical Operators (Without Boolean's)**

False|| True

True

False|| ‘Mosh’

“Mosh”

False|| 1

1

//Falsy is not a boolean false

// what are all the falsy values: Undefined , null, 0, false ,'', NaN(Not a Number)

// Anything that is not falsy ---> Truthy

//Short -Circulating false||1||2 output is :1

let userColor ='red';

let defaultColor ="blue";

let currentColor= userColor|| defaultColor;

console.log(currentColor);

// /the output is red

let userColor1 =undefined;

let defaultColor1 ="blue";

let currentColor1= userColor1|| defaultColor1;

console.log(currentColor1);

// /the output is blue it gets the default value

// With this techniques we can provide the value for default colors

1. Bitwise Operators

///1=00000001

// 2=00000010

// R=00000000(AND )

// R=00000011

console.log(1 | 2);// output is 3 Bitwise OR

console.log(1 & 2);//output is 0 in Bitwise AND

//Read , write, execute

//if the user has read permission the 1st 5 digits are common

// read=00000100

// read,write=00000010

// read,write, execute =00000001

const readPermission = 4;

const writePermission = 2;

const executePermission = 1;

let myPermission = 0;

myPermission =myPermission |readPermission |writePermission ;

console.log(myPermission);//prints 6

let message = (myPermission & readPermission) ? 'yes' : 'no';

console.log(message); //prints yes

const readPermission1 = 4;

const writePermission1 = 2;

const executePermission1 = 1;

let myPermission1 = 0;

myPermission1 =myPermission1 |writePermission1 ;

let message1 = (myPermission1 & readPermission1) ? 'yes' : 'no';

console.log(message1); //prints no

**Operators Precedence**

let x=2+3\*5;

console.log(x);//17 is output

let y = (2+3)\*5;

console.log(y);//25 is output

//Always use the paranthesis for operators for exact performance

**QUIZ**

7- What is the value of y?

let x = 10;

let y = (x > 5) && (x < 15);

a- 10

b- 5

c- 15

**d- true**

8- What is the value of x?

let x = 5;

x += 3;

a- 3

**b- 8**

c- 15

d- 5

9- What is the value of y?

let x = 10;

let y = x++;

**a- 10**

b- 11

c- 12

d- 13

10- What is the value of y?

let x = 1;

let y = x !== 2;

a- 1

b- 2

c- false

**d- true**

**Swapping of variables**

let a='blue';

let b='red';

let c=a;

a=b;

b=c;

console.log("the color is " + a);//the color is red

console.log("the color is " + b);//the color is blue

**IF-ELSE STATEMENTS**

// Hour

//if hour between 6am and 12pm , its shows Good Morning

//if it is between 12pm and 6pm, : Good Afternoon

//otherwise : Good Evening

let hour =1;

//if 1 prints Good Evening

// if 11 prints Good Morning

// if 14 prints Good Afternoon

if(hour >=6 && hour < 12)

console.log("Good Morning");

else if(hour>=12 && hour < 18)

console.log("Good Afternoon");

else

console.log("Good Evening");

**Switch Case Statement**

let role ='mediator';

switch(role){

case 'guest':

console.log('Guest User');

break;

case 'moderator':

console.log('Moderator User');

break;

case 'mediator':

console.log("Mediate User");

break;

default:

console.log("Unknown User");

}

// if role is null its prints unknown user

// if role is defined as guest it will print guest user

// if role is defined as mediator it will print as mediator user

// Same Code in IF ELSE CONDITION

if(role=='guest') console.log("Guest");

else if (role =='moderator') console.log("moderator");

else if(role == 'mediator') console.log ("mediator");

else console.log("unknown user");

Output:mediator

**LOOPS**

**FOR LOOP (initial expression; condition; increment)**

for(i=0;i<5;i++){

console.log("Hello World" , i);

}

// it prints 5 Hello World

// if i is mentioned it will print helloworld with numbers 0 1 2 3 4 like that

for(x=1;x<=5;x++){

console.log("Hello World x is" , x);

}

// it will print 5 hello world starts from 1 to 5

for(let z=1;z<=5;z++){

if(z%2!== 0)

console.log("The odd sequence are" +" " +z);

}

// it will print only odd number 1 ,3 and 5

for(let g = 5 ;g>=1;g--){

if(g%2!== 0)

console.log("The odd sequence are" +" " +g);

}

// it will print odd numbers in the revise order 5 3 1

**WHILE LOOP**

for (let i =0;i<=5;i++){ //this i loop is different from the other i beacuse it is accessible only in the for loop

if( i % 2!==0)

console.log(i);

}

let i = 0;

while(i <=5 ){ //while(Condition)

if( i % 2!==0)

console.log(i);

i++;

}

**DO- WHILE LOOP**

// Do While Loop

let i=9 ;

do{

if(i%2!== 0)

console.log(i);

i++;

}while(i <= 5); // 9 will be the output

// the condition will be executed 1st and after that statement is checked whether its working or not

**INFINITE -LOOP**

let i =0;

while(i < 5){

console.log(i);

// i++;

}

// it prints the endless output

// Another method to use infinte loop

// while(true){

// }

// let x= 0;

// do{

// //x++;

// }

// while(x<5){

// }

// for(let i = 0; i<10 ){

// }

**FOR IN LOOPS (improper format)**

// For in loops

const person = {

name: 'Mosh',

age:30

For this code we get output with undefined

name undefined

age undefined So we can move to the bracket notation

};

for (let key in person)

console.log(key, person.key);

// Dot Notation

person.name;

// Bracket Notation

person['name']

**FOR IN LOOPS (Proper format)**

// For in loops

const person = {

For this code we get output

name mosh

age 30

name: 'Mosh',

age:30

};

for (let key in person)

console.log(key, person[key]);

Another Example with index

// For in loops

const person = {

name: 'Mosh',

age:30

};

for (let key in person)

console.log(key, person[key]);

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

for(let index in colors)

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

OUTPUT:

name Mosh

age 30

0 red

1 green

2 blue

We use the for in loop to iterate over the properties of an object and the for of loop to iterate over the elements or items

**FOR OF LOOPS**

// For in loops

const person = {

name: 'Mosh',

age:30

};

for (let key in person)

console.log(key, person[key]);

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

for(let color of colors)

console.log(color);

OUTPUT:

name Mosh

age 30

red

green

blue

**BREAK AND CONTINUE**

Normal loop

let i =0;

while(i<=10){

console.log(i);

i++; //it gives output from 0 to 10

}

In this normal loop to get out from the loo at certain condition we can use break

**BREAK**

let i =0;

while(i<=10){

if(i===5)

break;

console.log(i);

i++; //it gives output from 0 to 4

}

**CONTINUE**

Using Continue Keyword

let i =0;

while(i<=10){

// if(i===5)

// break;

if(i % 2 === 0){

i++;

continue;

}

console.log(i);

i++; //it gives output from 1,3,5,7, 9

}

With Break keyword, you jump out of a loop, and with the continue keyword, we jump to the next iteration.

**EXERCISE**

Write a function that takes two numbers and returns the maximum of two

let number =max(1,3);

console.log(number);

function max(a,b){

if(a>b) return a;

else return b;

}

// it’s a normal method to print the maximum value

let number =max(1,3);

console.log(number);

function max(a,b){

return (a > b ) ? a:b //Condition to checks whether the given parameters is bigger or not , find the maximum value in JavaScript .

}

Function to check whether the given size is portrait or not ?(landscape)

console.log(isLandscape(200,89));

function isLandscape(width,height){

return (width>height)? true:false;

}//it prints true

**FUNCTIONS**

1. Function Declaration
2. Anonymous Function Declaration

// function Declaration

function walk(){

console.log('walk');

}

// Anonymous Function Declaration

let run = function(){

console.log('run');

}

// Named Function Declaration

let ran = function(){

console.log('run');

}

let move = run;

run();

move();

//Output is run

// Output is run

Function Declaration = We can able to run the method before initialization

Function Expression = We can able to run the method after the declaration and initialization .

**HOISTING**

Hoisting is the process of moving function declarations to the top of the file. And this is done automatically by the JavaScript engine that is executing this code. So that's the reason we can call functions that are defined using the function declaration syntax before their definition.

**Callee**:which returns the currently executed function, so you see we have a reference to our sum function.