BASIC JAVASCRIPT

1. What is JavaScript ?

* JavaScript was initially created to “make web pages alive”.
* JavaScript enables developers to add functionality to web pages by manipulating the content and behavior of elements on the page.
* Java Script is a object based scripting language which is designed to enhance the functionality of web page./make it to be dynamic

1. What is program ?

* A program is a set of instructions that perform a specific task but when a program start execution it is said to be process .

1. **What is programming language and what is scripting language**

* In programming language is a bulding blocks that put together and create a program and have compiler and interpreter so it convert source code machine code before it can run .
* It is used to build standalone application (like web,native application , no need of internet it runs by os ).
* Scripting languages are not compiled type it is interpreted , that means code is read and executed by line by line so it will easy to perform any task and take less efficient
* Ex : python ,js ruby – Scripting language

1. **What is Object based and object oritend language ?**

* Object oritented language are supports all features of opps like c++ and js , but object based languages support oops but there are lack of some features like run time polymorphism and inheritance , encapsulation .ex : js.
* Object based programming language has in bult object like window object but object oriented has not built in .
* Pytthon is a purely objecte oriented bcz it has not primitive type .

1. Why js is called Object Based language ?

// console.log("Hello World !");

// x=20;

// console.log(x);

// // alert("OK")

// alert('Hello');

// alert('World');

// // alert(3 + 1+ 2); // 6 will alert

// //[1,2].forEach(alert)  // 1 then 2 alert

// // ["Soumya ","Ranjan","Nayak"].forEach(alert)

// x=30;

// alert(x)+alert(2) // alert(30) then alert(2)

                                    //  Variable Concept :

// variable is a container which stores

// x=20

//  t=typeof (x)

// console.log(t); // number

//  let $=3000 -- allow

//  let @=3 // not allow

// 23abc=23 // not allow

//abc12=20// allow

//  var \_123=23 // allow

// var x\_\_=20 // alow

 //                           Types of variable

//  Note : Variable are very Case sensitive

// var apple ;

// var APPLE ;

// let x;

// const  pi=3.121

// console.log(pi);

//  There are two scope in variable i.e local scope and global scope

// x=20

// if variable present out side block , loop function then it is global and it can be acess any where .

// local varaible prsent inside the function,loop or any block .

// function fun1() {

//         const a = 1000;

//         if (a > 0) {

//             const b = 10;

//             console.log("Inside the if block :", b);

//         }

//         // console.log("Outside the if block :", b); // error referemce of b is not defined .

//         // console.log(a);

//     }

//     fun1();

    // console.log("Outside the function :", b); //error alsoe

    // console.log(a);

    var num =20;

function myfun(){

    if(num>20){

        var contain="A string ";

        //let x="i am let "

        console.log("inside block"+num);// print no doubut

    }

     console.log(conatin) //  if the vqariable is var and const type error

    // console.log(x); // undefined

    // var x=2 // if write after

}

myfun();

// ----------------------

// console.log(x); // Output: undefined

// var x = 5;

// console.log(y); // Error: ReferenceError: y is not defined

// let y = 10;

// so  Variables declared with "var"  have function scope or global scope,

// whereas variables declared with let have block scope.

//------------------------------

function example() {

    console.log(a); // Output: undefined

    var a = 10;

    console.log(b); // Error: ReferenceError: b is not defined

    let b = 20;

  }

  example();   // output is undefined  then show error

Variable hoisting is a behavior in JavaScript where variable declarations are moved to the top of their scope during the compilation phase, regardless of where the actual declaration appears in the code. This means that variables can be accessed and used before they are declared

console.log(x); // Output: undefined

var x = 5;

Unit-02 ( DATA TYPE )

  // DATA TYPE in JS

 /\*

    there are  2 types of data in js

    1. premitive

              - number,string,null,undefined,Bigint,Boolean,Symbol

    2. Non-Premitive

               - Array, Object

 \*/

let A=1/0;

console.log(typeof(A) ,A) //  number  infinity ♾️

let power = 2^10-1;

console.log(typeof(power),power); // number 8

const bigNum = 123n;

const regularNum = 456;

console.log( typeof bigNum); // bigint  , that means it is a big number 1000000000e83838383 like

// console.log(bigNum + regularNum); // Throws TypeError

let s="String"

console.log( typeof s); // string

console.log(1>2 ) //boolean type return true or false

console.log( typeof 1); // print number

let no1=100,no2=100;

console.log(no1==no2); // true

let n=1

var n1=1

console.log(n===n1); // true

(10>20)?(x=200):(x=100)

console.log( typeof x, x); // number print if then 100 lo

console.log(typeof true); // boolean

// let age = 19;

// if(age < 18){

//     console.log('unable to vote');

// }

// else if(age == 18){

//     console.log('go get your voter id')

// }

// else{

//     console.log('go out and vote')

// }

let variable;

console.log(variable); // Output: undefined

function doSomething() {

  // No return statement

}

console.log(doSomething()); // Output: undefined

let user = {

    name: "John",

    age: 25

  };

  user = null;

  console.log(user); // Output: null , that means user has no value it null

  // Symbol type is used to create a unique and immutable value

// that can be used as property keys for objects.

let id1=Symbol("id")

let id2=Symbol("id");

console.log(id1===id2); // false so symbol is unique

unit :03 (Operator)

let a = 10;

let b = 5;

// Arithmetic operators

console.log(a + b); // Output: 15

console.log(a - b); // Output: 5

console.log(a \* b); // Output: 50

console.log(a / b); // Output: 2

console.log(a % b); // Output: 0

console.log(a \*\* b); // Output: 100000

// Assignment operators

a += b;

console.log(a); // Output: 15

// Comparison operators

console.log(a > b); // Output: true

console.log(a < b); // Output: false

console.log(a >= b); // Output: true

console.log(a <= b); // Output: false

console.log(a === b); // Output: false

console.log(a !== b); // Output: true

// Logical operators

const c = true;

const d = false;

console.log(c && d); // Output: false

console.log(c || d); // Output: true

console.log(!c); // Output: false

// Bitwise operators

const e = 5; // Binary: 0101

const f = 3; // Binary: 0011

console.log(e & f); // Output: 1 (Binary: 0001)

console.log(e | f); // Output: 7 (Binary: 0111)

console.log(e ^ f); // Output: 6 (Binary: 0110)

console.log(~e); // Output: -6 (Binary: 1010)

console.log(e << 1); // Output: 10 (Binary: 1010)

console.log(e >> 1); // Output: 2 (Binary: 0010)

// Other operators

const arr = [1, 2, 3];

console.log(arr instanceof Array); // Output: true

console.log(typeof a); // Output: number

console.log(typeof 'Hello'); // Output: string

Reserve word in Js

await break case catch

class const continue debugger

default delete do else

enum export extends false

finally for function If

import in instanceof new

null return super switch

this throw true try

typeof var void while

with yield

Type Conversion

// let a=10;

// let b="a"

// console.log(a-b);// NAN

// console.log(0 / 0); // Output: NaN

// console.log(Infinity / Infinity); // Output: NaN

// console.log(NaN + 5); // Output: NaN

// console.log(NaN / 10); // Output: NaN

// console.log(isNaN("abc")); // Output: true

// console.log(Number.isNaN("abc")); // Output: true

// console.log(isNaN(123)); // Output: false

// console.log(Number.isNaN(123)); // Output: false

// console.log(isNaN(NaN)); // Output: true

// console.log(Number.isNaN(NaN)); // Output: true

//  type casting

let number=10;

let y=number.toString()

let z=String(number)

console.log(typeof y, y);

console.log(typeof z, z);

const str = "123";

const num1 = parseInt(str); // Convert to integer

const num2 = parseFloat(str); // Convert to floating-point number

console.log(typeof num1, num1); // Output: number 123

console.log(typeof num2, num2); // Output: number 123

const value = "Hello";

const bool = Boolean(value);

console.log(typeof bool, bool); // Output: boolean true

// Implicit Type Conversion: JavaScript also performs automatic or

// implicit type conversions in certain situations,

// such as when using operators between different types:

const num = 42;

const st = "The answer is " + num;

console.log(typeof st, st); // Output: string "The answer is 42"

// explicityly type conversion

// const bool = true;

// const num = Number(bool); // Convert boolean to number

// const str = String(bool); // Convert boolean to string

// console.log(typeof num, num); // Output: number 1

// console.log(typeof str, str); // Output: string "true"

 let x=10;

 let lett="a"+x

 console.log( typeof lett , lett); // string , a10 ; it im

 let x = 1 + "1";

console.log(x); // Output: "11"

let y = 5 - "2";

console.log(y); // Output: 3

let z = 2 \* "3";

console.log(z); // Output: 6

let a = 10 / "5";

console.log(a); // Output: 2

let b = +"42";

console.log(b); // Output: 42

Conditional