Table of Contents

[JAVASCRIPT 2](#_Toc127820298)

[VAR , LET , CONST 2](#_Toc127820299)

[Operators 4](#_Toc127820300)

[increment operator 4](#_Toc127820301)

[decrement operator 5](#_Toc127820302)

[Assignment operator 5](#_Toc127820303)

[pre/post increment / decrement 7](#_Toc127820304)

[Comparision operator 8](#_Toc127820305)

[ternary operator 8](#_Toc127820306)

[Logical Operator 9](#_Toc127820307)

[String 11](#_Toc127820308)

[indexing 11](#_Toc127820309)

[substring 12](#_Toc127820310)

[substr 13](#_Toc127820311)

[Replace :- 13](#_Toc127820312)

[replaceAll 13](#_Toc127820313)

[Array 13](#_Toc127820314)

[Push method- 13](#_Toc127820315)

[twodimensional array 14](#_Toc127820316)

[Methods on Array----> 15](#_Toc127820317)

[Loops – 16](#_Toc127820318)

[For loop 16](#_Toc127820319)

[While loop 17](#_Toc127820320)

[Do loop 19](#_Toc127820321)

[for-in loop 19](#_Toc127820322)

[for-of loop 20](#_Toc127820323)

[for-each loop 20](#_Toc127820324)

[Objects 21](#_Toc127820325)

[Example 1 21](#_Toc127820326)

[example 2 21](#_Toc127820327)

[Function 21](#_Toc127820328)

[Function keyword 22](#_Toc127820329)

[Arrow Funtion 22](#_Toc127820330)

[Anonymous keyword 23](#_Toc127820331)

[problems. 24](#_Toc127820332)

[break 25](#_Toc127820333)

[continue 25](#_Toc127820334)

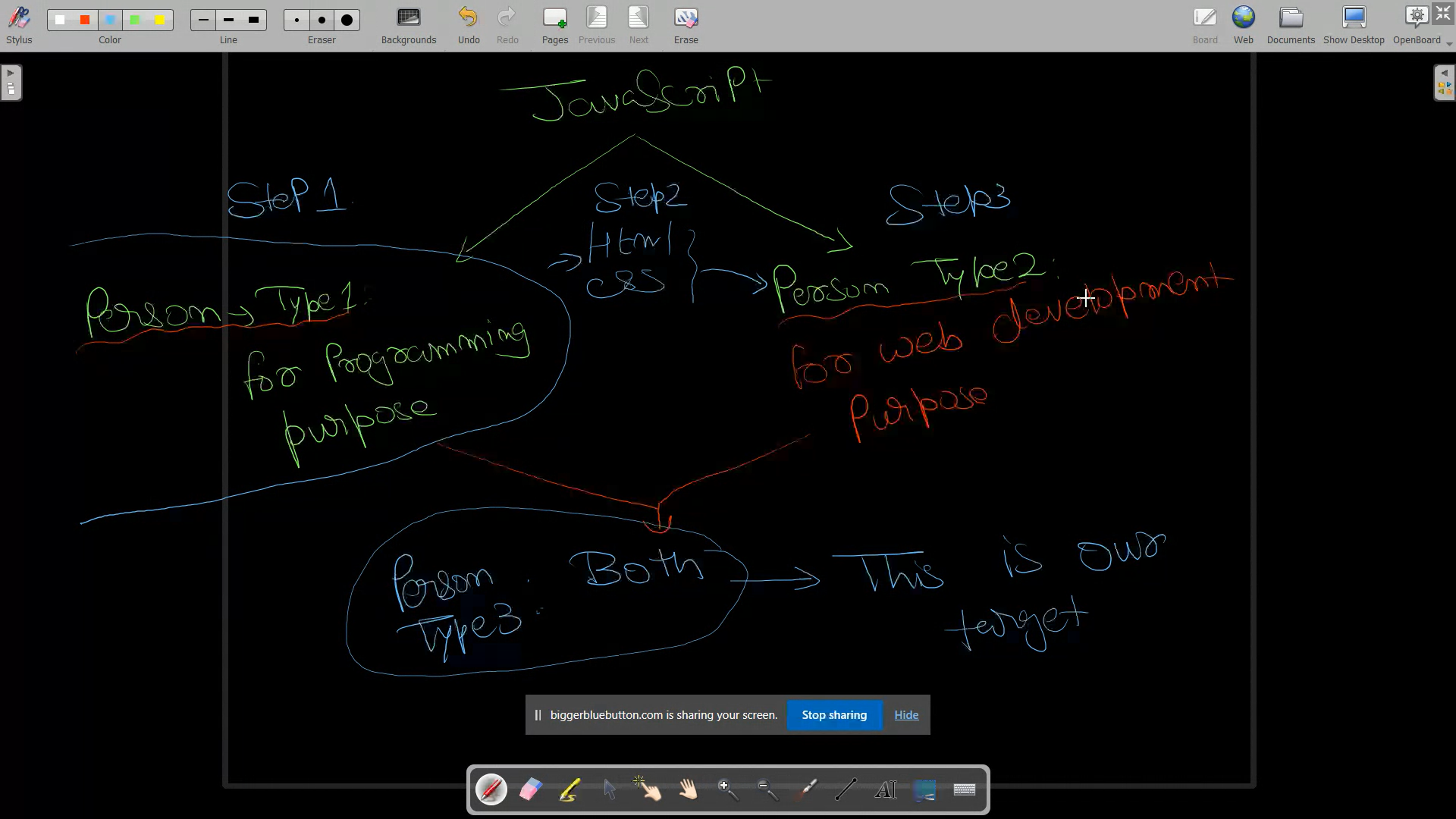
[Floor,ceil 26](#_Toc127820335)

[math.random() 26](#_Toc127820336)

# JAVASCRIPT

console.log(" Hello World")

{ console used for printing the output }



Containers to hold data can be of 3 types in javascript

# VAR , LET , CONST

VAR ---

Syntax - --- Var Container\_name

Var a;

Var firstname;

Var first\_name

Var first name spaces in variable name are not allowed

LET ---

Syntax --- Let Container\_name

Let a;

Let firstname;

Let first\_name

Let first name spaces in variable name are not allowed

CONST ---

Syntax --- const Container\_name

const a;

const firstname;

const first\_name

const first name spaces in variable name are not allowed

DIFFERENCE b/w VAR , LET, CONST

* Var, let, const are the containers for storing data
* You can also call it a variable

Re-declare

Re-Assign

Containers of type Var are re declared as well as re assign

Var x; declaring variable

Var x; re declaring variable

X=20; assigning value

X=21; re assigning value

Containers of type Let can be re assign but not re declare

let x; declaring variable

let x; produce error

X=20; assigning value

X=21; re assigning value

Containers of type const can not be re assign as well as re declare

let x; declaring variable

let x; produce error

X=20; assigning value

X=21; produce error

# Operators

operator in javascript

let x = 30 '=' is assignment operator

let y = 30 + 40 '+' addition operator

let z = 60 - 40 '-' subtraction operator

let a = 60 \* 40 '\*' multiplication operator

let b = 60 / 40 '/' division operator

let c = 60 % 40 '%' Modulus operator give remainder ex 20 will be output

let d = 60\*\*2 '\*\*' power/exponential operator ex- 60\*60

## increment operator

a=10

a++; i.e. a=a+1;

## decrement operator

m=20;

m--; i.e. m=m-1;

increment and decrement operator only used with 'var' and 'let' but not with const keyword.

ex-

var a=10 ; a++;

let b=50 ; b--;

const c=50 ; c--; it will produce error

increment and decrement operator not work with const keyword or constant values .it will produce error.

const c=20;

c++;

c--; const c is constant so value of c is also constant.so it will produce error;

var x=20++;

let y=21++;

both will produce error this cant be writen like this as 20 is constant in this

## Assignment operator

"=";

var a;

a=20; in this 20 is assigned to a;

'+=' a+=20 a=a+20

'-=' a-=20 a=a-20

'\*=' a\*=20 a=a\*20

'/=' a/=20 a=a/20

'\*\*=' a\*\*=20 a=a\*\*20

'%=' a%=20 a=a%20

x=3 y=5

1. x+=y

x=x+y

x=3+5

x=8

2. x\*\*=y

x=x\*\*y

x=3\*\*5

x=243

3. x-=y+20

x=x-(y+20)

x=-22

4. x=y++ in post increment firstly value assignd to x then it will increment

x=5

5. x=++y in pre increment first value will be increment then it will assigned to the x

x=6

6. x%=(y+3)

x=x%8

x=3%8

x=3

## pre/post increment / decrement

Pre increment= a=++b;

in pre increment first value will be increment then it will assigned to the a

Post increment= a=b++

in post increment firstly value assignd to a then it will increment

pre Decrement= x=--y

in pre decrement first value will be decrement then it will assigned to the x

Post decrement= x=y—

in post decrement firstly value assignd to x then it will decrement

var a=30,b=3;

c=a++ + b++;

console.log(c);

console.log(a);

console.log(b);

var a=3;let b=4;

const c= --a + a++ + b--  -  --b;

console.log(a);

console.log(b);

console.log(c);

## Comparision operator

a==b equality check

it will produce true if value of a is equal to value of b otherwise it will produce false

a===b strict equlity check

it will prodiuce true if value of a and b as well as type of a and b both are same then it is strict equality check

< less than

> greater than

<= less than equal to

>= grerater than equal to

!= not equal to

!== not equal to value as well as type

2=="2"

true

2==="2"

false

## ternary operator

condition?exp1:exp2

if the condition is true then exp1 is the output else condition is false then output will exp2

var a=true?5:3

output 5

var a=30<=30?5:6

output 5

## Logical Operator

**&& (Logical and)**

a b output

true true true

true false false

false true false

false false false

in logical and (&&) if any condition is false then output produced by this result in false

var a=3,b=6;

d=(a++>=3 && b-- <=6)

true true

a=4

b=5

d= true

**|| (logical or)**

a b output

true true true

true false true

false true true

false false false

in logical or (||) if any condition is true then output produced by this result in true

**! (logical not)**

!true = false

!false = true

var a= true

var b=!a

console.log(b); false

in logical and operator (&&)

var a= 3,b=6

var d= a++ >= b-- && b++ <= a--

3 >= 6 && 5<=4

false && (here first condition is false so the second condition will not evaluated because in and operator if one conditionis false the output is false)

so value are false

a=4

b=5 because nextcondition will not run

In logical or (||)

var a= 6,b=3

var d= a++ >= b-- || b++ >= a--

6 >= 3 || 2>=7

true || (here first condition is true so the second condition will not evaluated because in or operator if one conditionis true the output is true)

so value are true

a=7

b=2 because next condition will not run

# String

## indexing

var a = "ANkit"

a[0]=A

a[1]=N

a[2]=k

a[3]=i

a[4]=t

i. length()

syntax- var.length;

example:- s.length

4

"name".length

4

ii. slice

return the substring and ending point was not included.

syntax:- var.slice(start,end)

ex-: s.slice(1,3)

'am'

"name".slice(1,3)

'am'

"animesh".slice(2)

'imesh'

negative slice

negative slice also allowed like s.slice(1,-1) result in 'am'

example

s.slice(-3)= 'ame'

"animesh.slice(-6)='nimesh'

### substring

var s="sudhanshu";

console.log(s.substring(2,5))=dha

console.log(s.substring(2))= dhanshu

negative substring not allowed ever

### substr

s.substr(start,length) here start is the index of variable and end refer to lenth of string

var s="sudhanshu"

console.log(s.substr(2,6));= dhansh

### Replace :-

it will replace only first occurance

s.replace('u','a');

### replaceAll

it will replace all the occurance

s.replaceAll('u','a')

# Array

Array - Linear data structure used to hold multiple value of any type.

var arr[]=

arr=[1,2,3,4,5,6,7,8]

value inserting by indexing

arr[8]=15;

arr=[1,2,3,4,5,6,7,8,15]

arr[2]=16 { here the val

ue is override on the index no 2 }

arr=[1,2,16,4,5,6,7,8,15]

## Push method-

push method insert new values next to the last indexi of Array

ex- arr=[1,2,3,4]

arr.Push[15];

arr=[1,2,3,4,15]

arr.Push[16,17,18]

arr=[1,2,3,4,15,16,17,18]

push method also used to store multiple values at a time.

Linear data structure- container to hold item

in array eveery cell present at unique memory location

1024 1028 1032 1036 1040 - this is mwmory location consist of 4 bytes

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

## twodimensional array

arr=[[2,3,4],[7,8,9]]

arr[0]=[2,3,4]

arr[1]=[7,8,9]

arr= 0 1 2

arr[0][2]=4 0 2 3 4

1 7 8 9

sort= it sort the array in ascending order but first it convert all the element in string.

it sort only the first digit not the whole number like 2,15,32,12= 12,15,2,32

it just compare the first character of the number

length= it shows the length of array

arr= ['a','c','f','b']

arr[1]='g'== ['a','g','f','b']

arr.sort();== ['a','b','f','g']

arr[2]='k'; ['a','b','k','g']

arr= ['a','b','k','g']

arr=[2,8,6,9,11,16,32]

arr.sort()= [11,16,2,32,6,8,9]

arr[1]=7 11,7,2,32,6,8,9

arr.sort= 11,2,32,6,7,8,9

arr[3]=5 11,2,32,5,7,8,9

## Methods on Array---->

(refer to folder 09-02-2023)

arr=[1,8,9,5]

arr.length()----> return the number of elements in Array

arr.push(18)---> used to insert a new element in end of the array

arr=[1,8,9,5,18]

arr.pop()---->usedto remove the element from the last in array

arr.shift()---> remove an element from the beginning af an array

arr.unshift()----> add an element in beginning of an array

delete.arr[1]--->it will replace index1 vlue with empty or undefined val

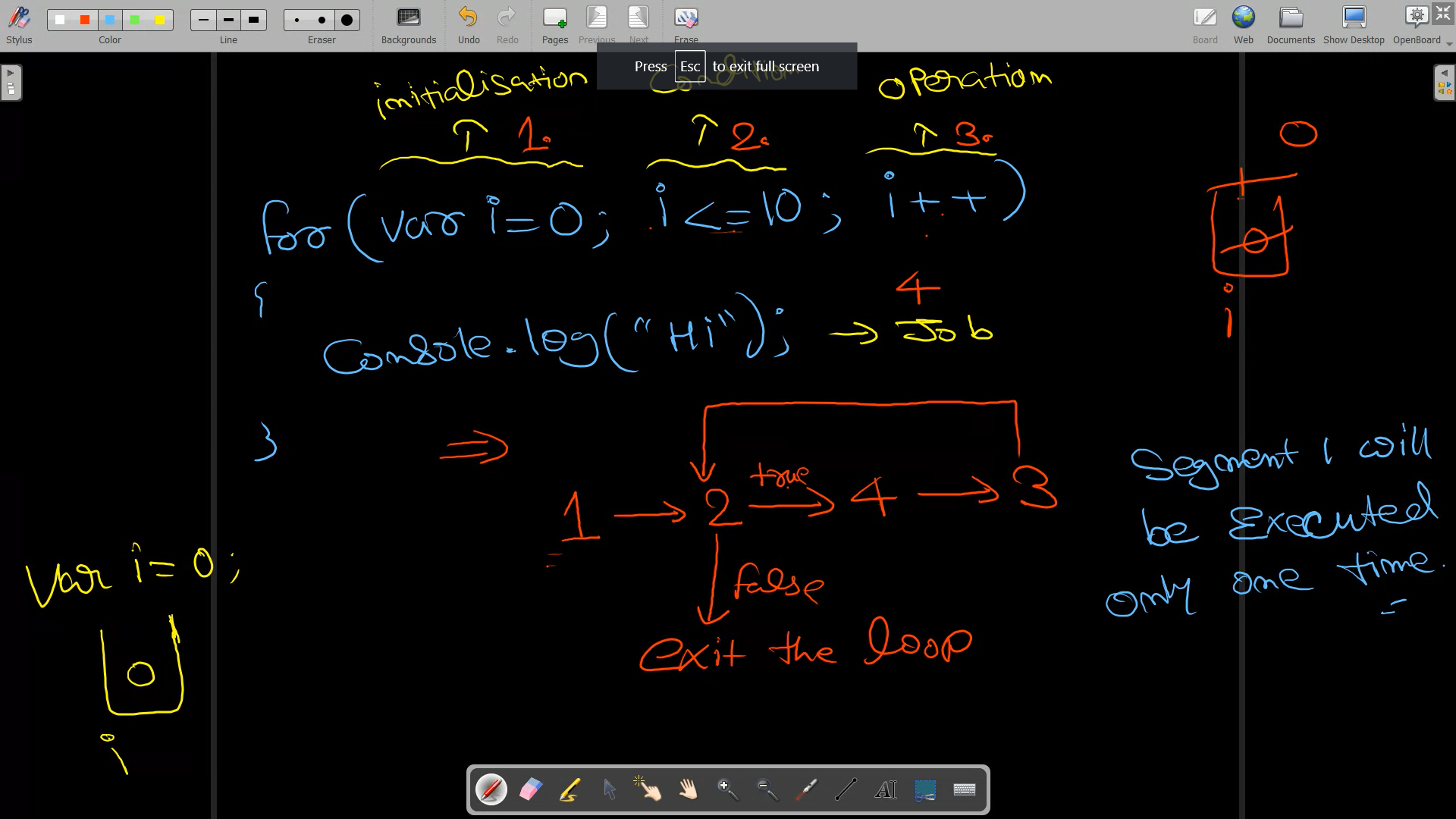
arr.splice(start,count)---> from index start remove count elemenmt

arr.splice(0,2);---->fom index 0 number of 2 element will removed

arr.splice(0,2,"string"/number)----> from index o number of two elements will removed and string or number will insert

# Loops –

## For loop



var j=0

for( ;j<=10;)

{

console.log("hi");

j++

}in this initialisation and operation can be done outside but not the condition of the for loop

Write afor loop to print no1 to 10

for(var num=1;num<=10;num++)

{

console.log(num);

}

Printing the array

arr=[2,3,8,6,9,10]

for(var ar=0;ar<arr.length;ar++)

{

console.log(arr[ar]);

}

sum of array

arr=[3,8,8,9,6,5]

var sum=0;

for(var i=0;i<arr.length;i++)

{

sum=sum+arr[i];

}

console.log(sum);

reversing the array

for(var i=arr.length-1;i>=0;i--)

{

console.log(arr[i]);

}

## While loop

While(condition){

job...

....

...

}

Job will keep on executing until the condition is true

var i=0

while(i<=10)

{

console.log("hi)

} this result in infinite loop

var i=0

while(i<=10){

console.log(i)

i=i+2

} this run upto 10

print odd nos.

/\*var i=1;

while(i<=100){

console.log(i);

i+=2;

}\*/

//print multiplication of 5 upto 100

var a=5,b=0;

var val=a\*b;

while(val<=100)

{

console.log(val)

b+=1;

val=a\*b;

}

for(var i=0;5\*i<=100;i++)

{

console.log(5\*i);

}

## Do loop

do {

job

}

while(condition)

job will be performed frst as long as condition is true.job will keep executuing

var i=10

do{

console.log("hi");

i-=1;

}while(i>=5)

do-while loop is guaranteed to be executed at least one time.

## for-in loop

const person={

name:"sudhanshu",

age:24,

height: 5ft 8 inch

}

for (var key in person){

console.log(key); {for in loop will return the key of object}

console.log(person[key]);

}

var arr=[2,3,4,5,6,4,5,9,5,6,];

for(var key in arr){

console.log(key);

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

}

## for-of loop

var arr=[2,3,2,5,4,6,6]

for(var v of arr){

console.log(v);

}

in for-of loop it gives the value of the index of arrow.

whereas in for-in loop it give the index

## for-each loop

var arr= [5,6,5,8,6,4,,4]

arr.forEach(myFunction); //myFunction will be 7 times, since it has 7 elements

function run for each element

function myFunction(key,value,index){

console.log(key,vlue,index);

}

function myFunction(value,index,arr){

every time will receive 3 arguments

1st--> value of the index

2nd--> index of the value in array

3rd--> array itself

}

# Objects

### Example 1

var person={

    name:"Sudhanshu",

    age:24,

    height:"5ft 8 inch"

}

console.log(person);

console.log(typeof person);

console.log(person.name);

console.log(person.age);

var x="x";

person[x]=90;

console.log(person);

person["g"]=15;

console.log(person);

### example 2

const person={

    person1:{

        name:"Ankit",

        age:24

    }

}

console.log(typeof person);

console.log(typeof person.person1);

console.log(person);

console.log("name:"+person.person1.name);

console.log(person.person1.age);

person["x"]=1;

console.log(person)

# Function

refer to folder 09-02-2023

Functions----->

Function keyword

Arrow

Anonymous

IIFE

immediately

invoked

Function

expression

## Function keyword

function dojob(){

console.log("alpha vernma)

}

dojob(); // calling function

dojobis a function that will print alpha verma on console.

waf to print all nos from 0 to 100

function odd()

{

var num;

for(var num=0;num%2==1;num++){

console.log("odd nos are:"+num)

}

}

odd();

### Arrow Funtion

( )=> {

}

Const dojob=func\_name()=>{

Declaring the function

}

Ex=

Var s=add(a,b)=>

{

Console.log(a+b)

}

Arrow function is that we can say that function without the function keyword.

### Anonymous keyword

When we don’t give any name to arrow function it is said to be anonymous function. In this we don’t gove any function name

Ex

()=>{

Console.log(a+b

}

function add(a,b) argument

{

console.log(a+b)

}

add(100,8); here it is the perimeter

function dojobs(){

return("alpjpa")

}

var name\_= dojobs() /// calling function by variable

console.log(name\_)

function addnos(...args){ // here 3 dots represent array.

console.log(args)

}

addnos()

addnos(2)

addnos(2,3)

addnos(3,1,5,1)

function dojob(a,b,...args) array always allowed in last

{

dojob(...args,a,b) this both line cannot be allowed in js

dojob(a,args...,b)

}

var arr=[3,8,9,8,9]

var arr2=[...arr] copy arr in arr2

## problems.

function dojob(){

    console.log("alpha vernma")

}

dojob(); // calling function

function oddno(){

    var i;

    console.log("odd nos are")

    for(i=0;i<100;i++){

     if(i%2==1){

       console.log(i);

    }

}

}

oddno();

function fact(a){

    var f=1;

    var a;

    for(var i=1;i<=a;i++){

        f=f\*i;

    }

    console.log(f);

}

fact(3)

refer to folder 13-02-2023

# break

for(var i=0;i<=100;i++){

if(i==51){

break // break statement used to break the loop

}

else{

process.stdout.write(i+"")

}

}

}

continue - it is used to skip an iteration

for(var i=0;i<=100;i++){

if(i==51)break;

if(i==21)continue;

console.log(i);

}

By using OR operator we can skip 2 iteration

# Floor,ceil

Math.floor

integer value just lesser than this number

ex- Math.floor(5.6)=5

Math.ceil

integer value just greater than number

ex- Math.floor(3.6)=4

math.random()-- it will generate a random value in b/w maximum and minimum value

math.floor(math.random()\*(max-min)+1)

while(true){

   var i=Math.floor(Math.random()\*(20-10)+10)

   if(i==15)break;

   if(i==12)continue;

    console.log(i);

    }

    while(true){

        var i=Math.floor(Math.random()\*(100-1)+1)

        if(i%7==0)break;

         console.log(i);

         }