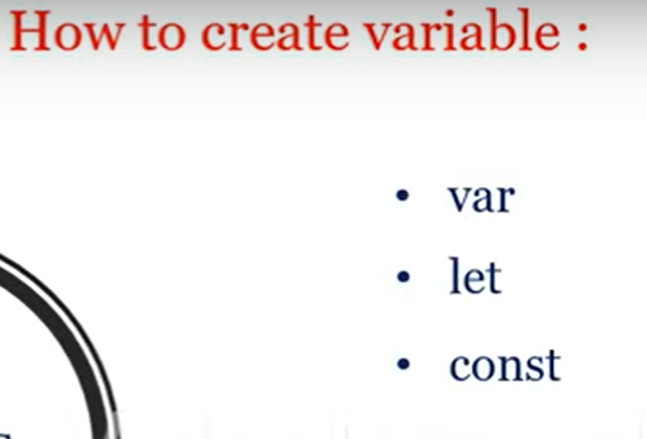


VARIABLES:->

g

->from 2014 before only we have one keyword is used declare variable

->after 2014 onwards there are taken the “let” and “const”

->when we are initialize with var keyword we can access before also

Like example

Console.log(sop);

Var sop=’hi’;

->in that var having the HOISTING process

->but let and const are reverse compare to var,it is not having the HOISTING PROCESS

->so we are getting the ERROR

->example

Console.log(sop); //error

let op=’hi’;

->let has block level scope

Means we are declaring variable inside the blocks so that is working in the block only

->{

Let x=19;

Console.log(x);//19

}

Console.log(x);//we are getting error

->var let

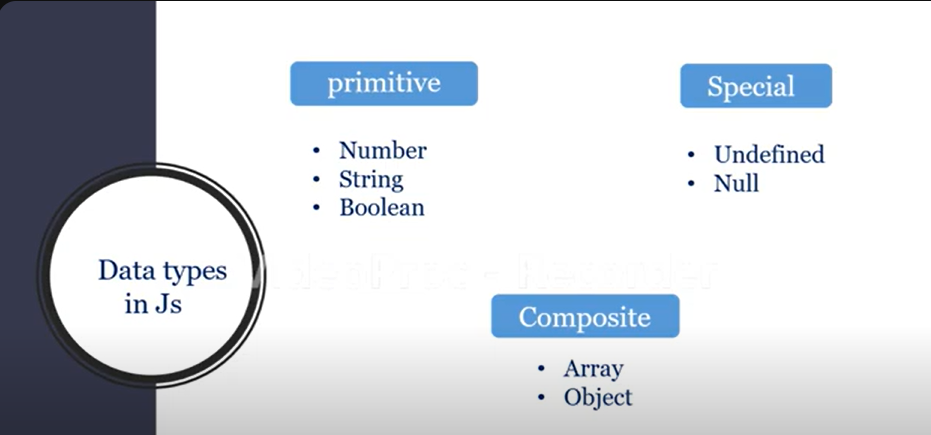
Var h=11; let j=10;

Console.log(h);//12 console.log(j);//10

Var h=12; j=11;

Console.log(h);//12 console.log(j);//11

DATA TYPES



Javascript is dynamically typed language it is automatically takes data

Var x=110;

Console.log(typeof(x));//number

Var h=’sai’;

Console.log(typeof(h));//string

Var t=true;

Console.log(typeof(t));//Boolean

Var g;

Console.log(typeof(g));//Undefined

Var d=null;

Console.log(typeof(d);//null

Var dd=[‘sai’,’kiran’,’tanniru’];

Console.log(typeof(dd));//array

Var obj={‘name’:’sai’,’age’:22,’sex’:’male’};

Console.log(typeof(obj));//object

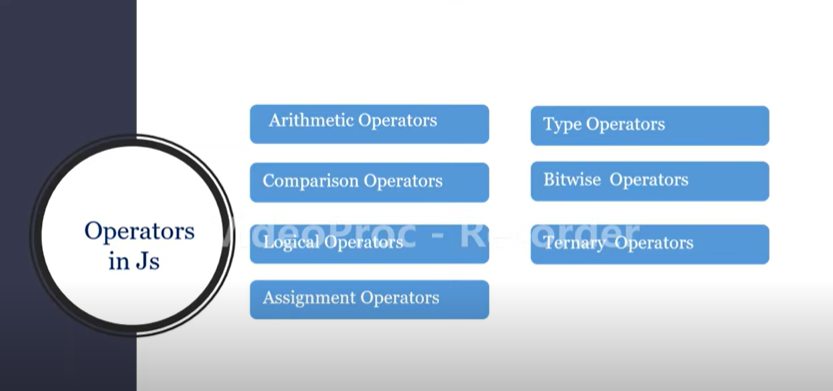
//\*\*\*\*\*\*WHEN WE ARE DEALING WITH ONLY ‘’.JS’’ FILES SO WE WANT TO RUN CODE IN THE “VS CODE” SO

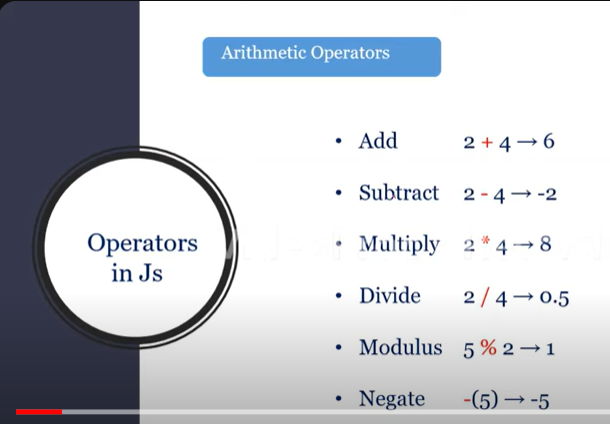
-> GO INTO TERMINAL

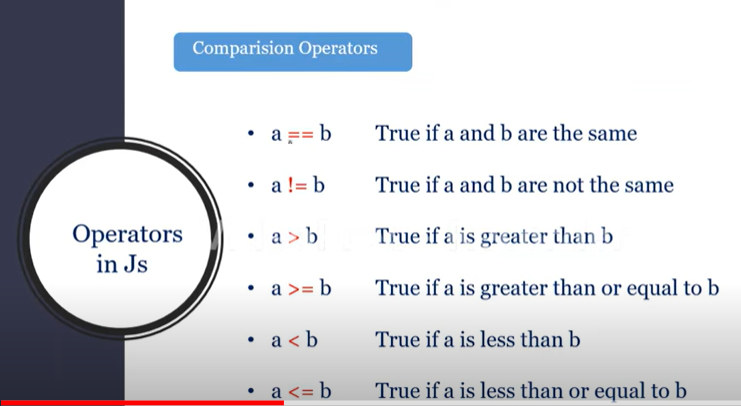
->SELECT CMD

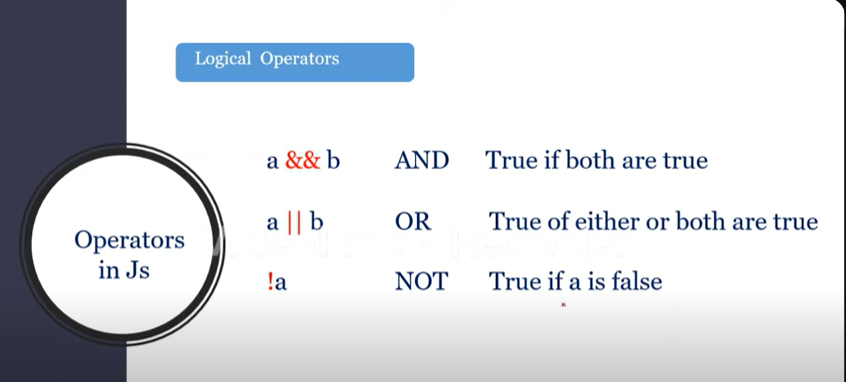
->node .js//

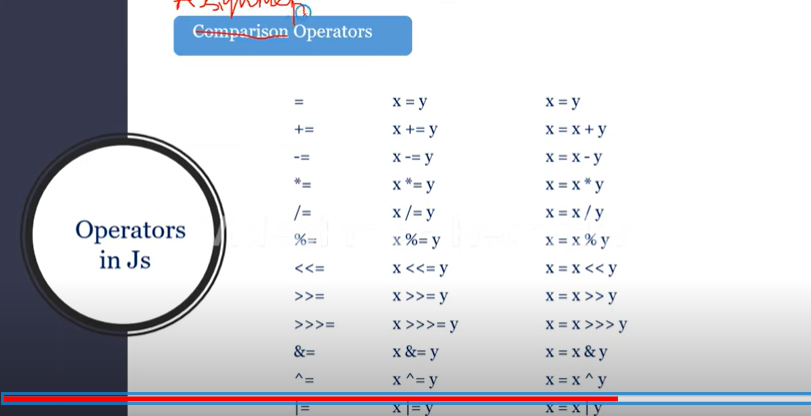
OPERATORS:->

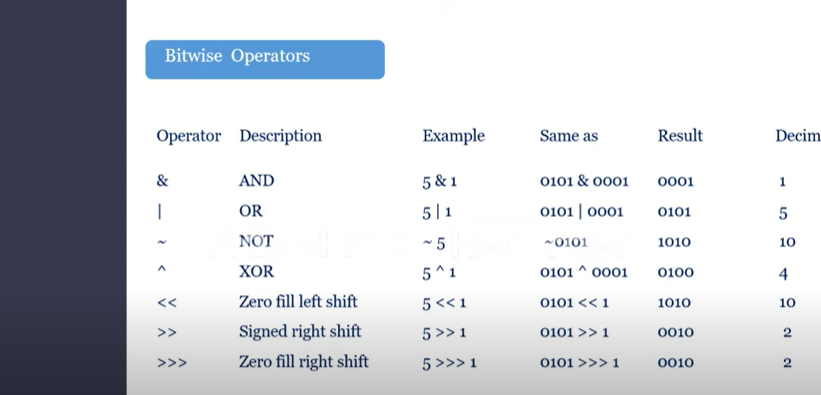












TYPE CONVERSION;->

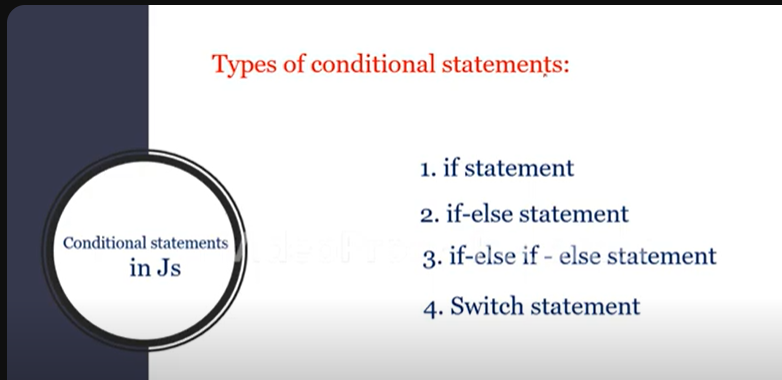
CHANGING ON DATA TYPE TO ANOTHER DATA TYPE

THEY ARE TWO TYPES

1 IMPLICIT TYPE CASTING

2.EXPLICIT TYPE CASTING

CONDITIONAL STATEMENTS:->



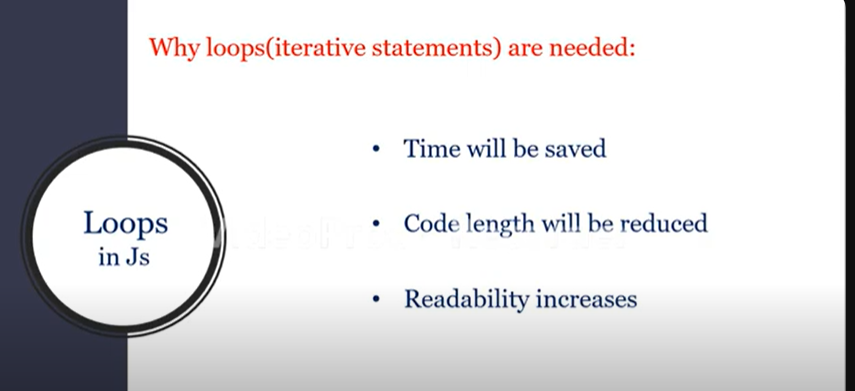
ALERT;->it is showing some message like alert

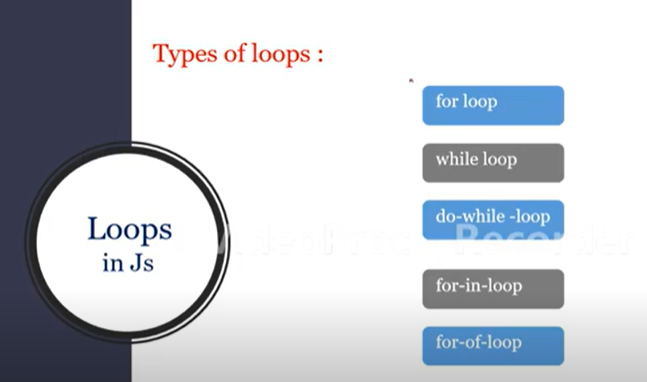
On browser

PROMPT:->it is taking the input from user

On browser

LOOPS:->





TEMPLATE LITERAL:->

===========

BACK TICK(`)

Var x=10;

Console.log(`your age is ${x}’);//your age is 10

FOR IN:->

++++++++++++

EX=

Var t=[‘sai’,’kiran’,tanniru’,’jon’,’snow’,’aegon’]

For( i in t){

Console.log(i+”<br>”);}//0 1 2 3 4 5

* For in is mainly concentrated on indexes

FOR OF:->

++++++++++

IT IS MAINLY CONCENTRATED ON VALUES

EX

Var t=[‘sai’,’kiran’,tanniru’,’jon’,’snow’,’aegon’]

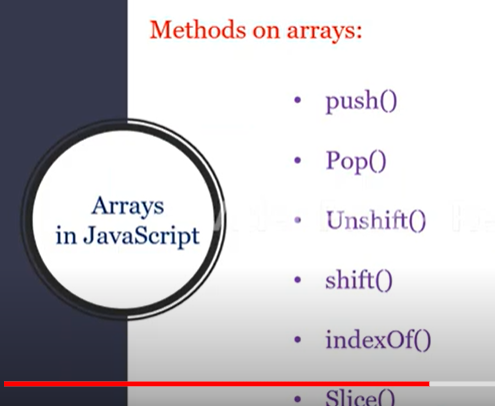
For(ele of t){

Console.log(t);//sai kiran Tanniru jon snow aegon

ARRAYS:->

Var s=[1,2,3,4];

Var s =new array(1,2,3,5);



Push:adding values last

Pop:deleting last value

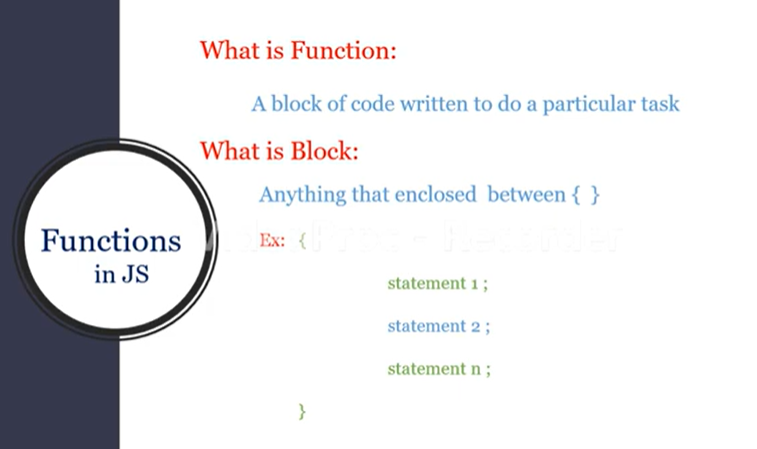
Unshift:adding value first

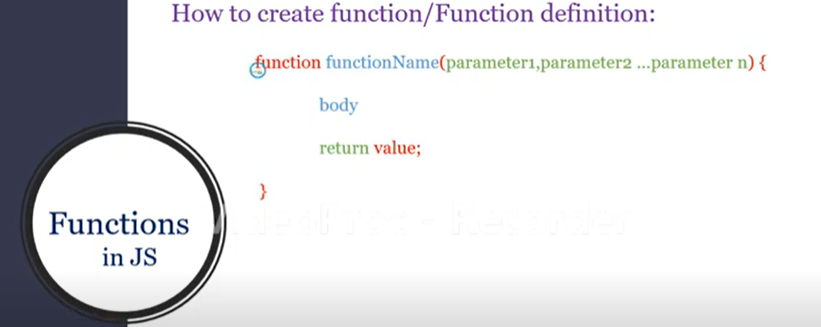
Shift:deleting first value

IndexOf :to know the index

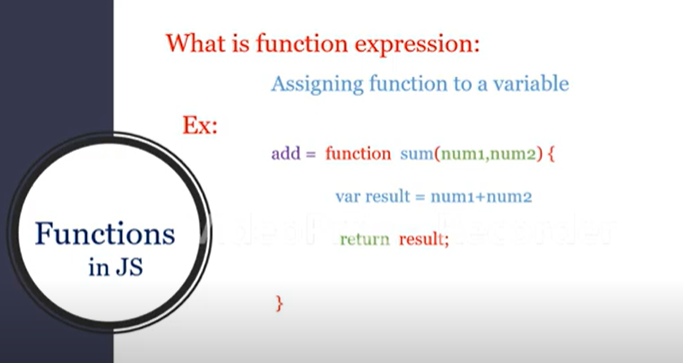
Slice:it is printing until you want ,it is like cuts

FUNCTIONS:->





FUNCTION EXPRESSION



var add=function sum(num1,num2){

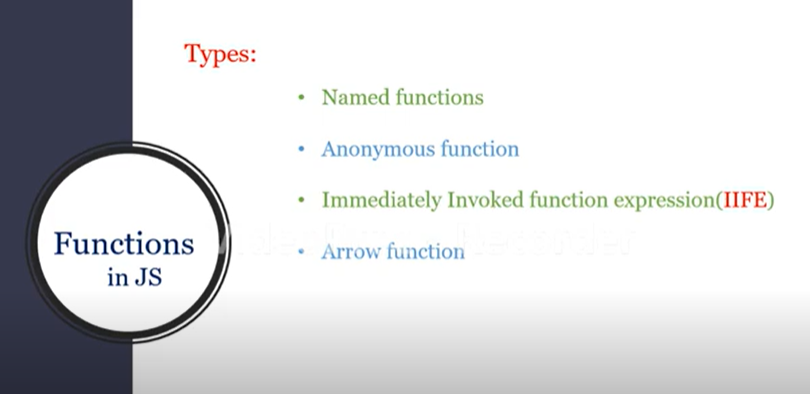
    var b=num1+num2;

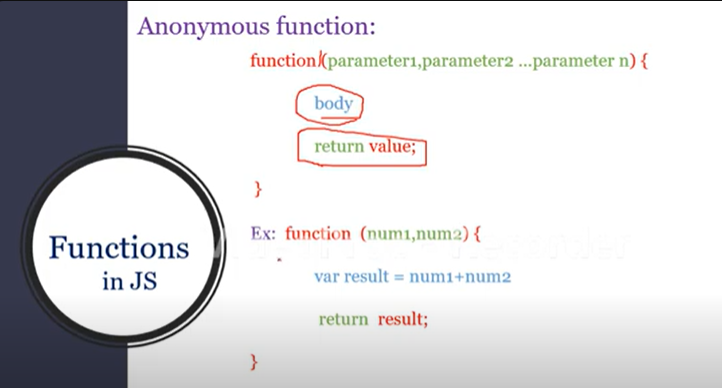
 return b;

}

 var c=add(10,20)

 console.log(c)

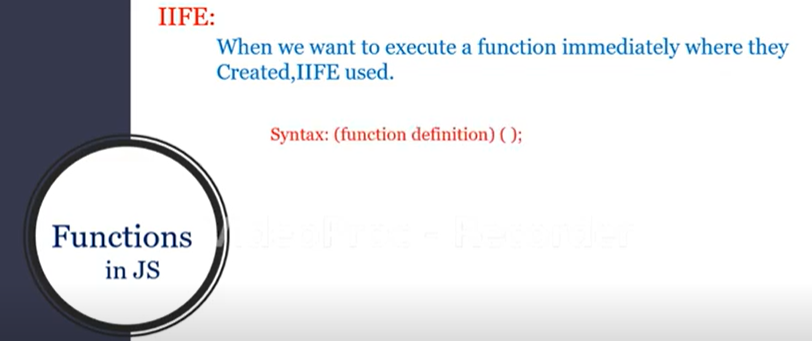




\*\*\*WE CANNOT WRITE THE ANONYMOUS FUNCTION DIRECTLY

\*\*\*WE HAVE TO ASSIGN DIRECTLY TO A VARIABLE

IMEDIATELY INVOKED FUNCTION EXPRESSION:->



(

    function product(n,m){

        var pro=n\*m;

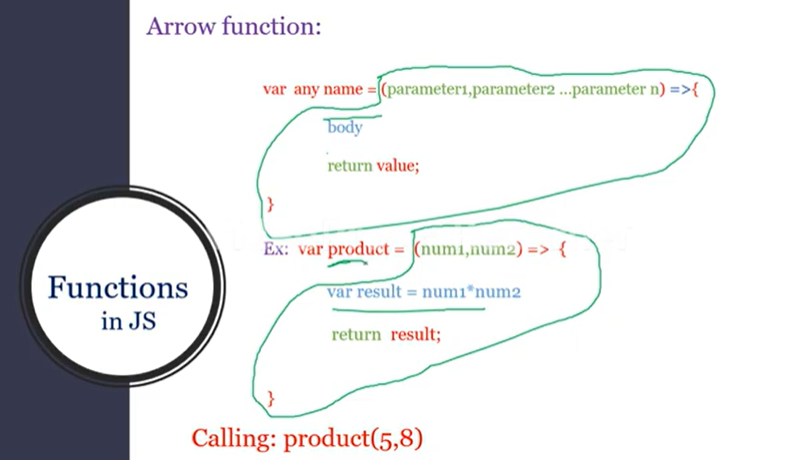
        console.log(pro);

    }

)

(10,20);

ARROW FUNCTION:->



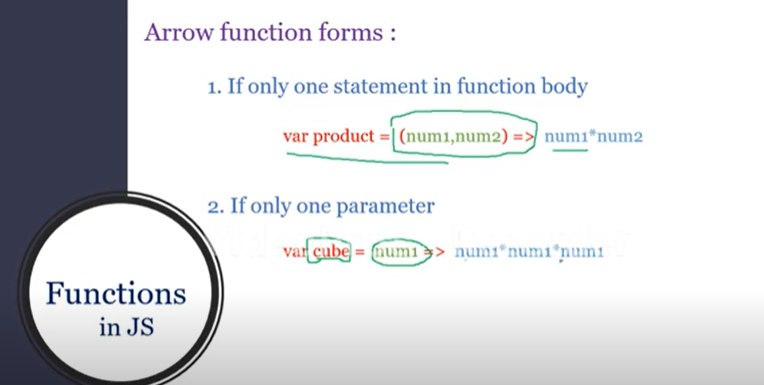
var ss=(R,S)=>{

    var mod=R%S;

    console.log(mod);

}

ss(20,7)



var ss=r=>r\*r\*r

console.log(ss(7))

var ss=(R,S)=>R%S

console.log(ss(20,7))

if no argument in arrow function

ex:->

greet=()=>console.log(“hi how are you”)

greet();//hi how are you

ex:->

greet=\_=>console.log(“hi how are you”)

greet();//hi how are you

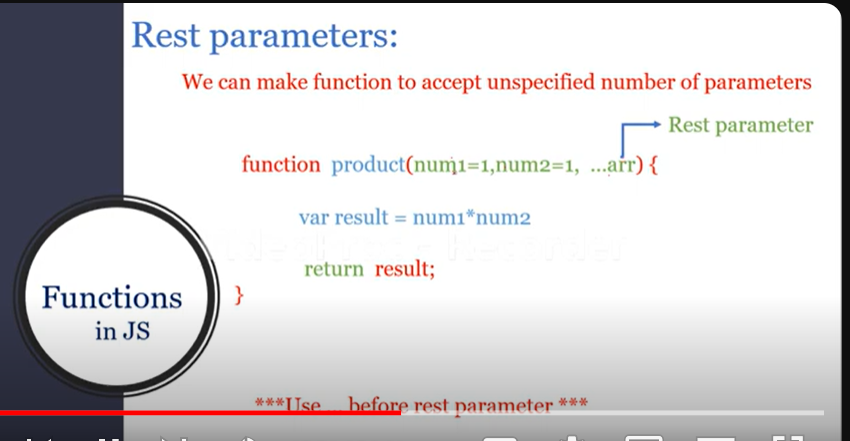
SCOPE

LOCAL SCOPE; inside the function,BLOCK

GLOBAL SCOPE: OUTSIDE THE FUNCTION ,BLOCK

BLOCK LEVEL SCOPE: INSIDE THE BLOCK

REST PARAMETERS



function product(m,n,...arr){

    var pro=m\*n;

    for(p of arr){

       pro=pro\*p;

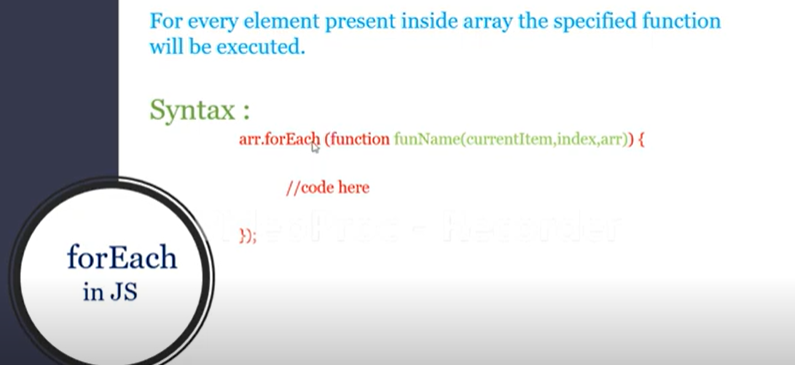
    }

    console.log(pro);

}

product(1,2,3,4,5,6,7,8,9);//362880

for each



var salaries =[10000,20000,30000,4000]

salaries.forEach(increment)

function increment(sal,index){

    let increment=sal\*0.1;

    salaries[index]=sal+increment;

}

console.log(salaries);

SET:->

SET is a collection of unique values

Array is collection of values

Syntax for creating set

let ss=new Set();

ex

let aa='hyderaabaad';

let ss=new Set(aa);

console.log(ss)

ss.delete('h')//deleting specfic item

console.log(ss)

var statuss=ss.has('d');//it finds the element is avaliable or not

console.log(statuss)

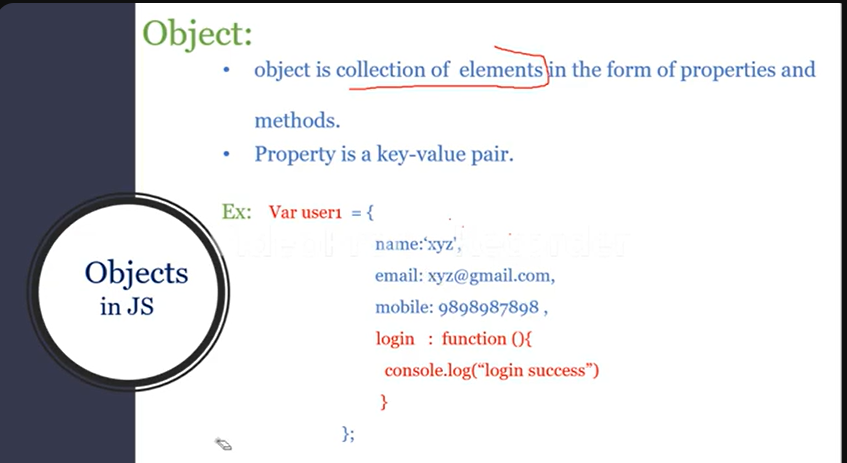
console.log(ss.size);//know the size

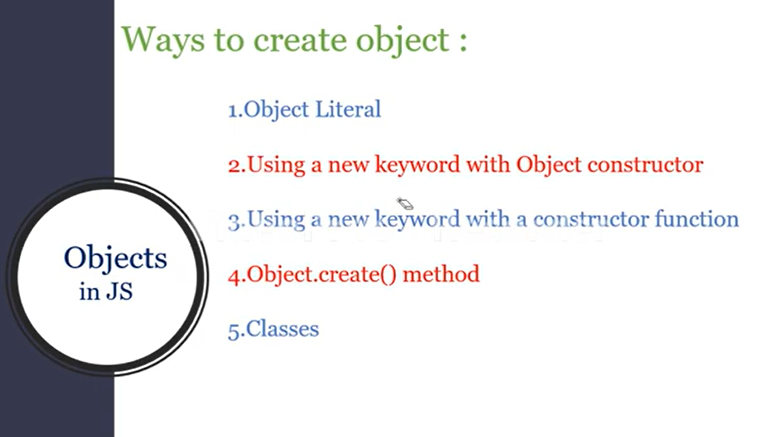
//ss.clear()//it clears the total data

var itr=ss.entries();

console.log(itr.next());

OBJECTS:->





OBJECT LITERAL:->

let movie={

    name:'kalki',

    director:'nagashwin',

    realse:'may'

}

console.log(movie)

console.log(movie['name']);//kalki

movie['budget']='500cr'

movie.hero='prabhas';

Using new operator Object Constructor:

let movie=new Object();

ex:

let movie=new Object();

movie.year=2024

movie.banner=’vyjayanthi’;

console.log(movie);

using new operator with constructer function:

function fun(a,b,c){

    this.age=a;

    this.name=b;

    this.place=c;

}

let rr=new fun(24,'sai','cmk')

console.log(rr)

Object.create method:

Let mov=Object.create(rr);//refer above example

It is likewise or copying another object