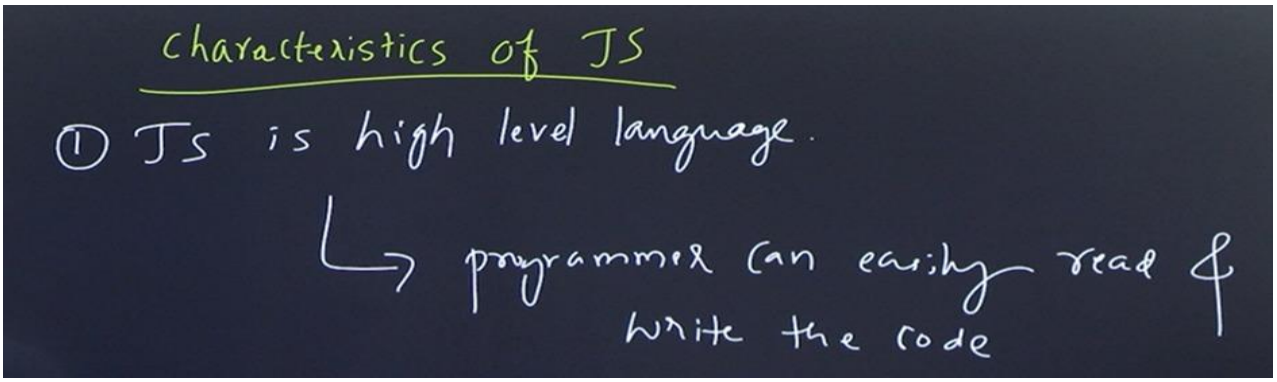
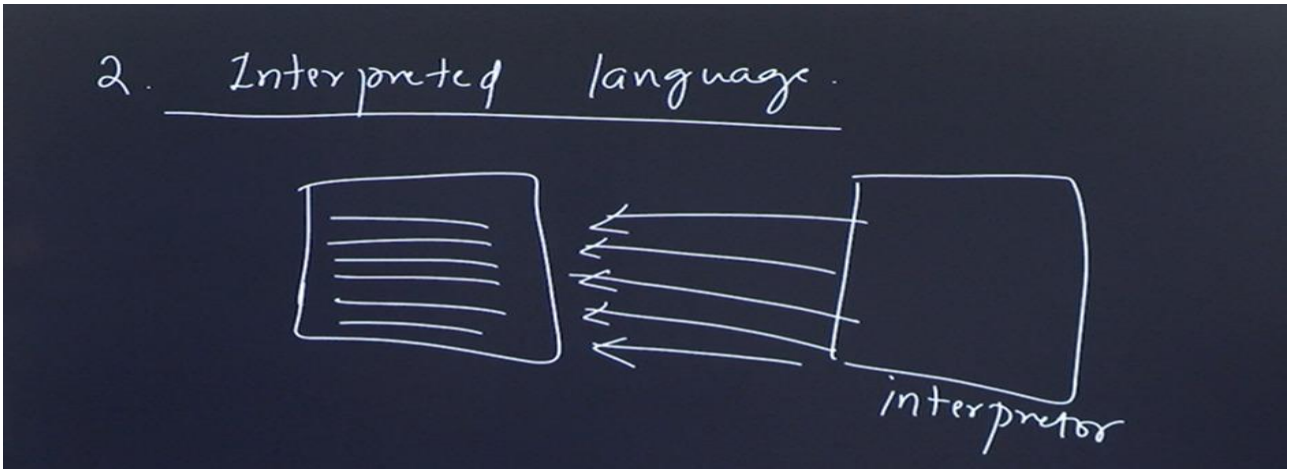
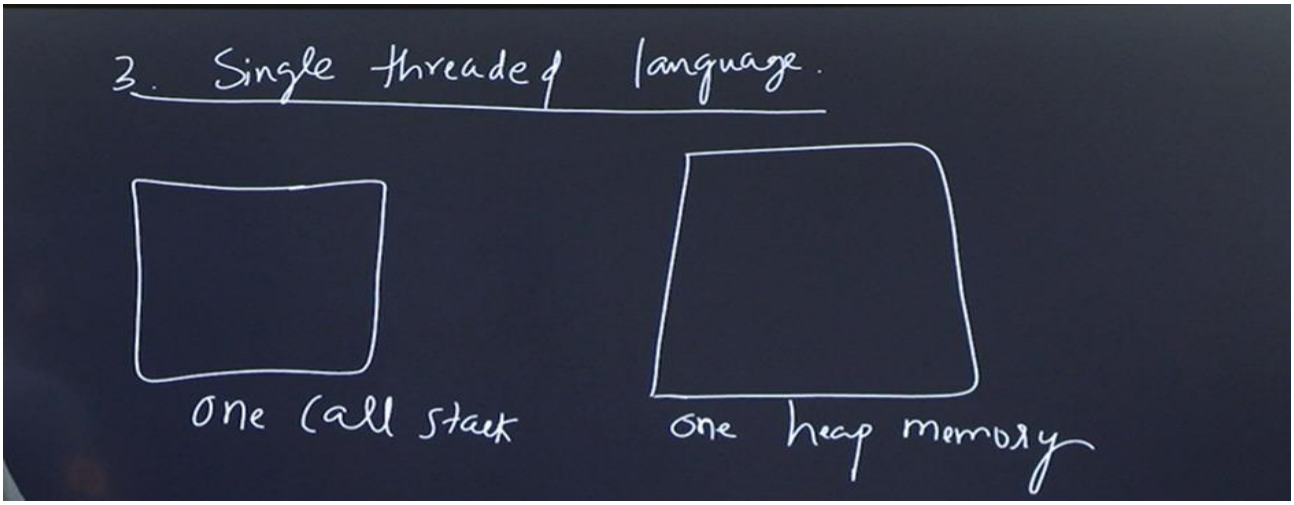
Characteristics Of JS





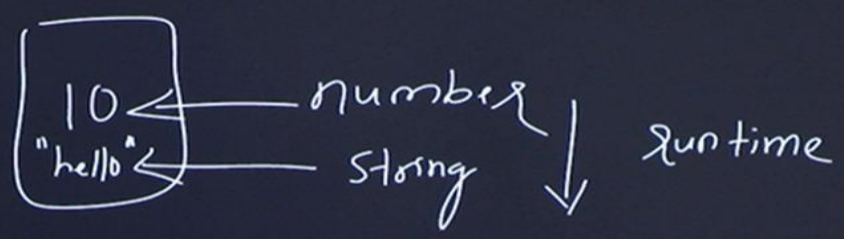


4. loosely typed language

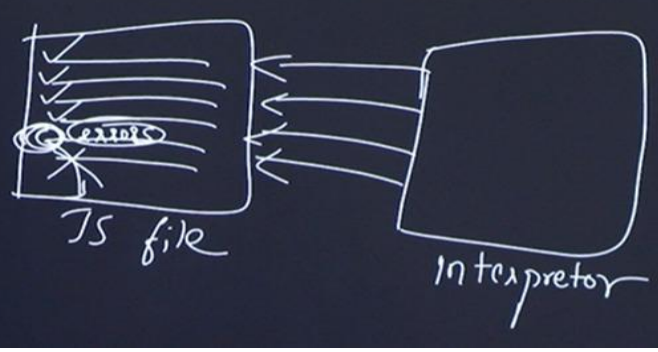
~~int~~

~~string~~

4. Dynamic language



6. synchronous language



1. Scripting language → no compiler
2. H.L.L → user friendly
3. interpreted → code gets executed line by line
4. Single threaded → one callstack and one heap memory
5. loosely typed → no particular variable to store values
6. dynamic language → datatype during runtime
7. Object based language → ??

JS file

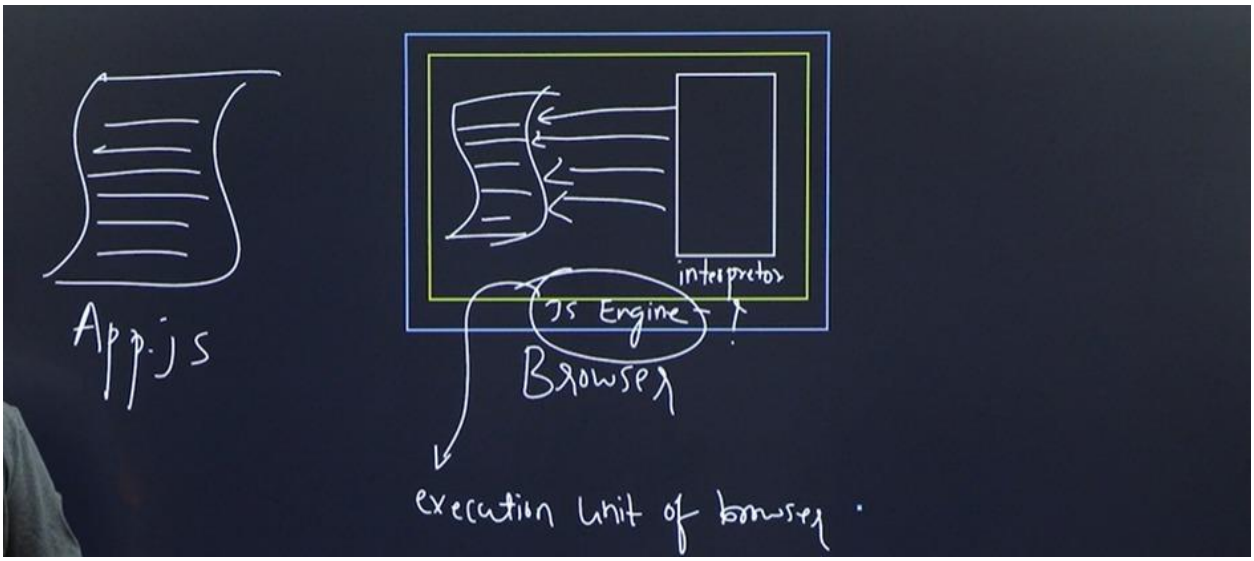
index.html

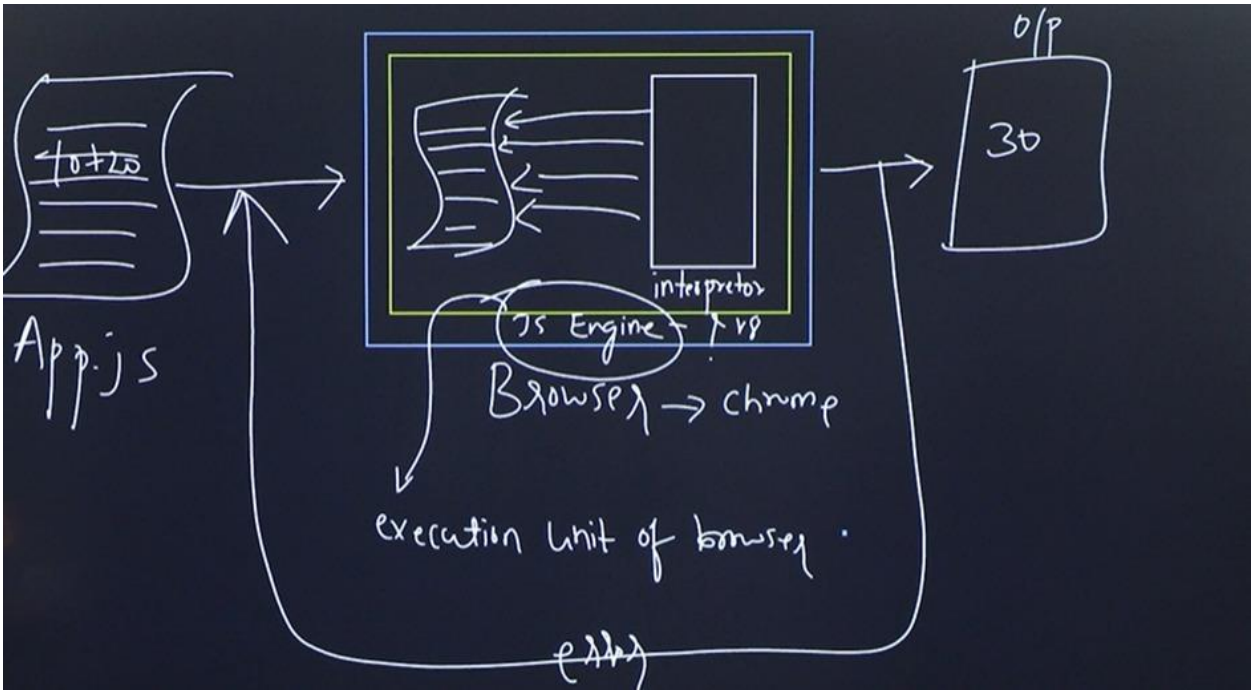
style.css

App.js → Extension

```
C:\Users\Alpha4>cd desktop

C:\Users\Alpha4\Desktop>mkdir javascript

C:\Users\Alpha4\Desktop>cd javascript

C:\Users\Alpha4\Desktop\javascript>code .
```

How To Run JS File





Browser	JS Engine
Google chrome →	V8
Mozilla →	Spider monkey
Microsoft internet explorer →	Chakra
Safari →	JS core

How to run JS file?

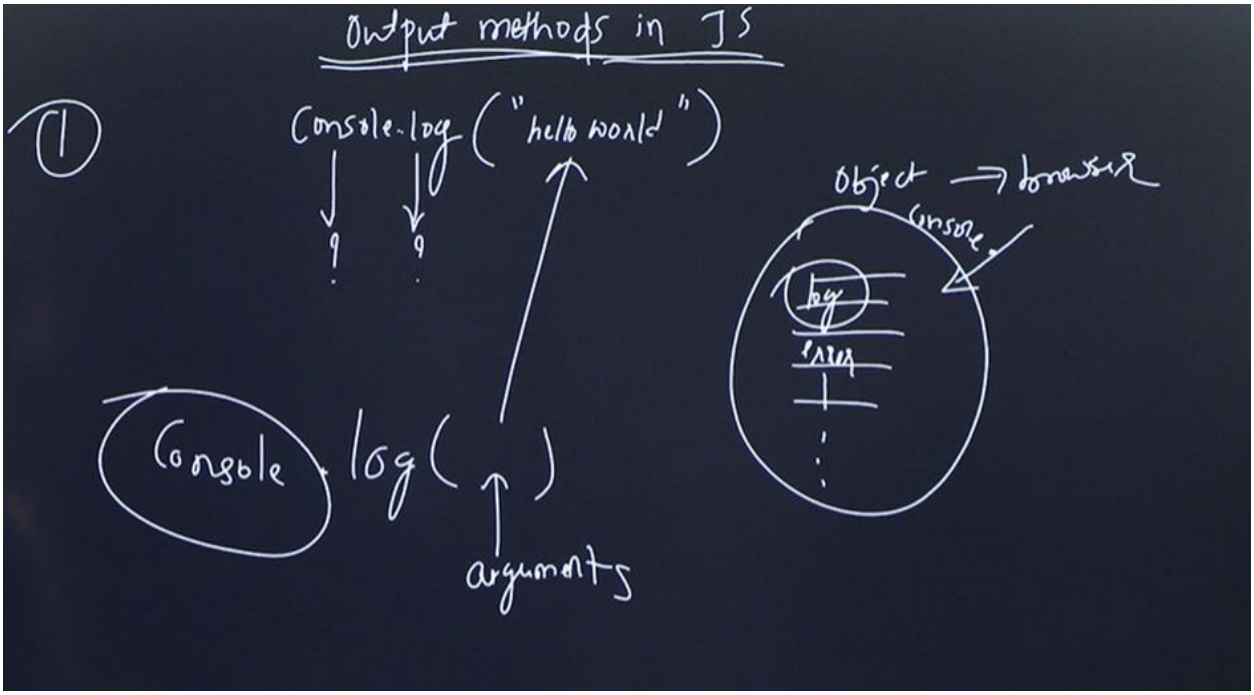
JS file

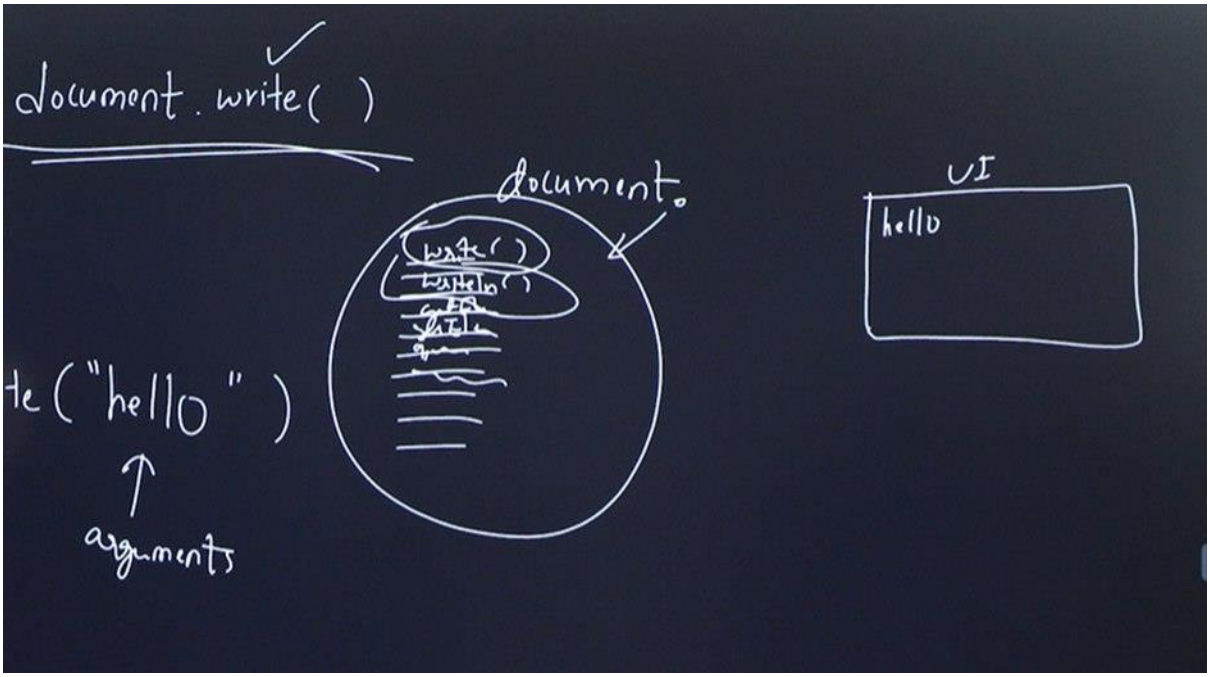
↓

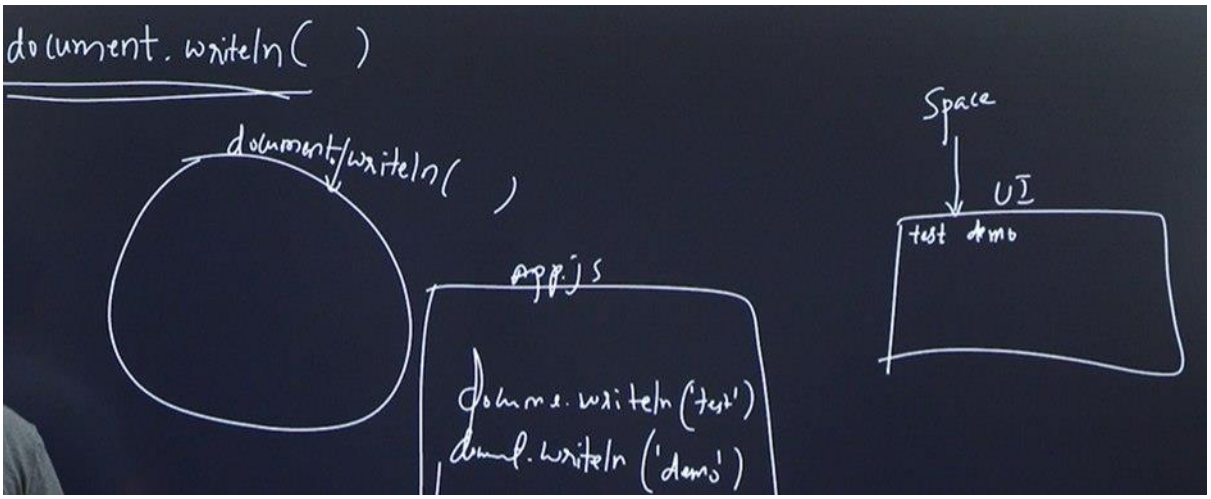
to run → inside browser → Browser

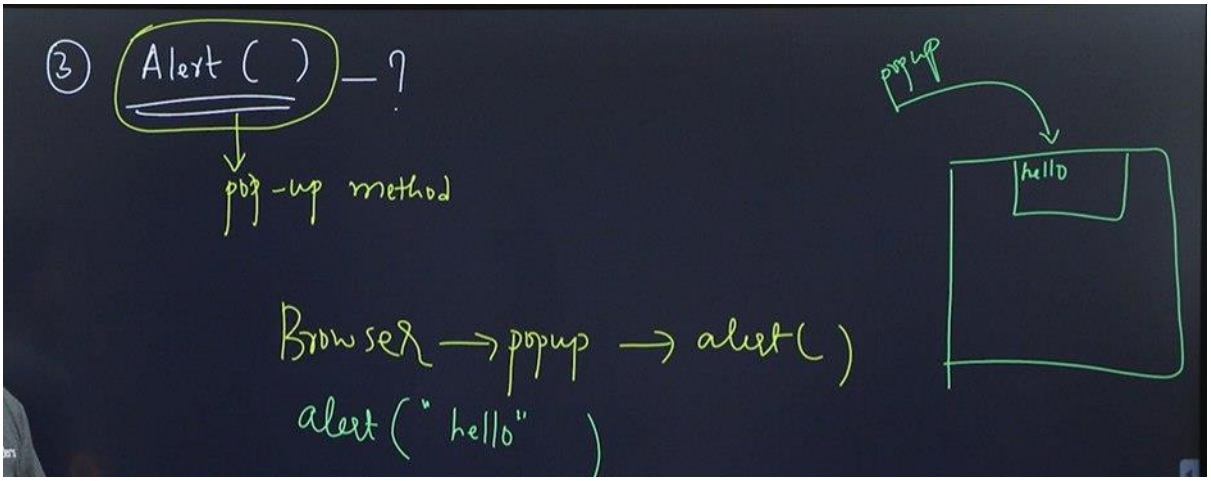
→ outside browser → Node js

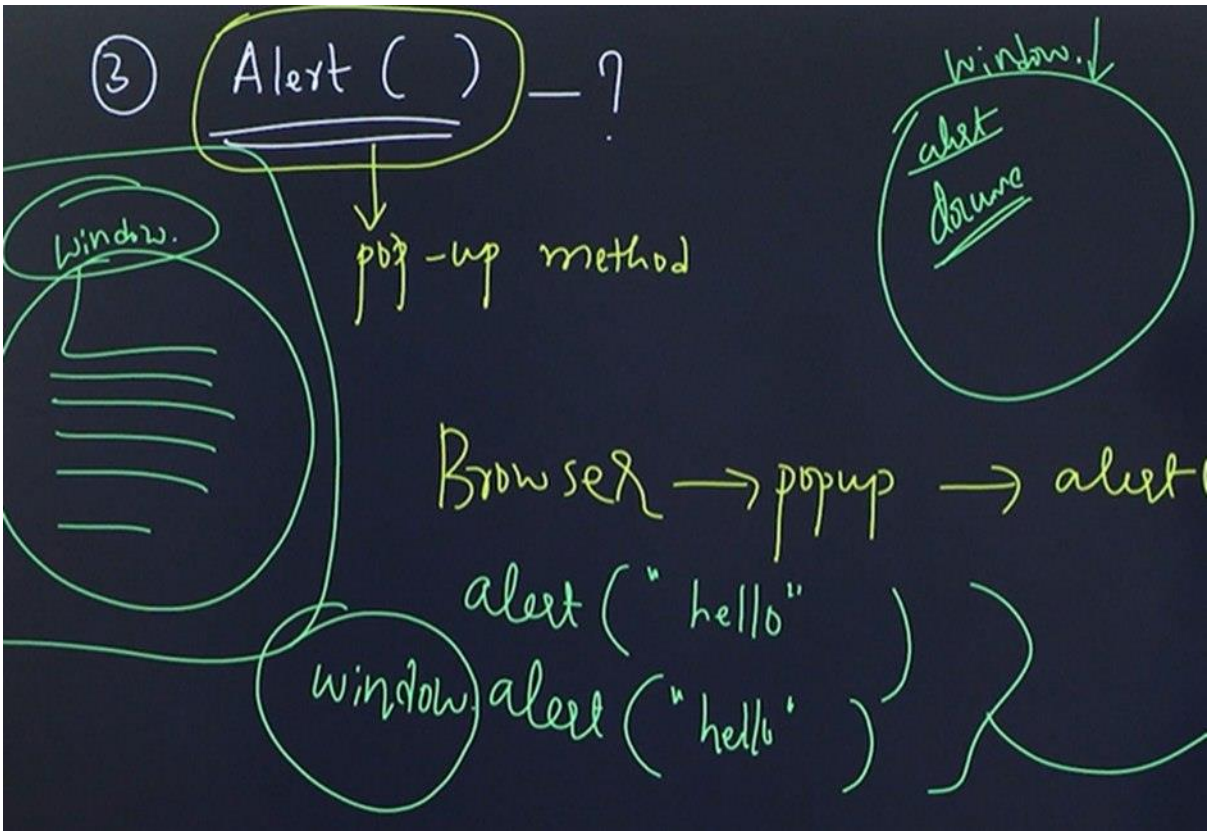
Output Methods In JS

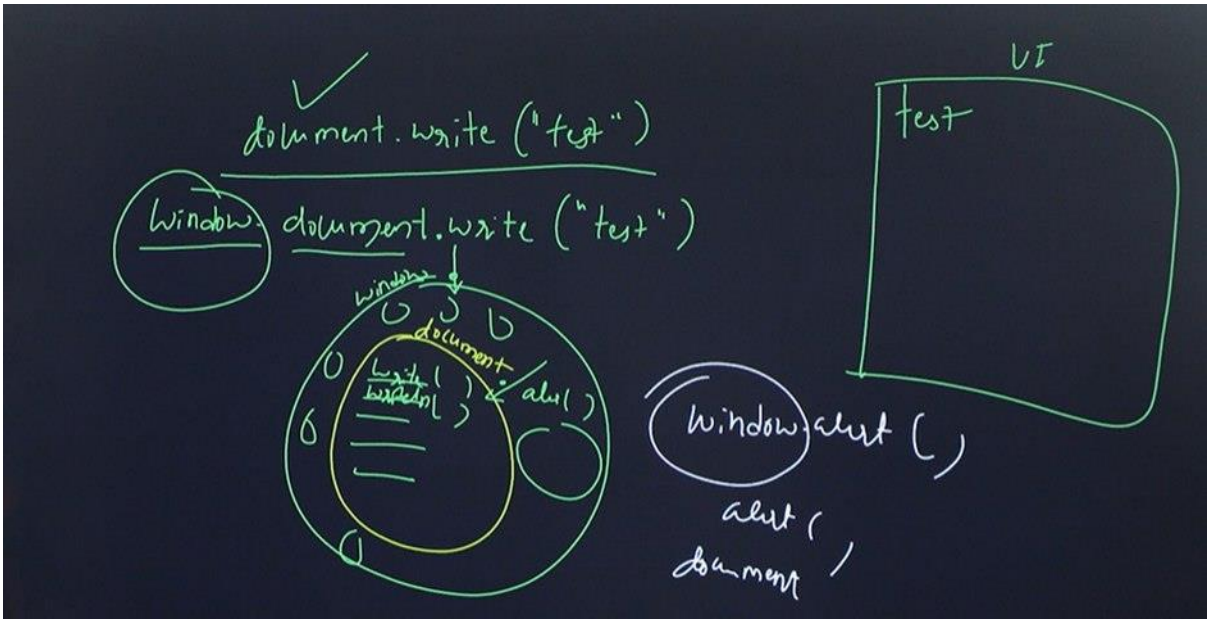





document.writeln( )
document.writeln( )
app.js
document.writeln('test')
document.writeln('demo')
Space
UI
test demo

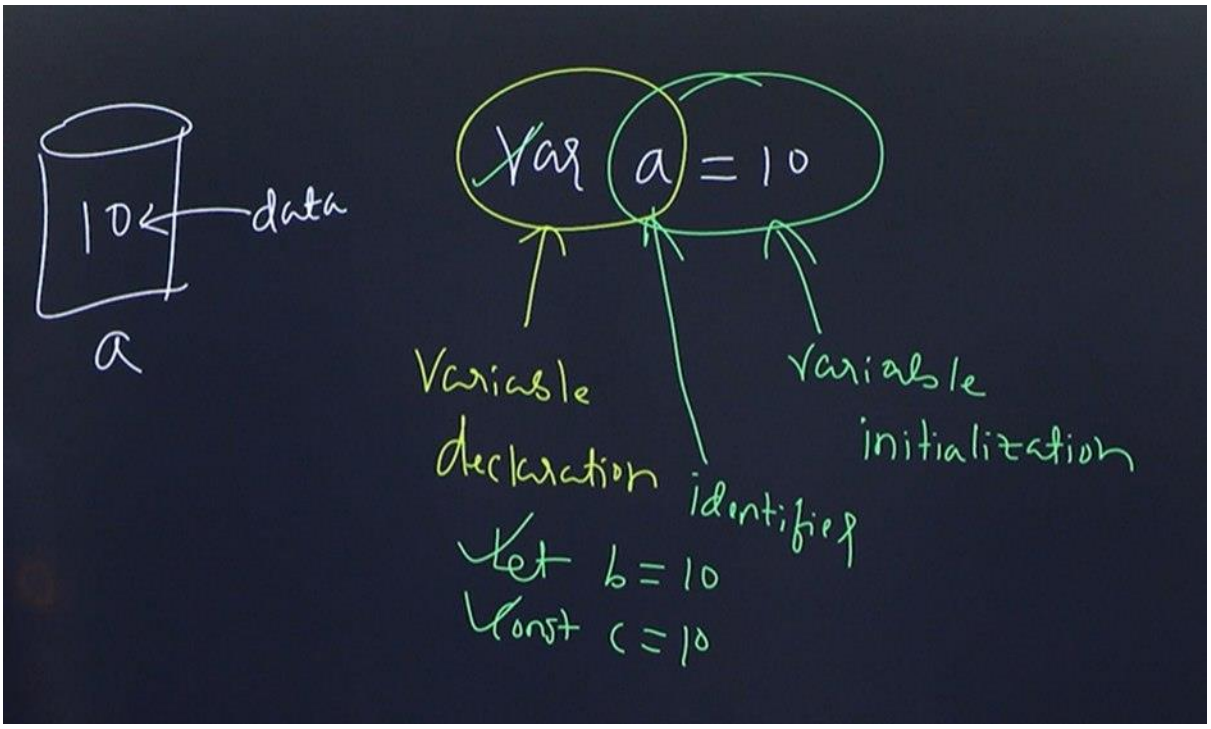

③ Alert ( ) — ?
pop-up method
popup
hello
Browser → popup → alert( )
alert(" hello")

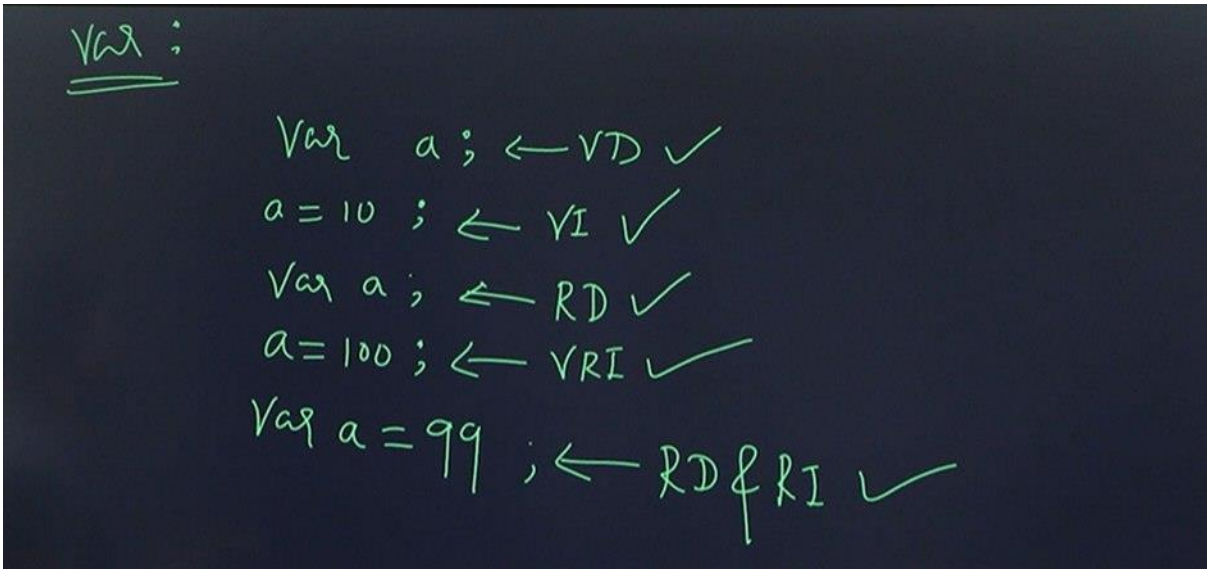

③ Alert ( ) — ?
pop-up method
window.
window.
alert
document
Browser → popup → alert
alert (" hello")
window.alert (" hello")


document.write ("test")
window.document.write ("test")
window
document
alert( )
UI
test
window.alert ( )
alert
document

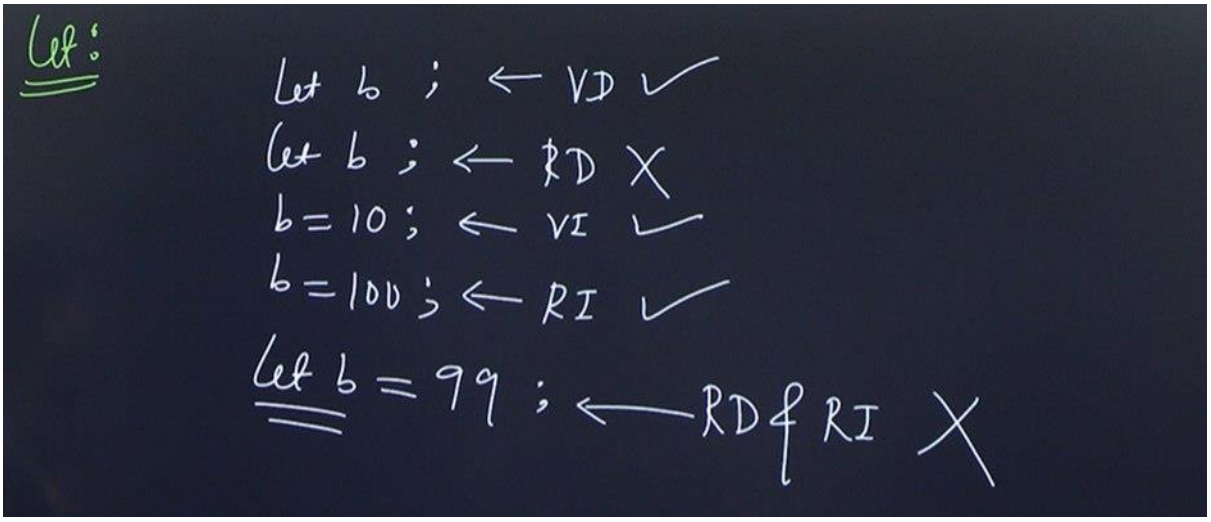
Declaration Of Variable

Var :

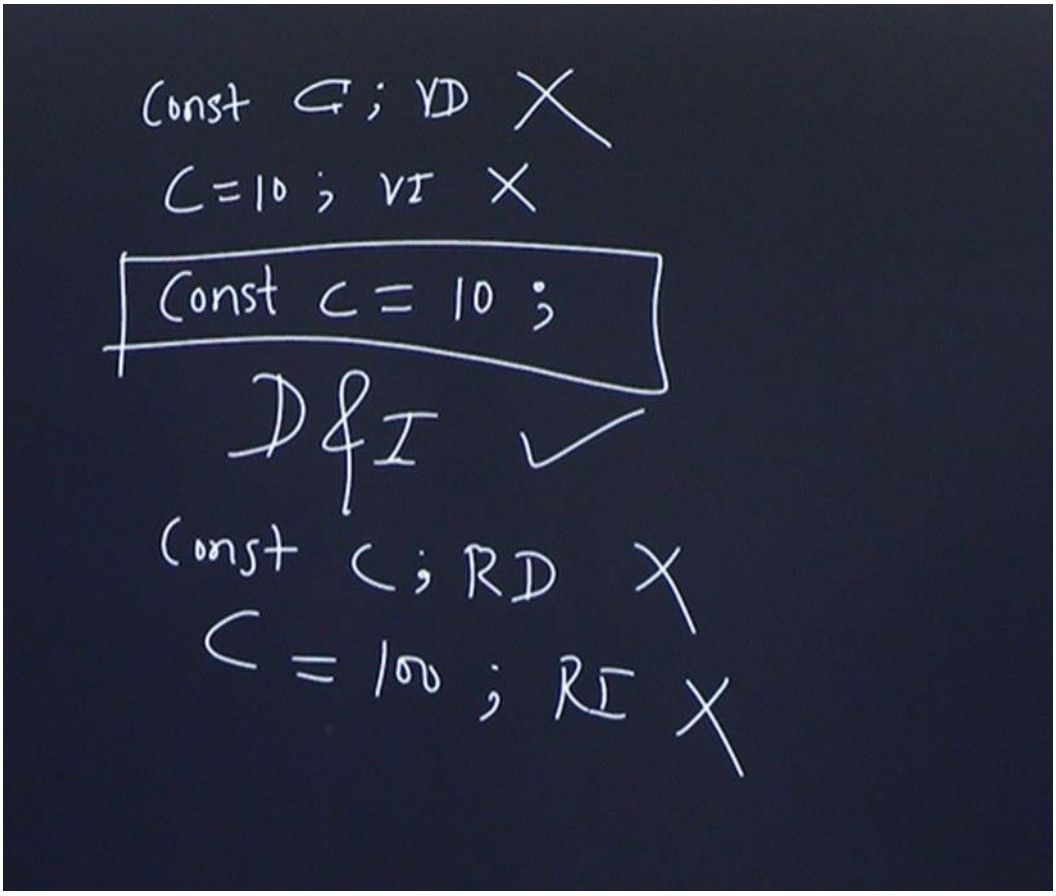




let :


Let:
let b ; ← VD ✓
let b ; ← RD X
b=10 ; ← VI ✓
b=100 ; ← RI ✓
let b = 99 ; ← RD & RI X

Const :


Const C ; VD X
C=10 ; VI X
Const c = 10 ;
D & I ✓
Const C ; RD X
C = 100 ; RI X

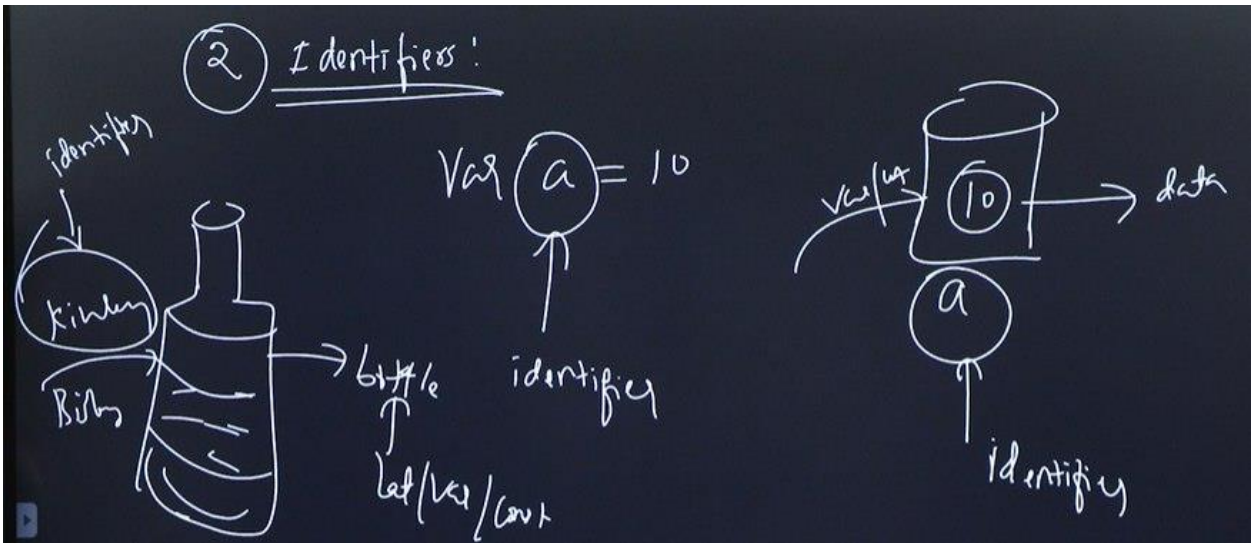
Tokens

1. keywords
2. identifiers
3. literals/values

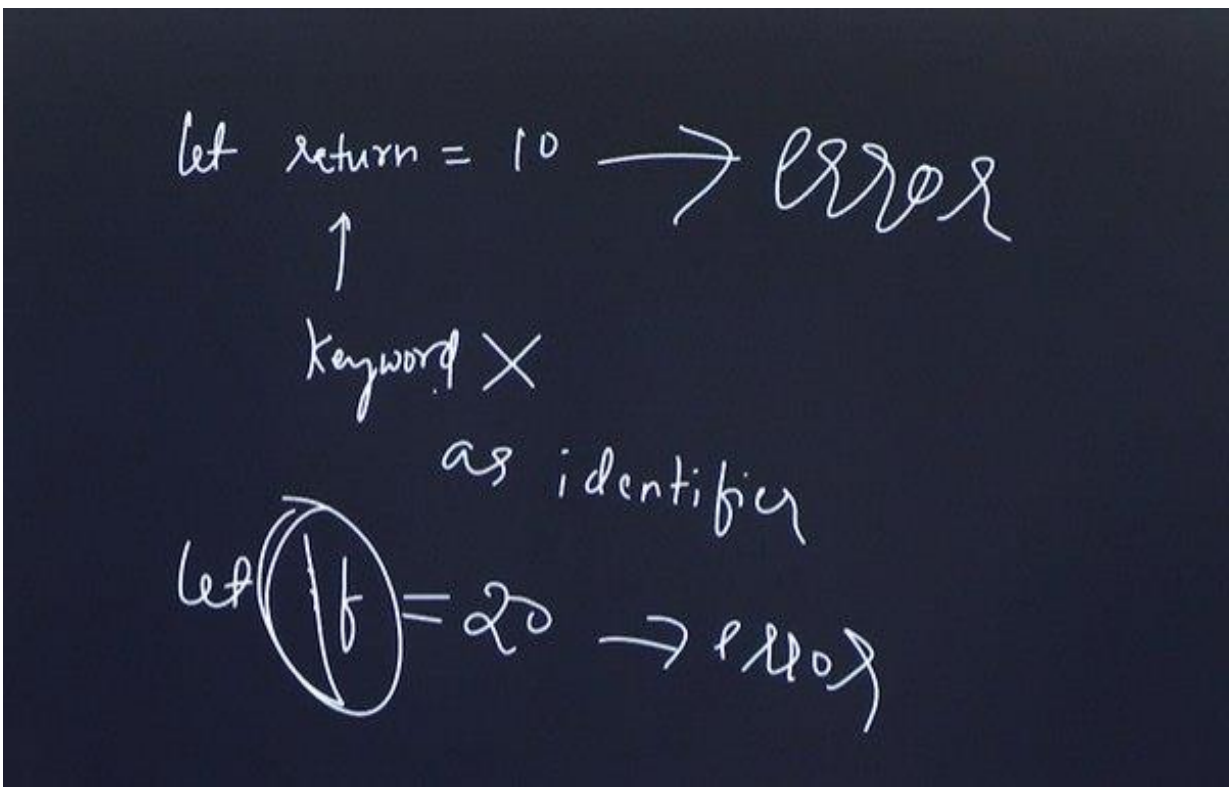
1. Keywords :

60+ → JS
keywords

this
return
if-else
switch
for
let
var
const
:



Rules Of Identifier :



(2)

let 10 = 20 → error

↓

number alone – ?

(3)

let a10 = 20 ✓

alpha-numeric

let empName = "Raj" → error

Space

let empName = 'Raj'

Lower camel case

$ and _

let * = 10 → error

Special Character

let $ = 10 ✓

let emp_name = 'Raj

Under Score

① Keyword ✓

② identifier ✓

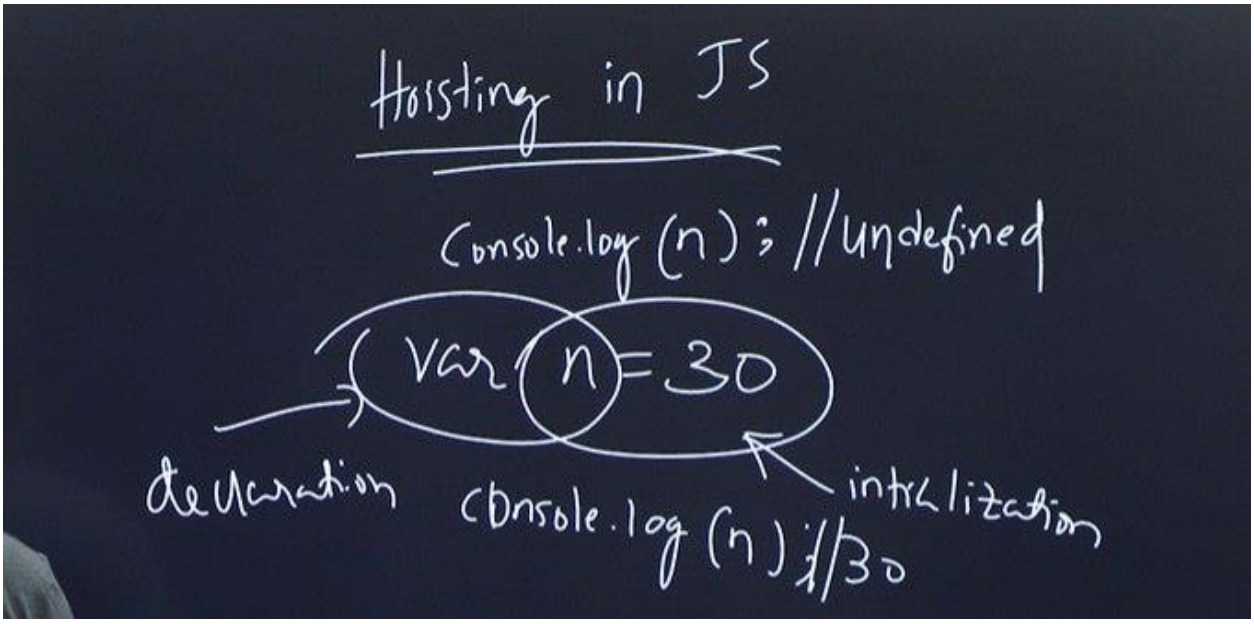
③ literals / values → data assigned to a variable

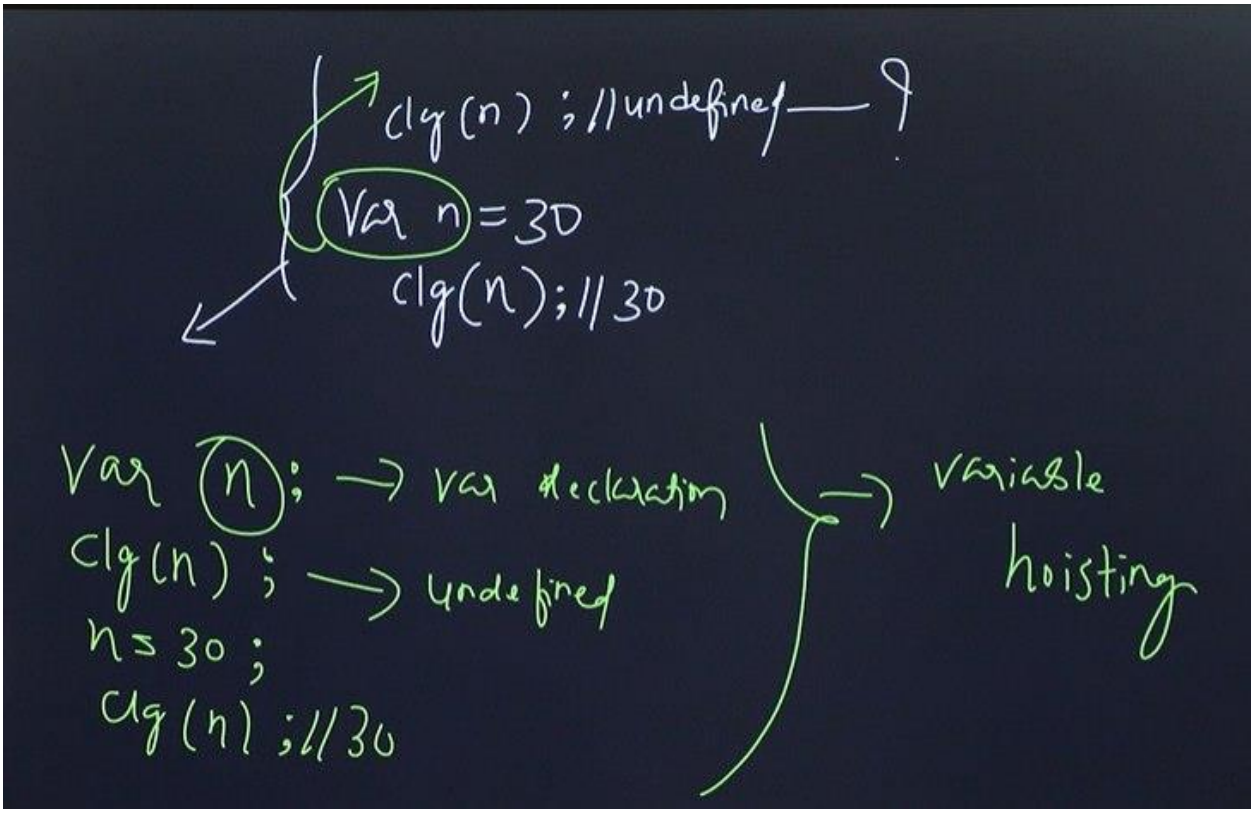
10

'Staff'

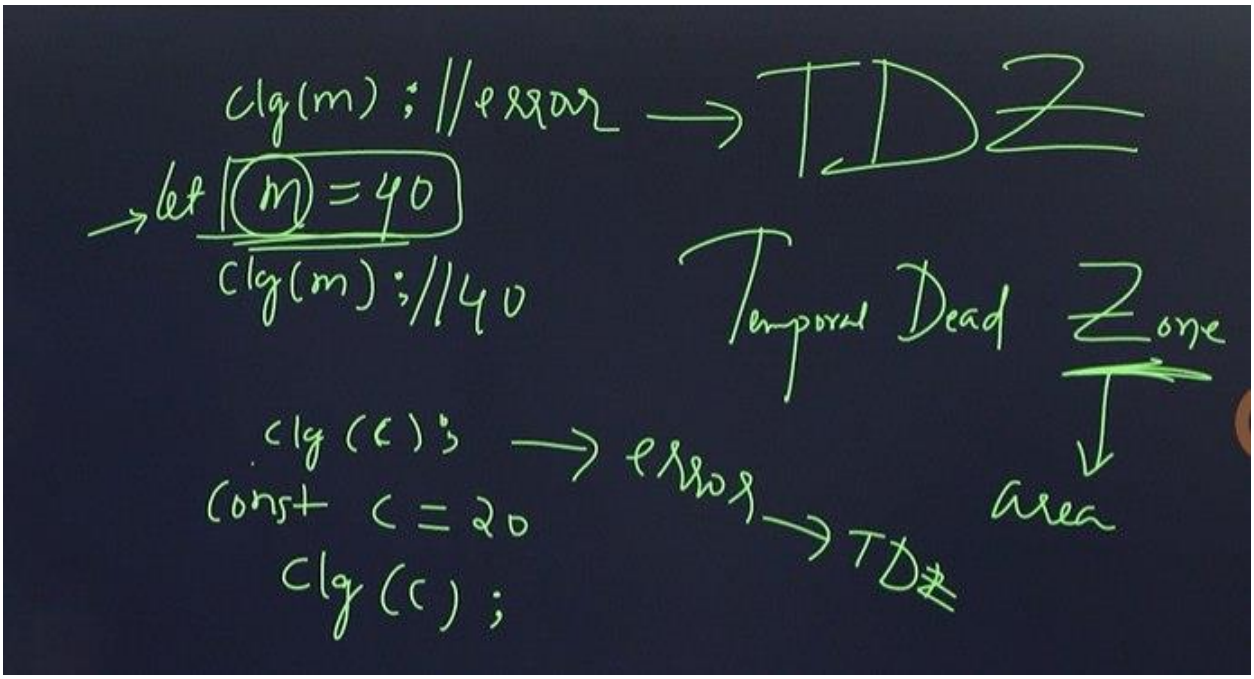
true

Hoisting In JS

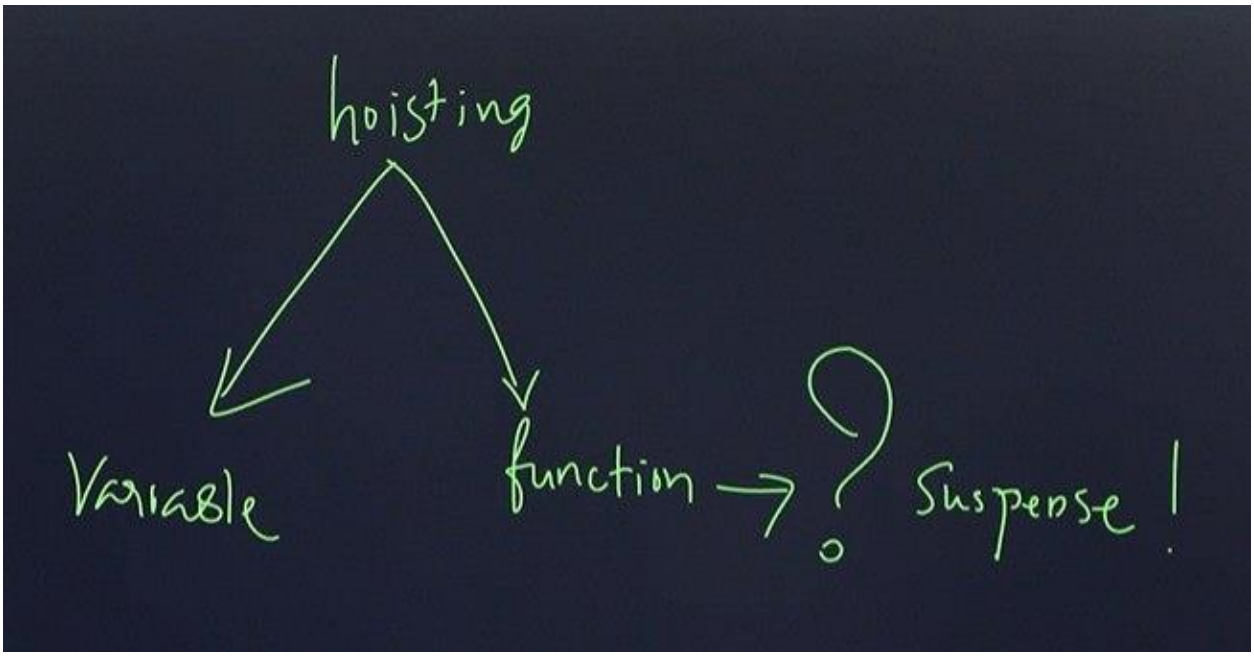




Note : It is done by js engine .



TDZ = > can’t access variable without initializing .



Statements

statements.js × index.html

statements.js

```
1.Conditional statements
2.Looping statements
3. Jump statements
*/
// !1.Conditional statements
//?if
if (true) {
  console.log("i will be executed when the given condition is true");
}
if (false) {
  console.log("i will be executed when the given condition is true");
} else {
  console.log("i will be executed when the given condition is false");
```

```
tatements.js > ...
//?switch
let num = Math.round(Math.random()
* 3);
console.log(num);
switch (num) {
  case 0:
    console.log("zero");

    break;
  case 1:
    console.log("one");
    break;
  case 2:
    console.log("two");
        break;
    case :3

  default:
    break;
```

```
//!2. looping statements
// for (let i = 1; i < 5; i++) {
//   console.log(i);
// }

//!3. Jump statements
// for (let i = 1; i < 5; i++) {
//   if (i == 3) {
//     break;
//   }
//   console.log(i);
// }
for (let i = 1; i < 5; i++) {
  if (i == 3) {
    continue;
  }
  console.log(i);
```

Operators

```
//!1.Arithmetic operators:
console.log(10 + 5);
console.log(10 - 5);
console.log(10 / 5);
console.log(10 % 5);
console.log(2**2)
console.log(3 ** 2);
```

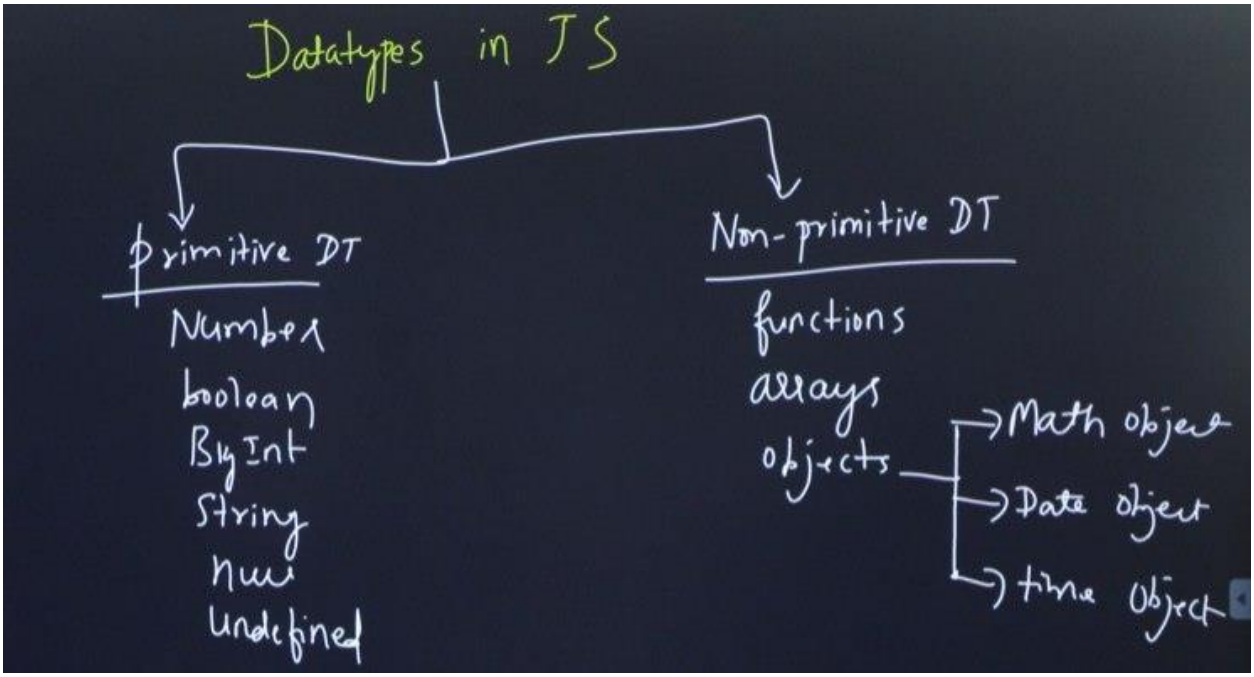
```
//!2.Assignment operators
let x = 10; //assignment operator
console.log((x += 10)); //x=x+10,
10+10=20
let y = 30;
console.log((y -= 15)); //
y=y-15=30-15=15;

let z = 2;
console.log((z *= 4)); //
z=z*4=2*4=8

let p = 40;
console.log(p /= 10);//p=p/10=40/
10=4
```

```
//!3.Comparison operators:
// == and ===
// == checks only value
// === checks both value and
datatype

console.log(10 == 10);
console.log(10 == "10");
console.log(10 === "10");
console.log(10 < 5);
console.log(10 > 5);
console.log(5 <= 5);
console.log(5 >= 5);
```

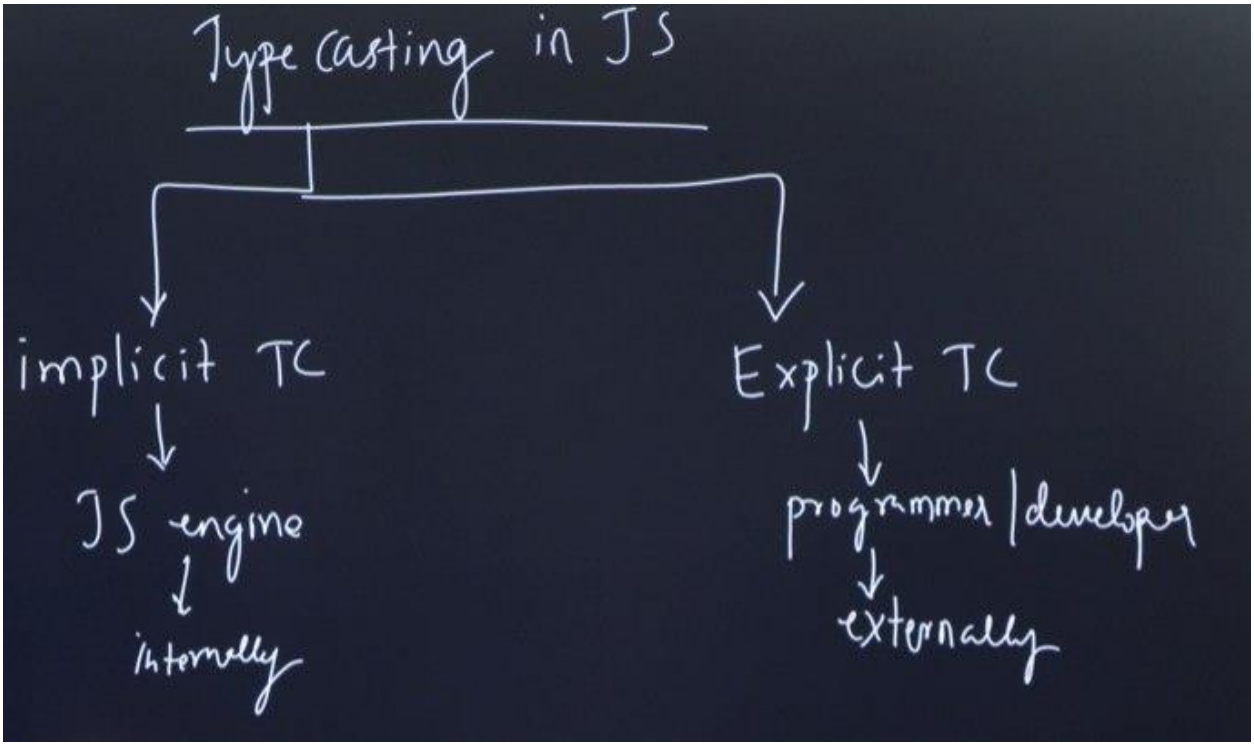
```
//!4. logical operator
if (10 == 10 && 10 === 10) {
  console.log("true");
} else {
  console.log("false");
}

if (10 == 10 || 10 === "10") {
  console.log("true");
}

if (10 != 5) {
  console.log("true");
}
```

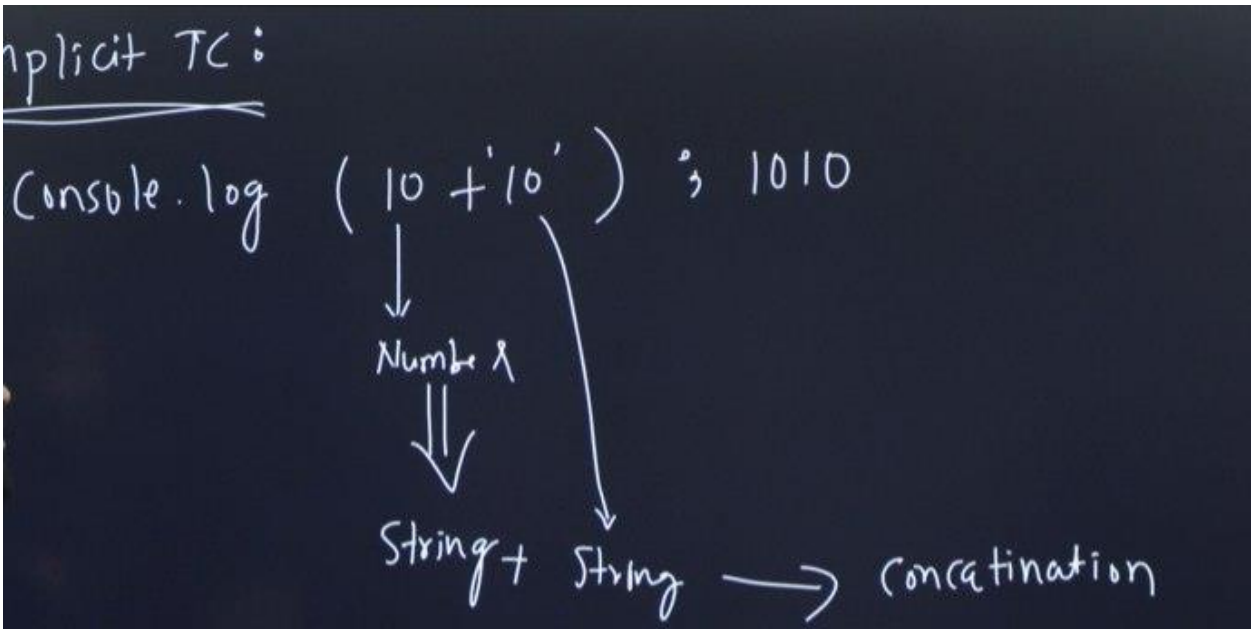
```
// ternary operator: ?

let m = 5;
m == 5 ? console.log("true") :
console.log("false");

let c = true;
c != true ? console.log("true") :
console.log("false");
```

Data Type


Datatypes in JS
Primitive DT
Number
boolean
BigInt
String
null
Undefined
Non-primitive DT
functions
arrays
objects
Math object
Date object
time Object

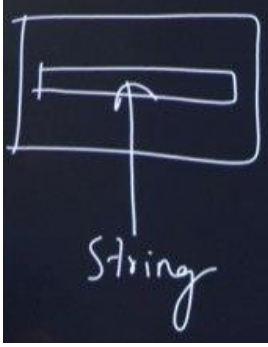
Type Casting





Explicit TC:

```
let n1 = prompt("enter n1")   <- 20
let n2 = prompt("enter n2")   <- 30
clg(n1+n2);  ~~50~~  2030
```




```
let n1 = Number(prompt("enter n1"))   <- 20
let n2 = Number(prompt("enter n2"))   <- 30

clg(n1+n2); 50
```

String to number

Type of

```
JS typeof.js > ...
1   let num = 1000;
2   console.log(typeof num);
3
4   let str = "hello";
5   console.log(typeof str);
6
7   let isEmployee = true;
8   console.log(isEmployee);
9
10  let arr = [10, 20];
11  console.log(typeof arr);
12  console.log(Array.isArray(arr));
13
14  let fun = function () {
15      console.log("i m fun")
16  }
17  console.log(typeof fun);
18
19  let obj = {};
20  console.log(typeof obj)
```

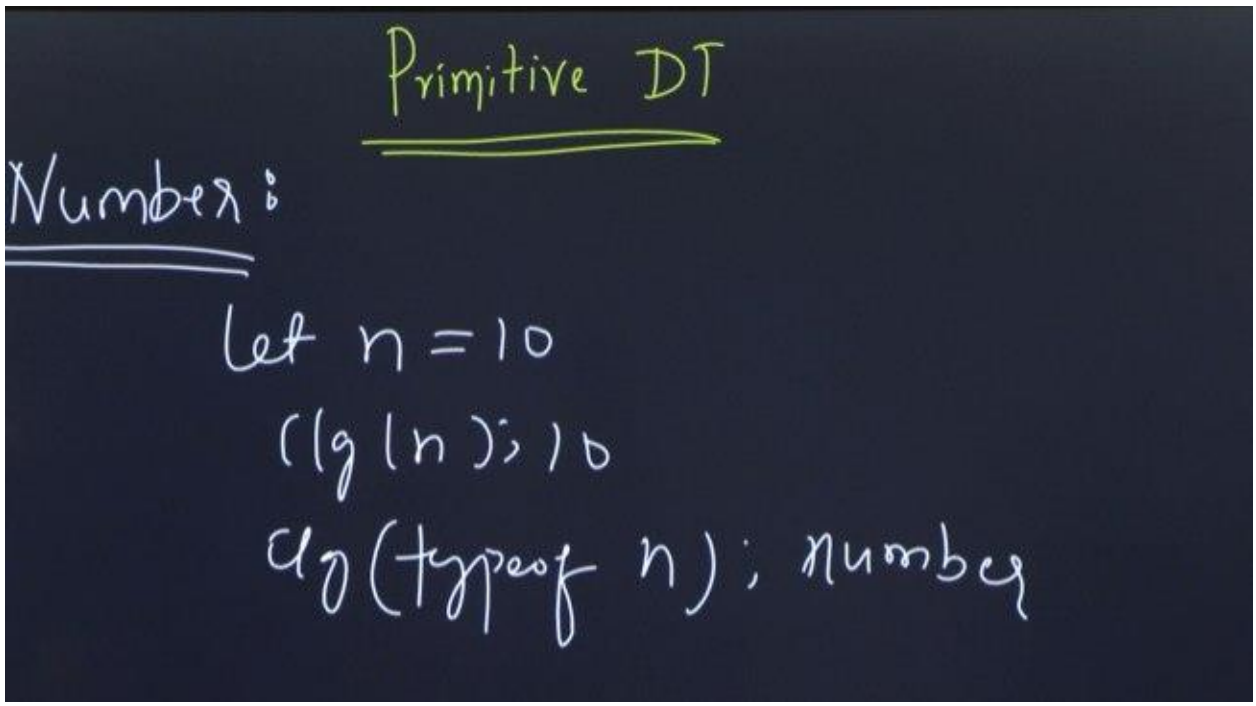
```
let n = 21212121;
console.log(typeof n);
console.log(typeof typeof n);

let x = false;
console.log(typeof x);
console.log(typeof typeof x);
```

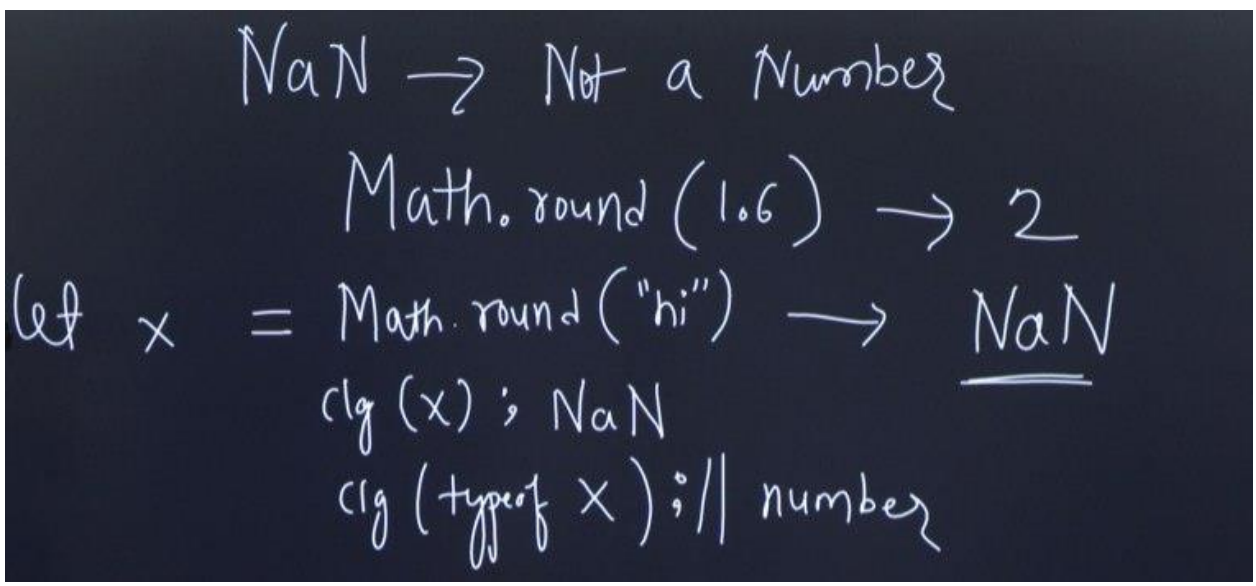
```
Elements  Cons
top ▾  Filt
No Issues
number
string
boolean
string
DevTools failed to load
```

Note : Tow times type of => string

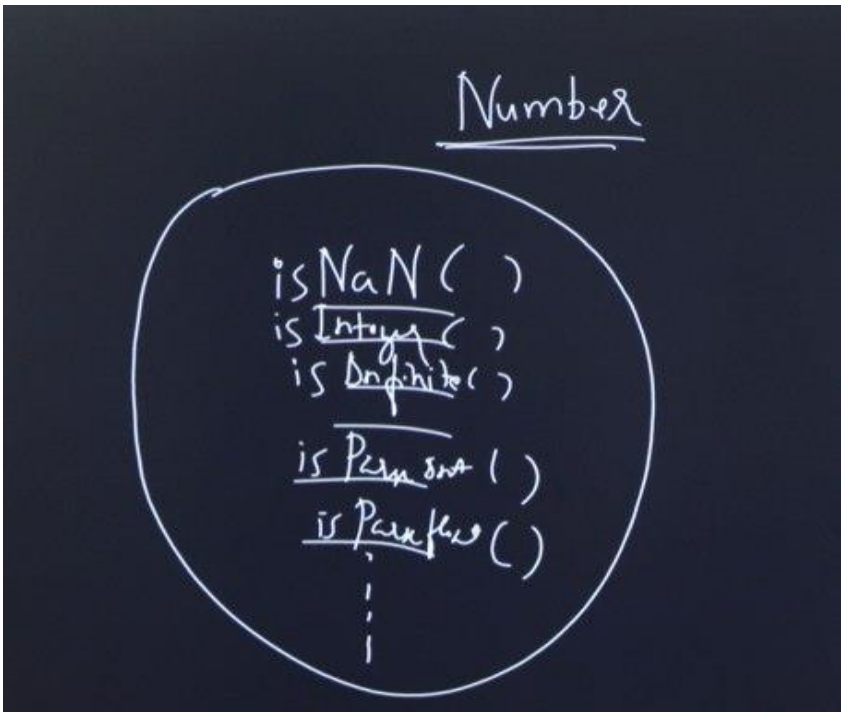
Premitive DataType

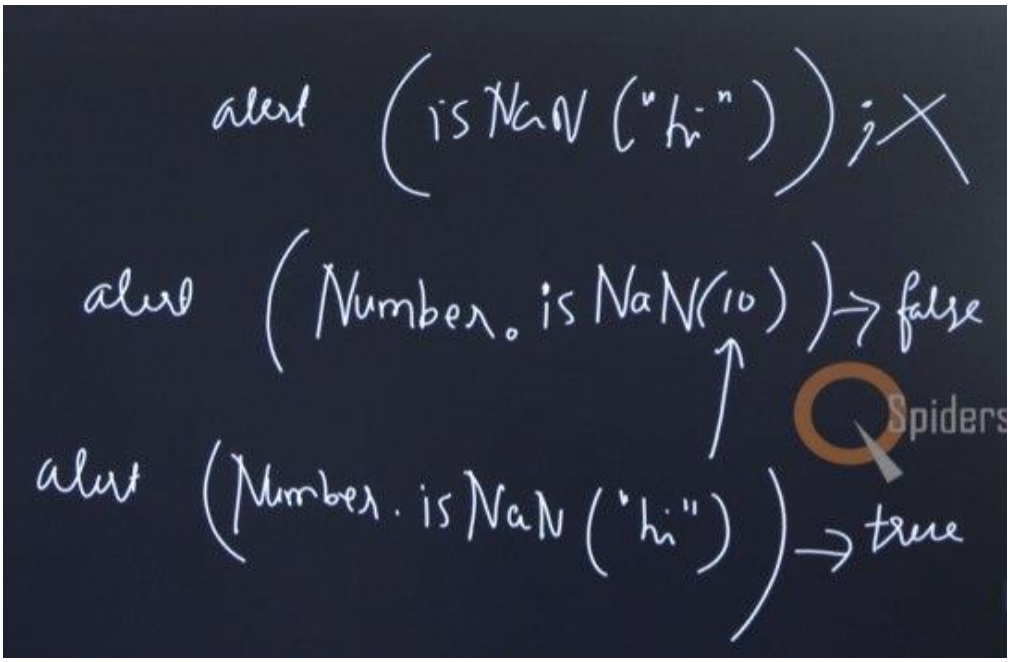
1. Number





Number Object And Its Methods





alert (Number.isInteger(10.5)) → false

alert (Number.isInteger(10)) → true

alert (Number.isFinite(10/5)) → true

alert (Number.isFinite(10/0)) → false

```
console.log(Number.isNaN(32));

let res = Math.round("demo");
console.log(res);
console.log(Number.isNaN(res))
```

Elements Co

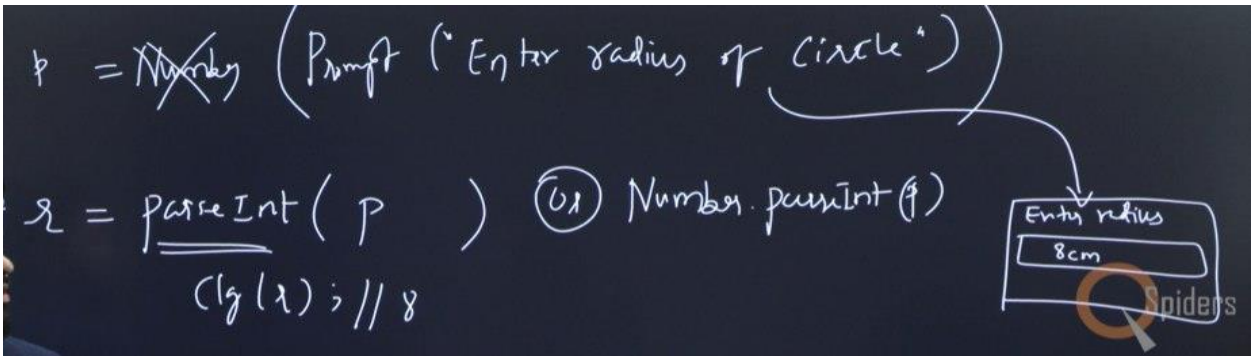
top ▾

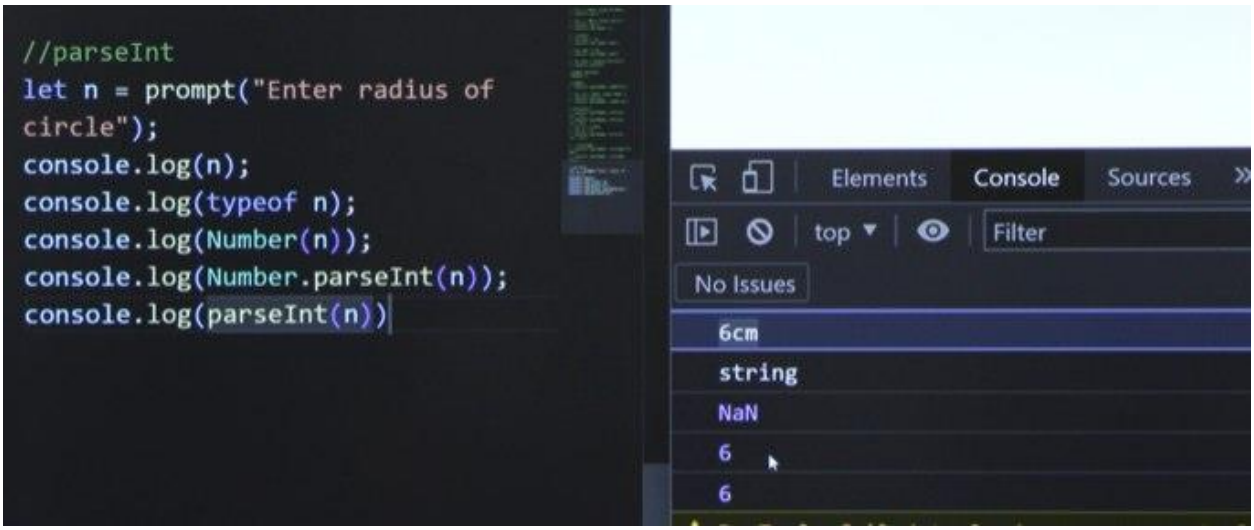
No Issues

```
false
NaN
true
```

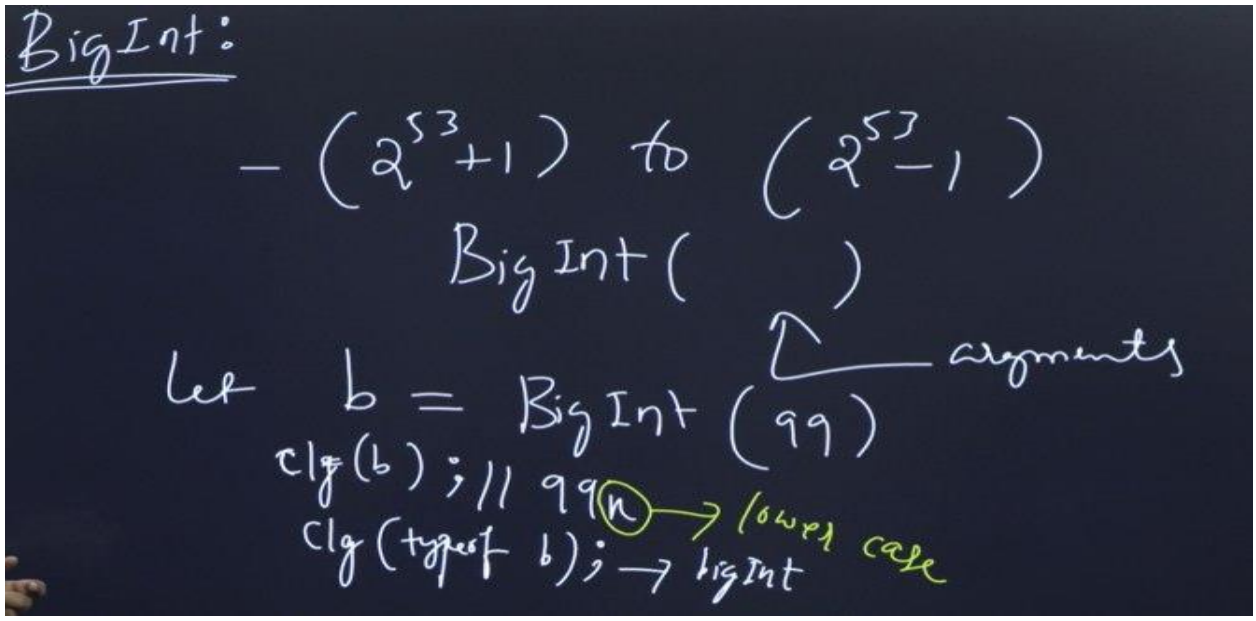
DevTools failed to lo

ParseInt ():


p = Number (Prompt ("Enter radius of circle"))
r = parseInt (p)
(or) Number.parseInt (p)
console.log (r); // 8
Enter radius
8cm
Spiders


//parseInt
let n = prompt("Enter radius of circle");
console.log(n);
console.log(typeof n);
console.log(Number(n));
console.log(Number.parseInt(n));
console.log(parseInt(n))
Elements
Console
Sources
top
Filter
No Issues
6cm
string
NaN
6
6

2. BigInt



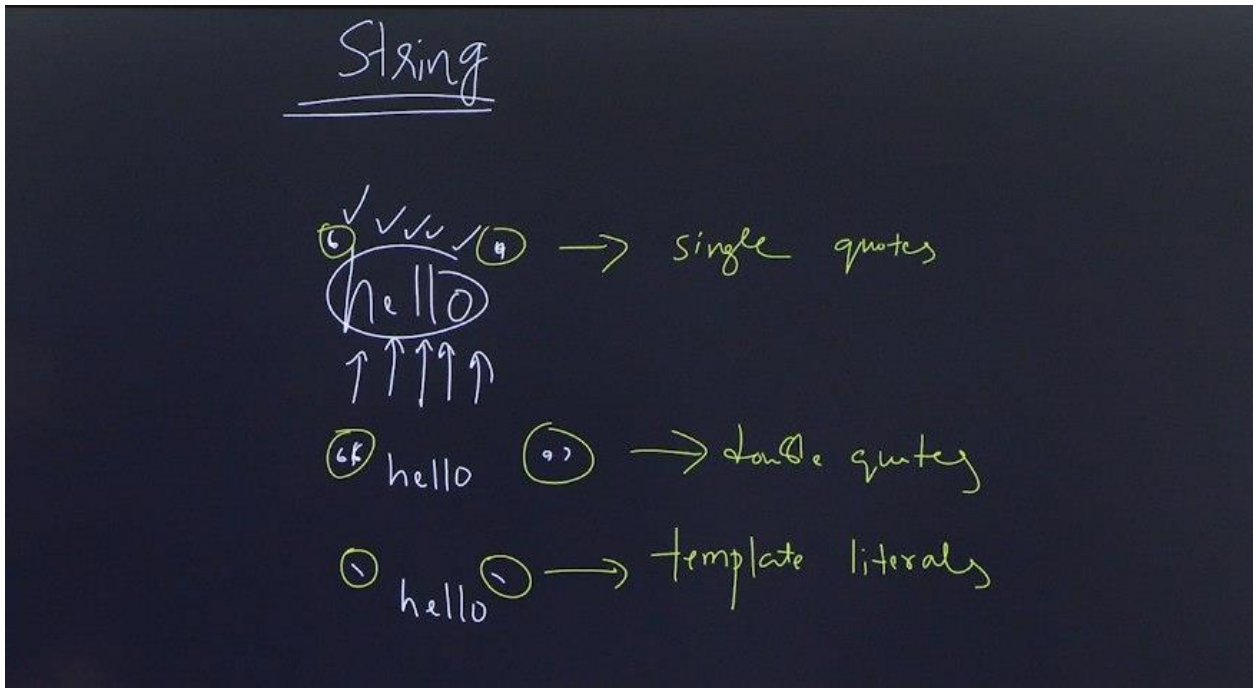
Note : BigInt() is a constructor .

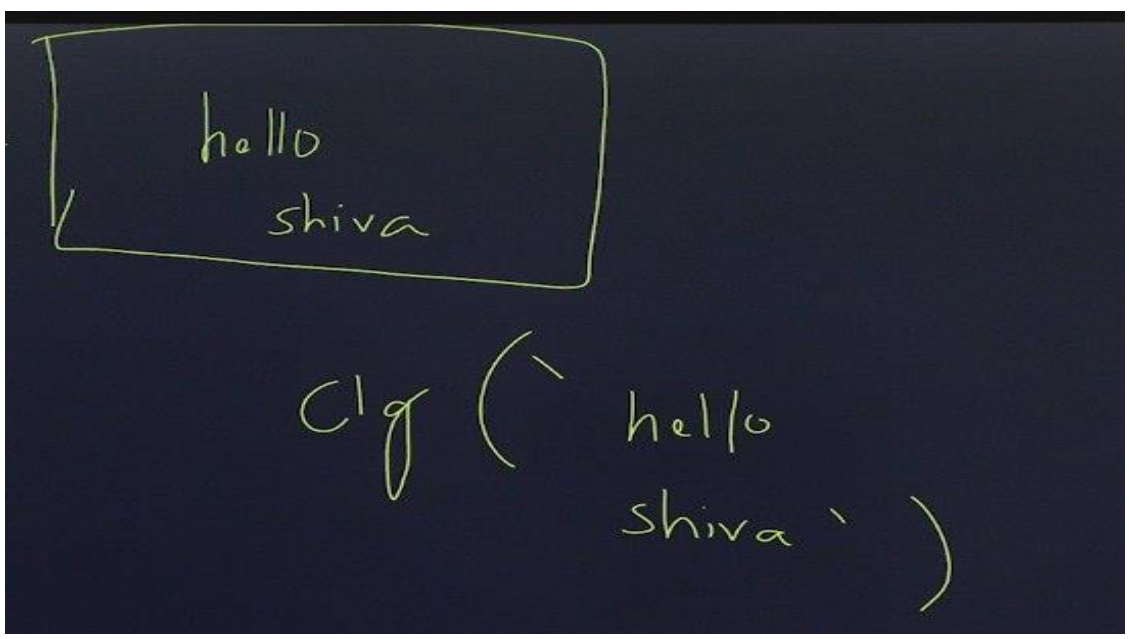
3. Boolean

```
let booleanVal = true;
console.log(booleanVal);
console.log(typeof booleanVal);

let str = "hello";
let booleanConstructor = Boolean(str);
console.log(booleanConstructor);

let number = 123;
console.log(Boolean(number));

console.log(Boolean(0));
console.log(Boolean(""));
console.log(Boolean(null));
console.log(Boolean(undefined));
console.log(Boolean(NaN));
```

Note : 12-16 ans will be false .

4. String


String
hello
single quotes
hello
double quotes
hello
template literals


hello
shiva
Clg (hello
shiva)

Interpolation

$\boxed{\$\{\ \}}$ → back ticks

Variables

Let eN = "Raj"

clg(`My name is ${ eN }`)

Different ways of creating string

① by using string literals : ' ' " " ` `

② by using string constructor :

clg $\boxed{\text{(new String("hi"))}}$ → hi

```
//!string methods
let str = "hello GUYS";
console.log(str.length);
console.log(str.toUpperCase());
console.log(str.toLowerCase());
console.log(str.indexOf("l"));//2
console.log(str.lastIndexOf("l"))//
3
console.log(str.indexOf("x"));
console.log(str.lastIndexOf("z"));
console.log(str.includes("G"));
console.log(str.includes("p"));
```

```
//slice(startIndex,endIndex);
let str = "welcome to qspiders";
let slicedItem = str.slice(0, 7);
console.log(slicedItem);
let slicedChars = str.slice(-19,
-16);
console.log(slicedChars);
console.log(str.slice(0))
console.log(str.slice(-19));
console.log(str.slice(3));
console.log(str.slice(-16));
```

Elements Console
top Filter
Default levels No Issues

welcome	string.js:74
wel	string.js:76
welcome to qspiders	string.js:77
welcome to qspiders	string.js:78
come to qspiders	string.js:79
come to qspiders	string.js:80

DevTools failed to load source map: Could not

```
//?substring(startIndex,endIndex)
let str = "welcome to qspiders";
let substring = str.substring(0,
7);
console.log(substring);
console.log(str.slice(0, 7));
console.log(str.substring(6,7))
```

Elements Console
top Filter
Default levels No Issues

```
welcome                    string.js:85
welcome                    string.js:86
welcome                    string.js:87
```

DevTools failed to load source map: Could not load

```
//? substr(startIndex,length)
let str = "welcome to qspiders";
console.log(str.substr(0, 2));
```

```
let str = "test";
console.log(str.slice(0, 3));
console.log(str.slice(-4,-2));
console.log(str.slice(-2, -1));
console.log(str.substring(0, 2));
console.log(str.substr(0, 2));
console.log(str.slice(0, 2));
```

top Filter
Default levels No Issues

```
tes
te
s
te
te
te
```

```
let str = "   demo  ";
console.log(str.length);
console.log(str.trim().length);
console.log(str.trimStart().
length);
console.log(str.trimEnd().length);
```

```
Elements  Console
top ▼  Filter
Default levels ▼  No Issues
9
4
6
7
```

```
let str = "x";
console.log(str.padStart(3, "p"));
console.log(str.padEnd(5, "z"));

let num = 5;
console.log(num.toString().padStart
(4, "0"));
console.log(num.toString().padEnd
(4, "0"));

let no = "8";
console.log(typeof no);
console.log(no.padStart(6,"7"))
```

```
Elements  Conso
top ▼  Filter
Default levels ▼  No Issues
ppx
xzzzz
0005
5000
string
777778
```

```
var a = 'paa';

console.log(a.padStart(5,'x'));
console.log(a.padEnd(5,'m'));

console.log(a.padStart(3,'b'));
//  it won't add because already the stirng size is 3
```

```
STDIN
Input for the program (
Output:
xxpaa
paamm
paa
```

```
let fName = "Shiva";
// console.log(fName.padEnd(8,
"Prakash"))
console.log(fName.concat
("Prakash"));

let lName = "Kumar";
console.log(fName + lName);
console.log(fName.concat(lName,
"Trainer"));
console.log(fName.concat(lName).
concat("Trainer"));
```

```
ShivaPrakash
ShivaKumar
ShivaKumarTrainer
ShivaKumarTrainer
DevTools failed to load s
content for chrome-extens
dbepaooicaho/sourceMap/sc
```

```
let statement =
  "I am from Mysuru city, and
  Mysuru is also a district in
  Karnataka";
// console.log(statement.replace
("Mysuru","Bengaluru"))
// console.log(statement.replaceAll
("Mysuru", "Bengaluru"));

let replaceEx = statement.replace
("Mysuru", "Bengaluru");
console.log(replaceEx);

let replaceAllEx = statement.
replaceAll("Mysuru", "Bengaluru");
console.log(replaceAllEx);
```

```
I am from Bengaluru city, and Mysuru  string.js:135
is also a district in Karnataka
I am from Bengaluru city, and         string.js:138
Bengaluru is also a district in Karnataka
DevTools failed to load source map: Could not load
content for chrome-extension://fheoggkfdfchfphceeif
dbepaooicaho/sourceMap/scripts/iframe_form_check.j
s.map: System error: net::ERR_BLOCKED_BY_CLIENT
>
```

```
//?split():
let str = "demo";
console.log(str.split(""));//
string is converted to array
```

```
▸(4) ['d', 'e', 'm', 'o']
DevTools failed to load source
content for chrome-extension://
```

Undefined

```
eDT > JS undefined.js > ...
let a;
console.log(a);

console.log(test); //ud, because
of hoisting
var test;
console.log(test);

// console.log(demo);//error,
because of TDZ
// let demo;
// console.log(demo);

// console.log(c);
// const c = 100;
// console.log(c)
```

```
Elemer
top ▼
Default levels ▼
undefined
undefined
undefined
DevTools faile
content for ch
dbepaooicaho/s
s.map: System
```

```
console.log(demo);
var demo = "hello";
console.log(demo);

//variable hoisting
var demo1;
console.log(demo1);
demo1 = "hello";
console.log(demo1);
```

```
Elements  Console
top ▼  Filter
Default levels ▼  No Issues
undefined
hello
undefined
hello
```

```
//difference between undefined and
not-defined
let x;
console.log(x);//ud

console.log(z);//not defined
```

Null

```
let x;
console.log(x);

let p = prompt("enter anything");
console.log(p);
console.log(typeof p);//bug in
javascript
```

Elements Console »
top ▼ Filter
Default levels ▼ No Issues

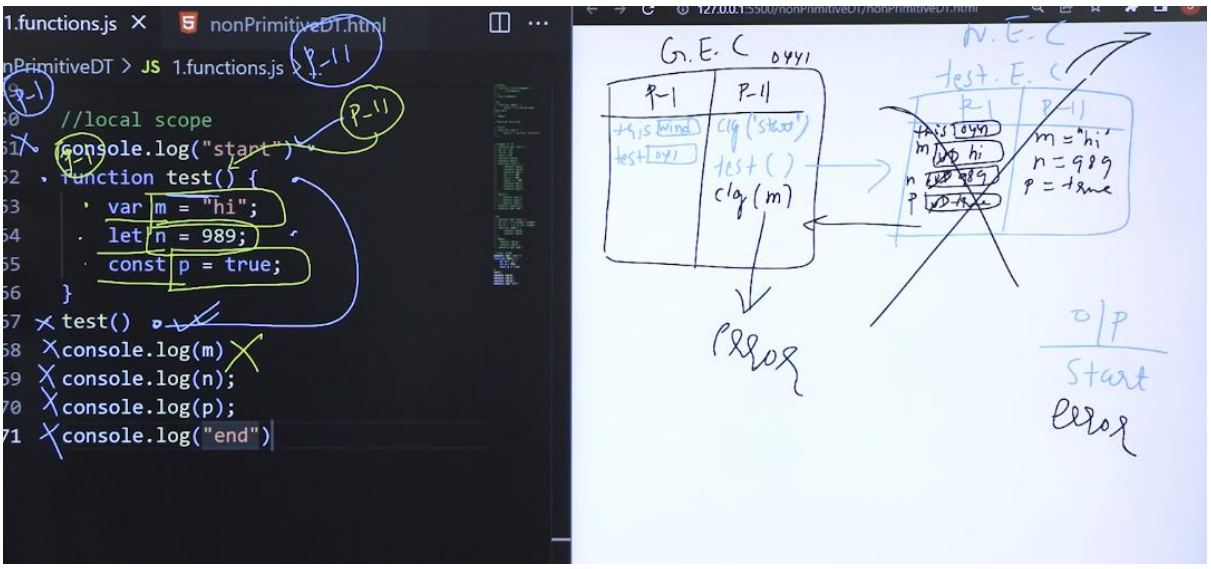
```
undefined
null
object
```

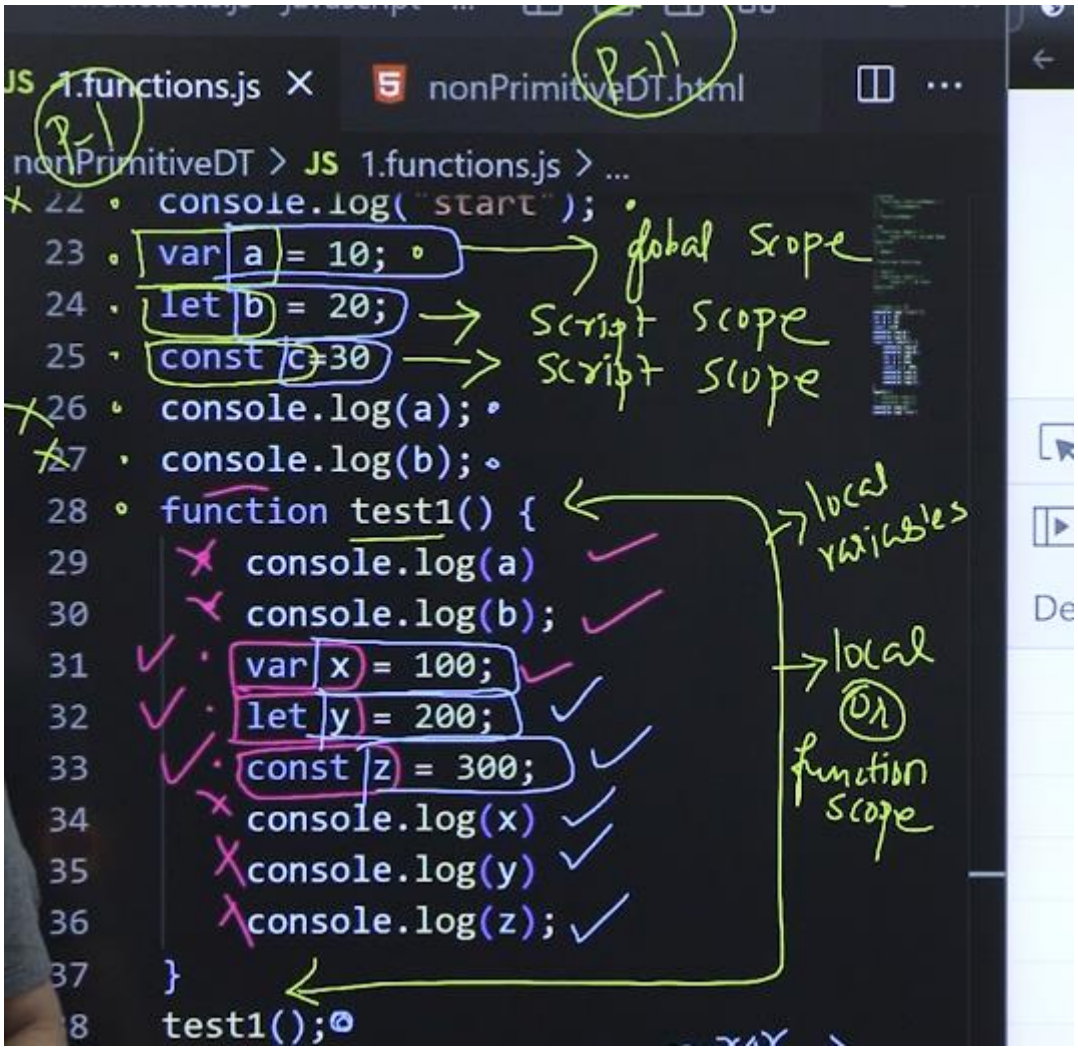
Functions

```
//syntax:
function functionName() {
    //statements
}
functionName()
```

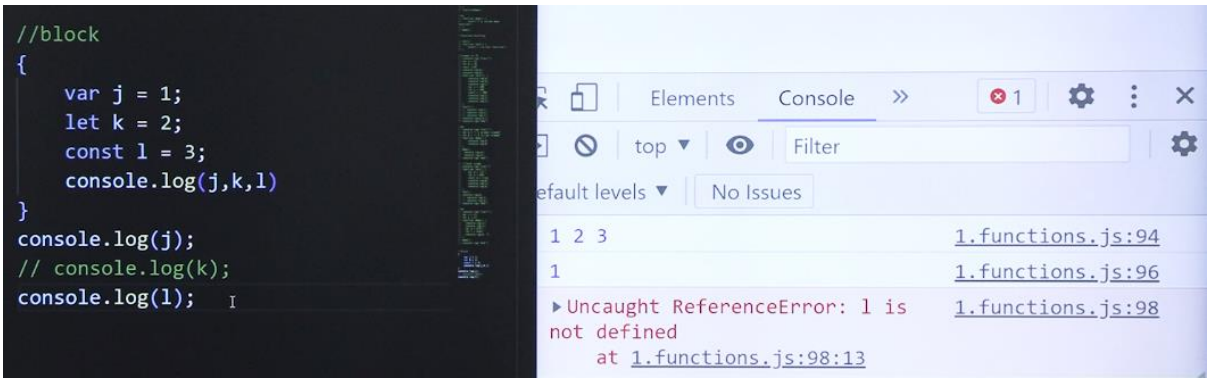
```
//function hoisting

test()
function test() {
    alert(" i m test function")
}
```

Local and Global Scope




1.functions.js
nonPrimitiveDT.html
nonPrimitiveDT > JS 1.functions.js > ...
22 console.log("start");
23 var a = 10;
24 let b = 20;
25 const c=30
26 console.log(a);
27 console.log(b);
28 function test1() {
29 console.log(a)
30 console.log(b);
31 var x = 100;
32 let y = 200;
33 const z = 300;
34 console.log(x)
35 console.log(y)
36 console.log(z);
37 }
38 test1();
global Scope
Script Scope
Script Scope
local variables
local or function scope

Block Scope


//block
{
var j = 1;
let k = 2;
const l = 3;
console.log(j,k,l)
}
console.log(j);
// console.log(k);
console.log(l);
Elements
Console
top
Filter
No Issues
1 2 3
1.functions.js:94
1
1.functions.js:96
Uncaught ReferenceError: l is not defined
at 1.functions.js:98:13
1.functions.js:98

Functions in JS

1. Anonymous
2. Named function
3. function with expression
4. first class function and citizen
5. arrow function
6. higher order function
7. callback function
8. nested function → closures
9. generator function

Anonymous

No name

S/x:

```
function () {
    // instructions
}
```

E/X:

```
function () { alert ("I am anonymous fn")
```

② Named function

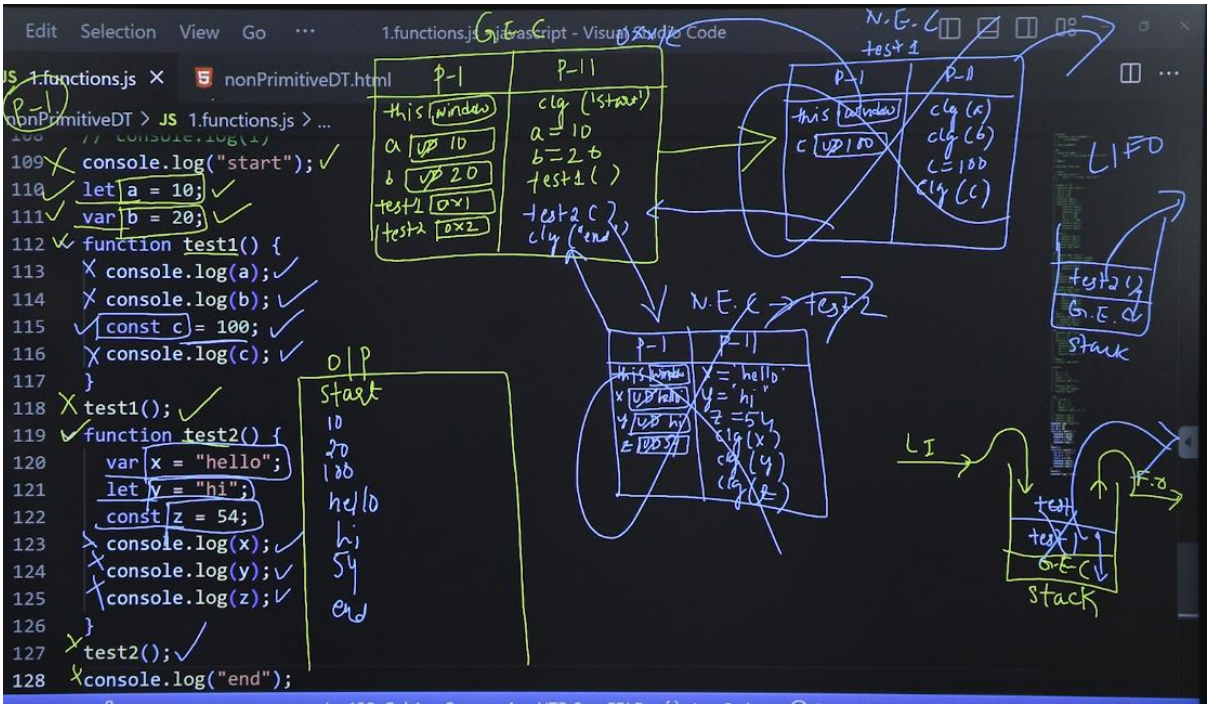
S/x:

```
function fnName () {
    // Statements
}
fnName()
```

E/x:

```
function demo () { alert ("I am alert") }
demo ()
```

How JS Engine executes functions

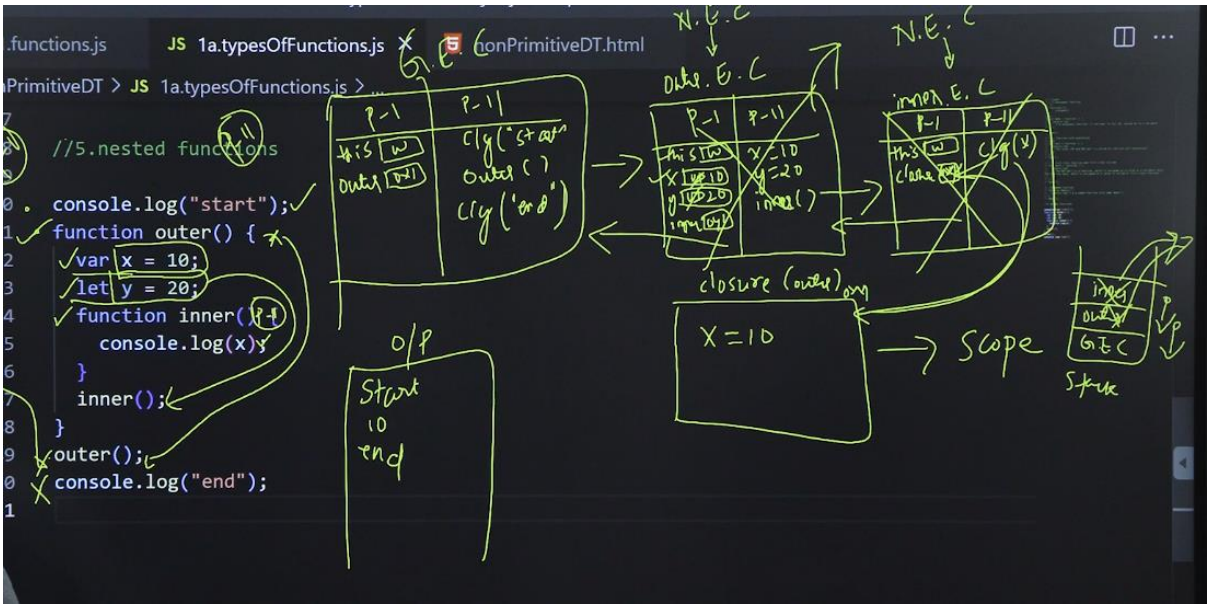


```
//1.anonymous function:

let demo = function () {
  console.log(
    "I m anonymous function, if you want to call me, assign me to a variable"
  );
};
demo();

//2. function with expression
//ex
let test = function () {
  console.log(
    "The total LHS and RHS part is called as function with expression"
  );
};
test();
```

```
//3.first class function and first class citizen
let test2 = function () {
  console.log(
    "The RHS part is a function, which is assigned as a value to a variable test2.
    The function here, which is assigned as a value is called as first class function"
  );
};
test2();

//4.Named function
function demo1() {
  consolo.log("I m a named function with name demo1");
}
demo1();
```

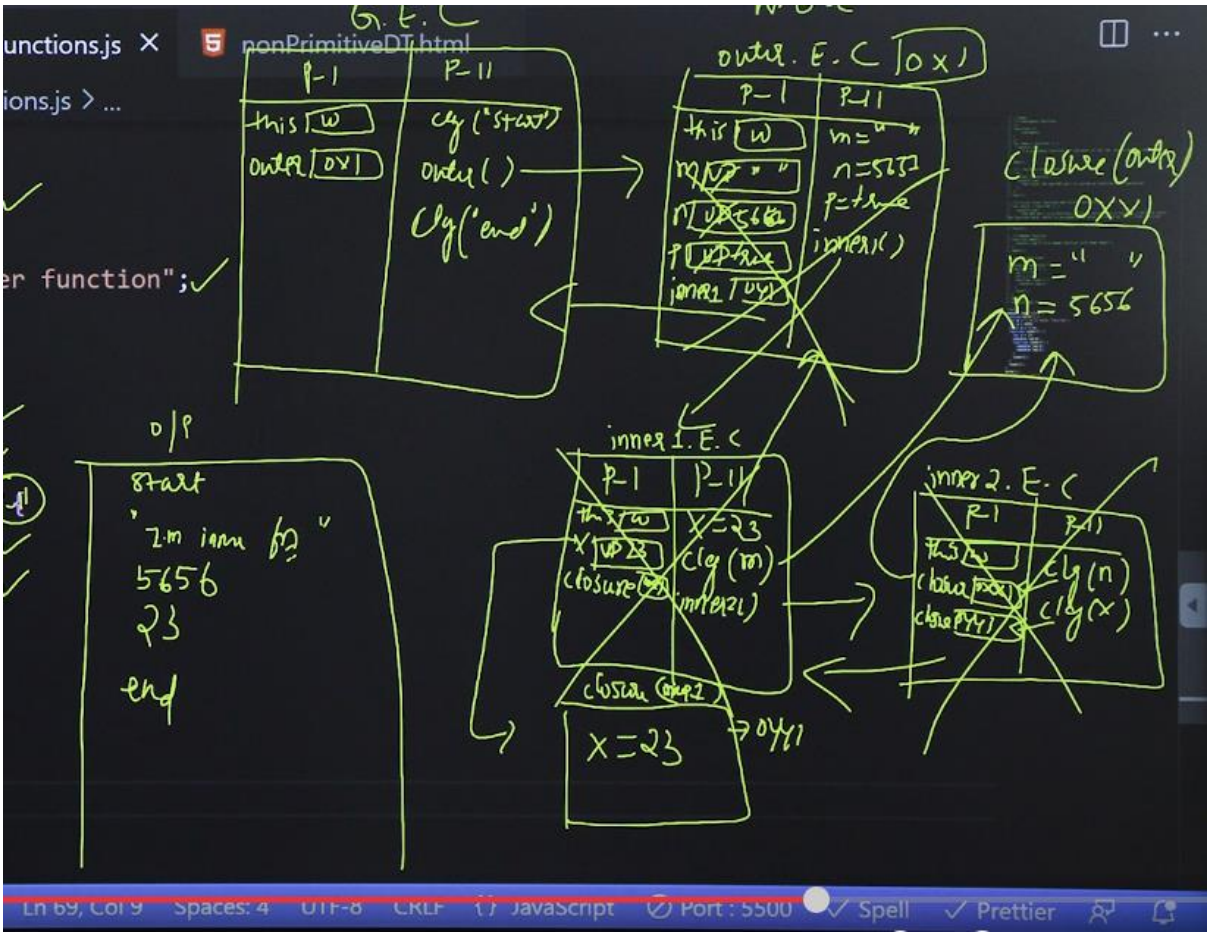
demo1(): void

```
function outer() {
  var x = 10;
  function inner() {
    console.log(x);//closures
  }
  return inner;
}
outer()();
```



```
//ex2
console.log("start");
function outer() {
  var m = "i m in outer function";
  let n = 5656;
  const p = true;
  function inner1() {
    var x = 23;
    console.log(m);
    function inner2() {
      console.log(n);
      console.log(x);
    }
    inner2();
  }
  inner1();
}
outer();
console.log("end");
```

Note : All clousers are nested functions but All nested function are not clouser .



life :

```
//ex
(function () {
  var x = 100;
  console.log(x);
})();
```

Note : we have to call it immediately and only one time we can call it .

HigherOrder Function

index.js 3zxhsbuq8 NEW JAVASCRIPT

```javascript


var add = ( a ,  b ) => console.log(a+b) ;


var hof = ( msg ,  myFunction) =>
{
  console.log(msg);

  myFunction(10,20);
};


hof('hello', add);
```

STDIN

Input for the program (

Output:

```
hello
30
```

index.js 3zxhsbuq8 NEW JAVASC

```javascript


var cbf = ( a ,  b ) => console.log(a+b) ;


var hof = function ( m ,n ,  myFunction)
{


  myFunction(m,n);
};


hof(10 , 90, cbf);
```

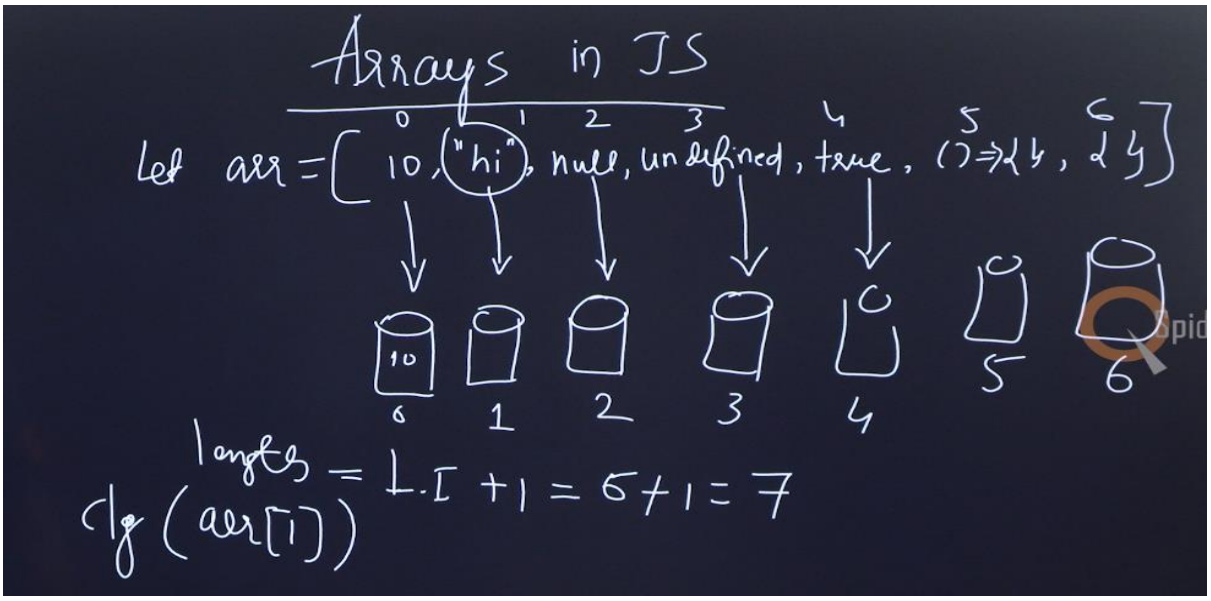
STDIN

Input for the pr

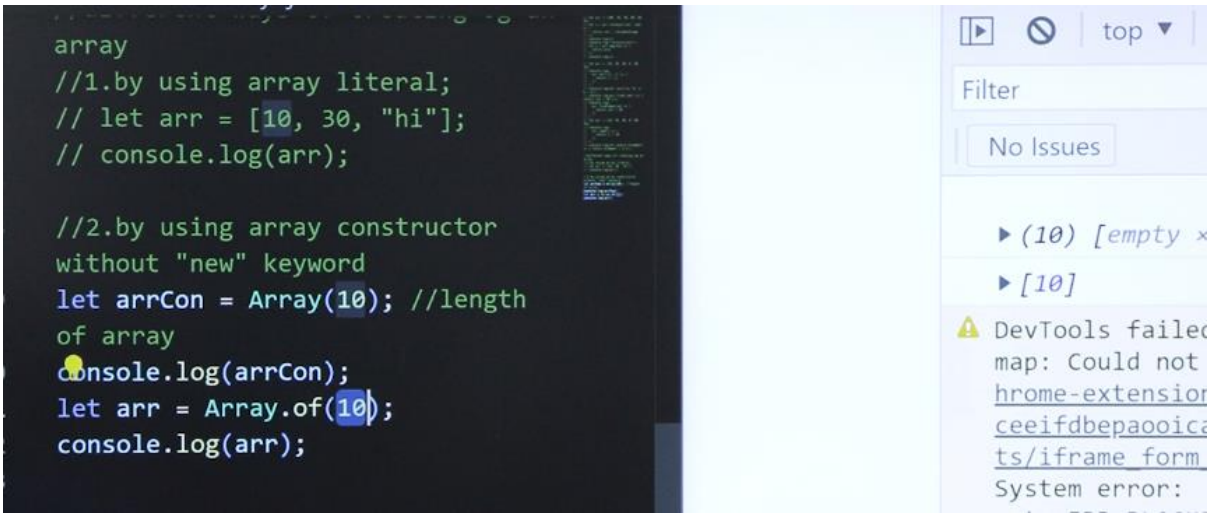
Output:

```
100
```

Arrays



How To Create Array In JS



```
//3.by using array constructor
with "new" keyword;
let arrConObj = new Array(10,20);
console.log(arrConObj);
```

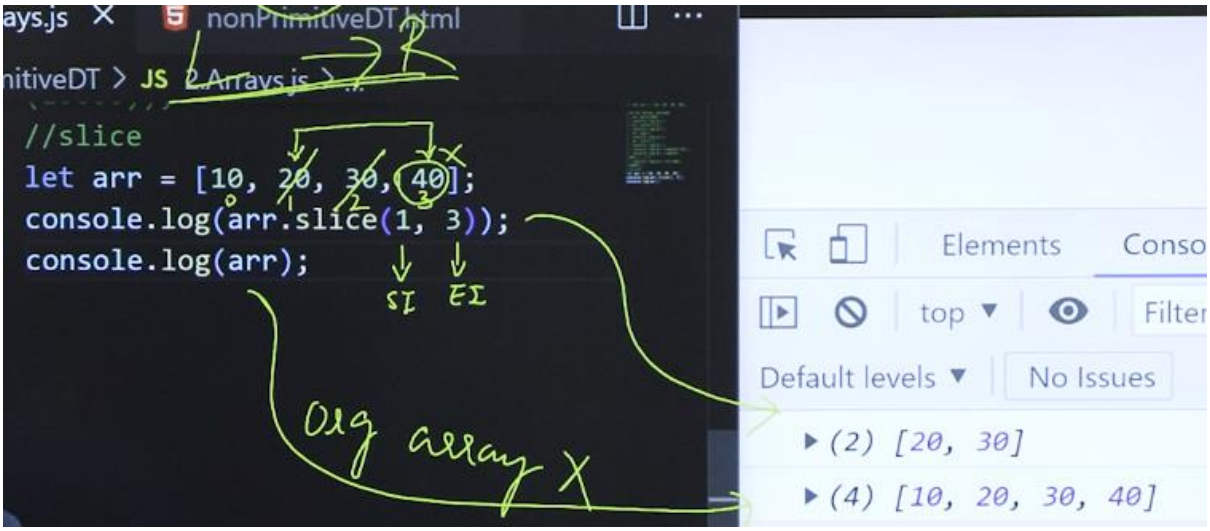
```
Filter
No Issues
▸ (2) [10, 20]
DevTools failed to l
```

Array Methods

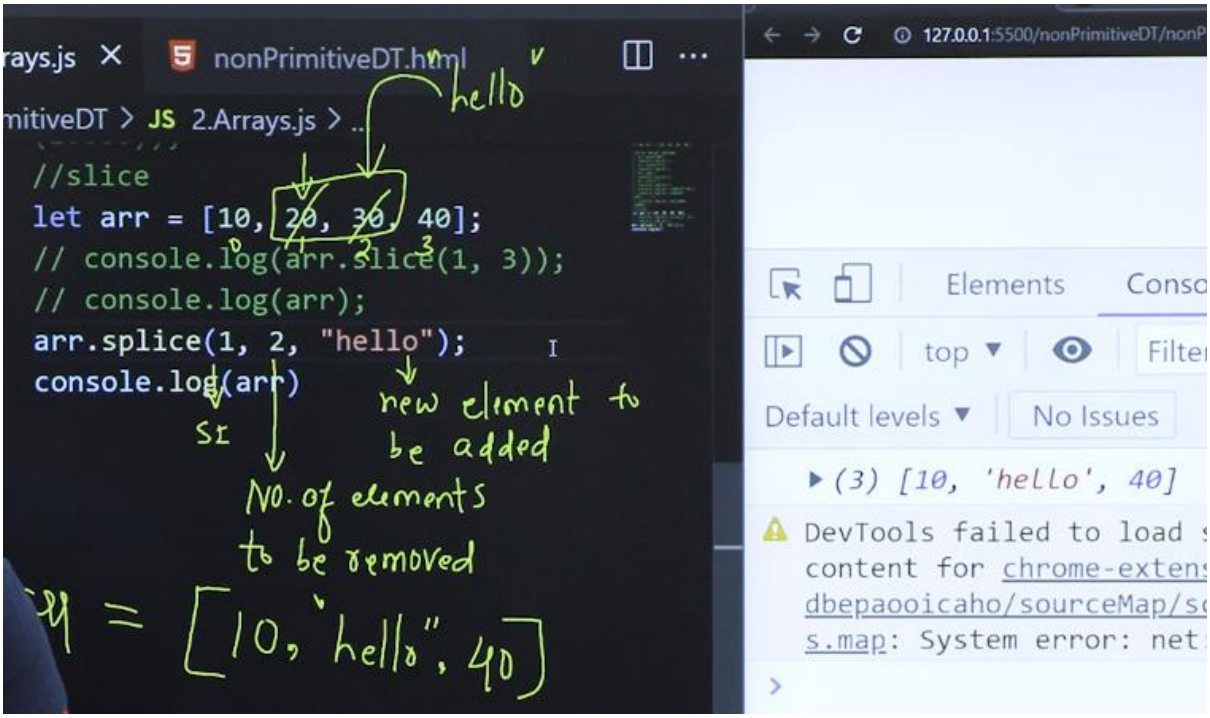
```
mitiveDT > JS 2.Arrays.js > ...
let arr = [10, 20, 30, 40];

//Array helper methods
arr.push(600);
console.log(arr);
arr.unshift(5);
console.log(arr);
arr.pop();
console.log(arr);
arr.shift();
console.log(arr);
console.log(arr.indexOf(20));
console.log(arr.indexOf(100));//-1
console.log(arr.includes(10000));
```

```
Elements  Console
top ▾  Filter
Default levels ▾  No Issues
▸ (6) [5, 10, 20, 30, 40, 600]
▸ (5) [5, 10, 20, 30, 40]
▸ (4) [10, 20, 30, 40]
1
-1
false
```

Slice



Splice



Concat

```
let arr1 = [10, 20];
let arr2 = [30, 40];
let arr3=arr1.concat(arr2,[50,60],
["hello"])
console.log(arr3)
```

```
Elements   Console   »
top ▼   Filter
Default levels ▼   No Issues
2.Array
▶ (7) [10, 20, 30, 40, 50, 60, 'hello']
```

Reverse And Join Methods

```
2.Arrays.js ×   nonPrimitiveDT.html
onPrimitiveDT > JS 2.Arrays.js > ...
50   // let arr = [1, 2, 3];
51   // // console.log(arr.reverse());
52   // console.log(arr.join("*"));
53   // console.log(arr);
54   //ex
55   let str = "hello";
56   let arr = str.split("");
57   console.log(arr.reverse().join
     (""));
```

```
Elements   Console   »
top ▼   Filter
Default levels ▼   No Issues
olleh
⚠ DevTools failed to load source
content for chrome-extension:
```

```
let arr = [50, 6];
// console.log(Array.isArray(arr));

//array to string
console.log(arr.join())
console.log(typeof arr.join());
console.log(arr.toString())
console.log(typeof arr.toString());
```

```
Elements
top
Default levels
No
50,6
string
50,6
string
```

```
let arr = [10, 20, 10, 20, 20, 30,
40];
console.log(arr);
console.log(arr.indexOf(20));
console.log(arr.indexOf(20, 2));
console.log(arr.lastIndexOf(20));
```

Array Higher Order Methods

```
//!Array higher order methods
//filter
let arr = [10, 20, 30, 40];
let filteredElements = arr.filter
(f => {
  return f > 20;
});
console.log(filteredElements);
```

Elements Conso
top ▼ Filter
Default levels ▼ No Issues
▶ (2) [30, 40]
DevTools failed to load s
content for chrome-extens

Without Filter method :

```
let arr =[10,29,30,40,50];

let arr1 = [];

let greater = (e) => e>30 ;


let hof =(myFunc)=>
{
  for(var i =0 ; i<arr.length ;i++)
  {
    if (myFunc(arr[i]))
     arr1.push(arr[i]);
  }
}

hof(greater);

console.log(arr1);
```

STDIN
Input for the pro

Output:
[40, 50]

Map Method

```
//?map
let arr = [20, 40, 60];
let mappedArr = arr.map(ele => {
  return ele + 5;
});
console.log(mappedArr);
```

Elements Consol
top Filter
Default levels No Issues
▶ (3) [25, 45, 65]
DevTools failed to load s

Without Map Method

```
let arr =[10,29,30,40,50];

let arr1 = [];

let add = (e) => e+30 ;


let hof =(myFunc)=>
{
  for(var i =0 ; i<arr.length ;i++)
  {

    arr1.push(myFunc(arr[i]));

  }
}

hof(add);

console.log(arr1);
```

STDIN

Input for the program (Optional)

Output:

```
[ 40, 59, 60, 70, 80 ]
```

Reduce Method

```
//?reduce
let arr = [10, 20, 30];
let redArr = arr.reduce((acc, LI)
=> { return acc + LI },2);
console.log(redArr)
```

initial Val

2 + 10 = 12
acc

12 + 20 = 32
acc

32 + 30 = 62
acc + LI = 2v

Default leve

62

DevTool
content
dbepaoo
s.map:

```
let arr = [20, 30, 40, 50, 100];
console.log(
    arr.filter(f=>f>40)
)
console.log(arr);
console.log(
    arr.map(m=>m+5)
)
console.log(arr);

console.log(
    arr.reduce((acc,l)=>acc+l)
)
console.log(arr)
```

top ▾ Filter

Default levels ▾ No Issues

▸ (2) [50, 100]	2.Arrays.js:161
▸ (5) [20, 30, 40, 50, 100]	2.Arrays.js:164
▸ (5) [25, 35, 45, 55, 105]	2.Arrays.js:165
▸ (5) [20, 30, 40, 50, 100]	2.Arrays.js:168
240	2.Arrays.js:170
▸ (5) [20, 30, 40, 50, 100]	2.Arrays.js:173

DevTools failed to load source map: Could not load content for chrome-extension://fheoggkfdfchfphceeifdbepaooicaho/sourceMap/scripts/iframe_form_check.js.map: System error: net::ERR_BLOCKED_BY_CLIENT

For each

```
let arr = [100, 20, 40, 60, 80,
70];
let x = arr.forEach((val, ind) => {
  return val;
});
console.log(x);
console.log("***********");
let y = arr.map(ele => {
  return ele;
});
console.log(y);
```

```
Elements   Console
top ▼   Filter
Default levels ▼   No Issues
undefined
***********
▶ (6) [100, 20, 40, 60, 80, 70]
⚠ DevTools failed to load source map:
```

Sort

```
let arr = [100, 20, 40, 60, 80,
70];
console.log(
  arr.sort((x, y) => {
    return x - y;
  })
);
```

```
Elements   Console
top ▼   Filter
Default levels ▼   No Issues
▶ (6) [20, 40, 60, 70, 80, 100]   2.Arrays.js
⚠ DevTools failed to load source map: Could not l
content for chrome-extension://fheoggkfdfchfphc
dbepaooicaho/sourceMap/scripts/iframe_form_chec
```

Find and find index

```
let arr = [10, 20, 40, 6, 80, 70];
// console.log(
//   arr.sort((x, y) => {
//     return x - y;
//   })
// );
// console.log(arr.sort((a, b) =>
b - a));
console.log(arr.find((val) => {
return val > 20 }));
console.log(
  arr.findIndex(val => {
    return val > 20;
  })
```

```
Default leve
40
2
DevTool
content
dbepaoo
s.map:
>
```

Some

```
let arr = [10, 20, 40, 6, 80, 70];
console.log(
  arr.some(x => {
    return x > 10;
  })
);
```

```
Elements
top
Default levels
No Issue
true
DevTools failed to lo
content for chrome-ex
```

Note : If any of the value is satisfying the given condition then it will return true .

Every

```
let arr = [10, 20, 40, 6, 80, 70];
// console.log(
//   arr.some(x => {
//     return x > 10;
//   })
// );
console.log(arr.every((element)=>
{return element>40}))
```

```
Default levels
false
DevTools
content
dbepaooi
```

If all the elements are satisfying with the given condition then only it will return true or else it will return false .

Object

```
              creating object

let emp = {
    eName : 'kl',
    id : 1,

}

console.log(emp);

               Adding

emp.sal = 60000;
console.log(emp)

               Updating

emp.id = 2 ;
console.log(emp)

         Adding Object to Object

 emp.address = {
    city : 'bangalore',
    pin : 560989
}

console.log(emp)


emp.address.landmark = "gottigere";
console.log(emp)

     delete element from object

delete emp.id ;
console.log(emp)
```

Object Methods

Object.keys()

```
let keys = Object.keys(emp);
console.log(keys)             //? converts object to array
```

Object.values()

```
let values = Object.values(emp);
console.log(values)         //? converts object to array
```

Object.entries()

```
let key_value = Object.entries(emp);
console.log(key_value)          //? converts object to array, we
                                     will  get array of arrays
```

Object.freeze()

if we freeze any object then we can't perform add, update , delete

```
Object.freeze(emp);
console.log(Object.isFrozen(emp));
emp.isPlayer = true ;
console.log(emp);
```

Object.seal()

similar to freeze() but here we can do update .

```
Object.seal(emp);
console.log(Object.isSealed(emp));
emp.sal = 90000;                  //? it can be done
console.log(emp)
```

Object.fromEntries()

it is opposite of Object.entries() , it converts the array to object

```
let stu={
    name : 'rishav',
    age : 18
}

let itrobj = Object.entries(stu);
console.log(itrobj)

console.log(Object.fromEntries(itrobj))
```

Object.defineProperties()

here we can add more than one key_value pair into the object

```
let stu ={
    sname :"san"
}

console.log(
    Object.defineProperties(stu,{
        id: {value : 12},
        class:{value:'nine'},
        section: {value:'A'}
    })
)

console.log(stu)
```

Object.defineProperty()

here we can add only single property

```
console.log(Object.defineProperty(stu,"age",{value:14}))
```

object.assign()

adding one object's properties into another object it's used

```
let ob1 ={
    id: 101
}

let ob2 ={
    name : 'bumrah'
}

console.log(Object.assign(ob2,ob1));
```

```
▶ {name: 'bumrah', id: 101}                    script.js:131
```

Different way to Create Object

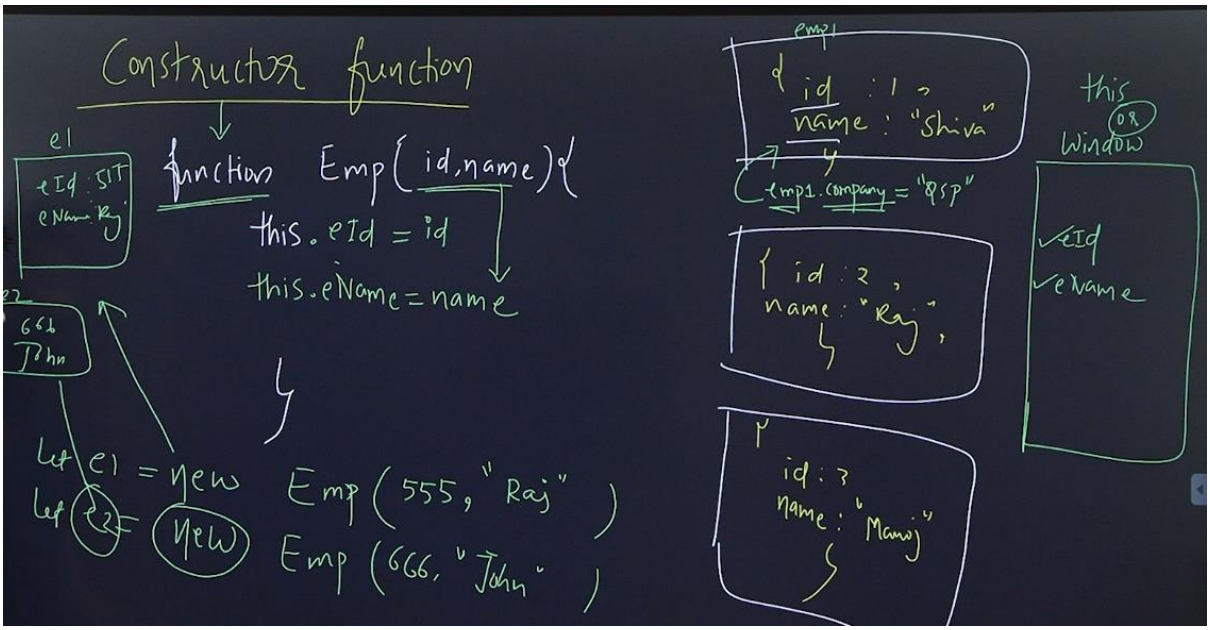
```
//!Different ways of creating an
object
/*
1.by using Object literal
2.by using Object constructor
3.by using constructor function
*/
//?1.by using Object literal
let emp = {
  en: "Shiva",
  id: 123,
};
```

By Using Object Constructor

```
//?by using Object constructor
console.log(Object("hello"))
console.log(Object(123));
console.log(Object(true));
console.log(Object(1n));
```

```
▶ String {'hello'}
▶ Number {123}
▶ Boolean {true}
▶ BigInt {1n}
```

By Using Constructor Function




```
function Emp(id, en) {
  //   console.log(this);
  this.empId = id;
  this.empName = en;
  this.city = "Bengaluru";
}

let e1 = new Emp(555, "Shiva");
console.log(e1);

let e2 = new Emp(666, "Shoaib");
e2.company = "QSP";
console.log(e2);

let e3 = new Emp(777, "Raj");
console.log(e3);
```

```
3.objects.js:178
▶ Emp {empId: 555, empName: 'Shiva', city: 'Bengaluru'}
3.objects.js:182
▶ Emp {empId: 666, empName: 'Shoaib', city: 'Bengaluru', company: 'QSP'}
3.objects.js:185
▶ Emp {empId: 777, empName: 'Raj', city: 'Bengaluru'}
>
```

Math Object

```
//!Math object
console.log(Math)
console.log(Math.max(10, 20, 6, 8,
100))
console.log(Math.min(10, 20, 6, 8,
100));
console.log(Math.floor(10.9))
console.log(Math.floor(10.6));
console.log(Math.floor(10.1));
console.log(Math.ceil(10.1));
console.log(Math.ceil(10.3));
console.log(Math.ceil(10.9));
console.log(Math.ceil(10.5));
console.log(Math.round(10.1))
console.log(Math.round(10.9));
```

```
console.log(Math.sqrt(4))
console.log(Math.sqrt(9));
console.log(Math.cbrt(8));
console.log(Math.cbrt(27));
console.log(Math.pow(2,2))
console.log(Math.pow(2, 3));
```

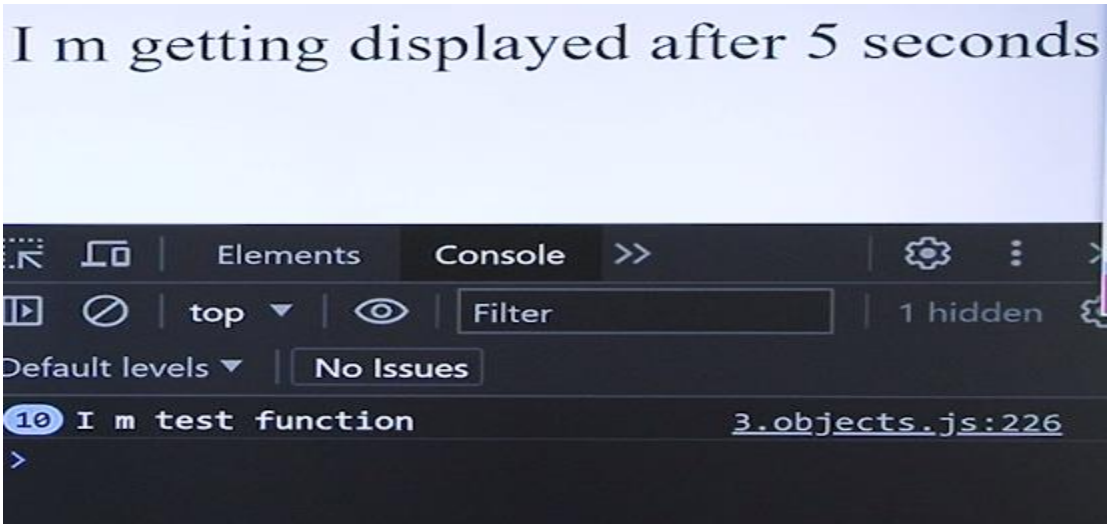
Date Object

```
//?Date object
console.log(new Date()); //date
constructor
let x = new Date();
console.log(x.getDate());
console.log(x.getDay()); //0 means
Sunday, 6-Saturday
console.log(x.getMonth()); //0
means January, 11-December
console.log(x.getFullYear());

let day = ["Sun", "Mon", "Tue",
"Wed", "Thur", "Fri", "Sat"];

console.log(day[1]);
console.log(day[x.getDay()]);
```

```
Mon Jun 26 2023 12:52:18 GMT+0530
(India Standard Time)
26
1
5
2023
Mon
Mon
>
```

Time Object

```
//!Time object
/*
1.setInterval(function,
timeInMilliSeconds)
2.setTimeout(function,
timeInMilliSeconds)
*/
function test() {
  console.log("I m test function");
}
function demo() {
  document.write("I m getting
  displayed after 5 seconds");
}
setInterval(test, 3000);
setTimeout(demo, 5000);
```



For-in And for-Of Loop

```
let str = "hello";
for (let i in str) {
    console.log(i)
}
for (let char of str) {
    console.log(char)
}
```

```
0
1
2
3
4
h
e
l
o
```

```
let emp = {
  eName: "Shiva",
  id: 123,
};
for (let x in emp) {
  console.log(x); //keys will be
  iterated
}
for (let val of emp) {
  console.log(val);
}
```

Note :

- **For-in loop prints the index**
- **For-of loop prints the value**
- **In object for-in loop prints keys**
- **We can’t use for-of loop in objects .**

Destructuring

JS destructuringInJS.js > ...

```js
//syntax: ...identifier

let str = "hello";
console.log(...str); //spread operator

let [a, b, ...c] = str;
console.log(a);
console.log(b);
console.log(c); //rest parameter; (packing)
console.log(...c); //spread operator;
```

```
h e l l o                    destructuringInJS.js:5
h                            destructuringInJS.js:8
e                            destructuringInJS.js:9
▶ (3) ['l', 'l', 'o']        destructuringInJS.js:10
l l o                        destructuringInJS.js:11
```

Number

```js
let num = 12345;

let [x,y,...z]= num ;
console.log(x);     //? for number,bollean,null it won't do
                        bcz  these are  not iterable
```

```
28  let arr = [54, "hi", 654, true,
    {}, null];
29  console.log(...arr);//spread
    operator
30  //!array destructuring
31  let [x, y, ...z] = arr;
32  console.log(x)
33  console.log(y);
34  console.log(z);//rest parameter
35  console.log(...z)//spread operator
```

```
54 'hi' 654 true ▶{} null   destructuringInJS.js:29
54                          destructuringInJS.js:32
hi                          destructuringInJS.js:33
                            destructuringInJS.js:34
▶(4) [654, true, {…}, null]
654 true ▶{} null           destructuringInJS.js:35
>
```

```
let emp = {
  eName: "Shiva",
  id: 456,
  sal: 65465,
  city: "Pune",
};
// console.log(...emp)//can not
use spread operator

//!object destructing
let { eName, id, ...r } = emp;
console.log(eName);
console.log(id);
console.log(r); //rest parameter
console.log(...r); //can not use
  ead operator
```

```
let {x , y , ...z} = emp ;

console.log(x)      //? it will give undefined
```

JSON

→ JavaScript Object Notation

→ Format → to store and transfer the data

→ Provides 2 functions → ① Stringify()
② Parse()

① "Id" : 123

key/name value

JSON is Similar to JS Object

② { "id" : 123 } → JSON

Rules for JSON :

Data is stored in name and value pair

Eg: "id" : 123

Data should be separated by comma

Eg: { "id" : 123,
"empName" : "Raj
}

the data in curly braces holds object

Eg: { "id" : 123 }

the data in square braces holds array

Eg: ["id" : 123]

```
let emp = {
    eName : 'san',
    eId : 123,
    isDeveloper : true
}


// ?  converting object to json

let jsonEx1 = JSON.stringify(emp);

console.log(jsonEx1)
console.log(typeof jsonEx1)
console.log(emp)
```

```
{"eName":"san","eId":123,"isDeveloper":true}  script.js:13
string                                        script.js:14
                                              script.js:15
▶ {eName: 'san', eId: 123, isDeveloper: true}
```

```
//? converting json to object

let ob = JSON.parse(jsonEx1);
console.log(ob);
console.log(typeof ob)
```

```
                                              script.js:20
▶ {eName: 'san', eId: 123, isDeveloper: true}
object                                        script.js:21
```

Shallow copy and deep copy

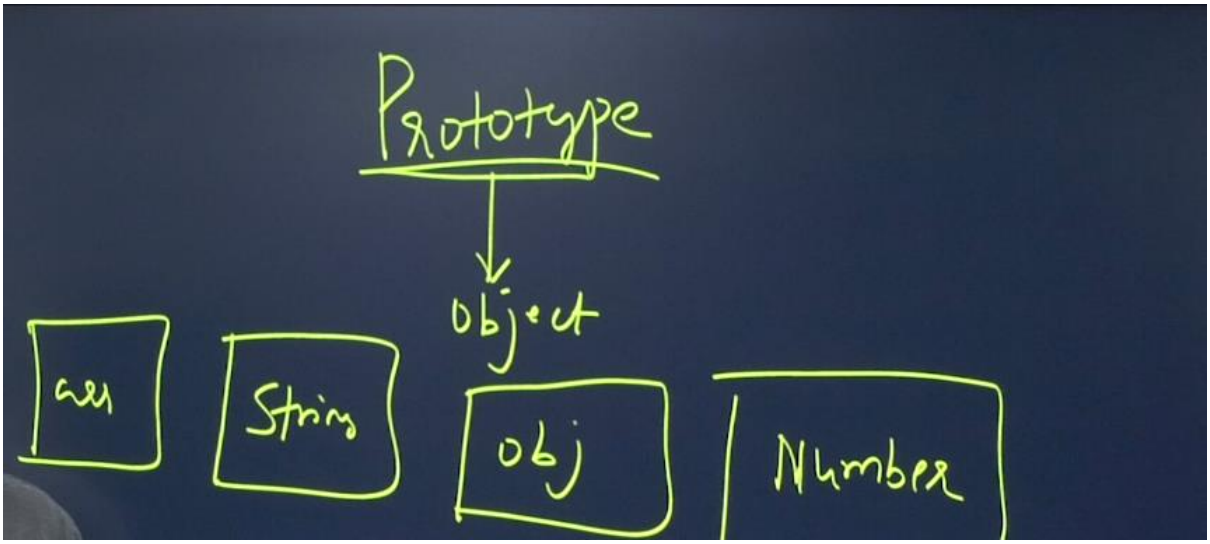
```
let arr = [10 , 20 , 30];

let copy1 = arr ;    //? shallow copy

copy1.push("hi")
console.log(copy1);
console.log(arr);


let jsonEx1 = JSON.stringify(arr);
//it will converts the object into json

console.log(jsonEx1);
console.log(typeof jsonEx1)

copy2 = JSON.parse(jsonEx1)   //? deep copy

console.log(typeof copy2)

copy2.push('hello');
console.log(copy2)
console.log(arr)


//!  deep copy can be done by using rest parameter
without affecting the original array


let [...copy3] = arr;   //? deep copy
copy3.push('welcome')
console.log(copy3)
console.log(arr)
```

```
▶ (4) [10, 20, 30, 'hi']                    script.js:8
▶ (4) [10, 20, 30, 'hi']                    script.js:9
[10,20,30,"hi"]                             script.js:15
string                                      script.js:16
object                                      script.js:20
▶ (5) [10, 20, 30, 'hi', 'hello']           script.js:23
▶ (4) [10, 20, 30, 'hi']                    script.js:24
▶ (5) [10, 20, 30, 'hi', 'welcome']         script.js:32
▶ (4) [10, 20, 30, 'hi']                    script.js:33
```

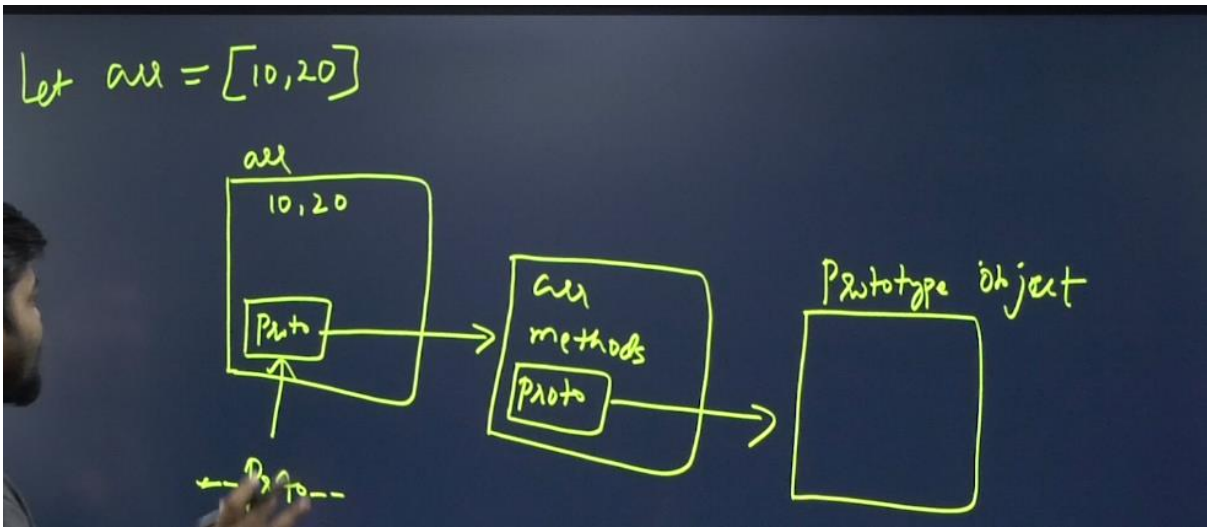
```
//! we can perform shallow and deep copy on object
also

let emp = {
    eName : 'san',
    isDeveloper: true
}

let copy4 = emp ;            //? shallow copy
copy4.address = 'bangalore'
console.log(copy4);
console.log(emp);


let jsonEx2 = JSON.stringify(emp);
console.log(jsonEx2)

let copy5 = JSON.parse(jsonEx2);  //? deep copy
copy5.pin= 560083

console.log(copy5);
console.log(emp);
```

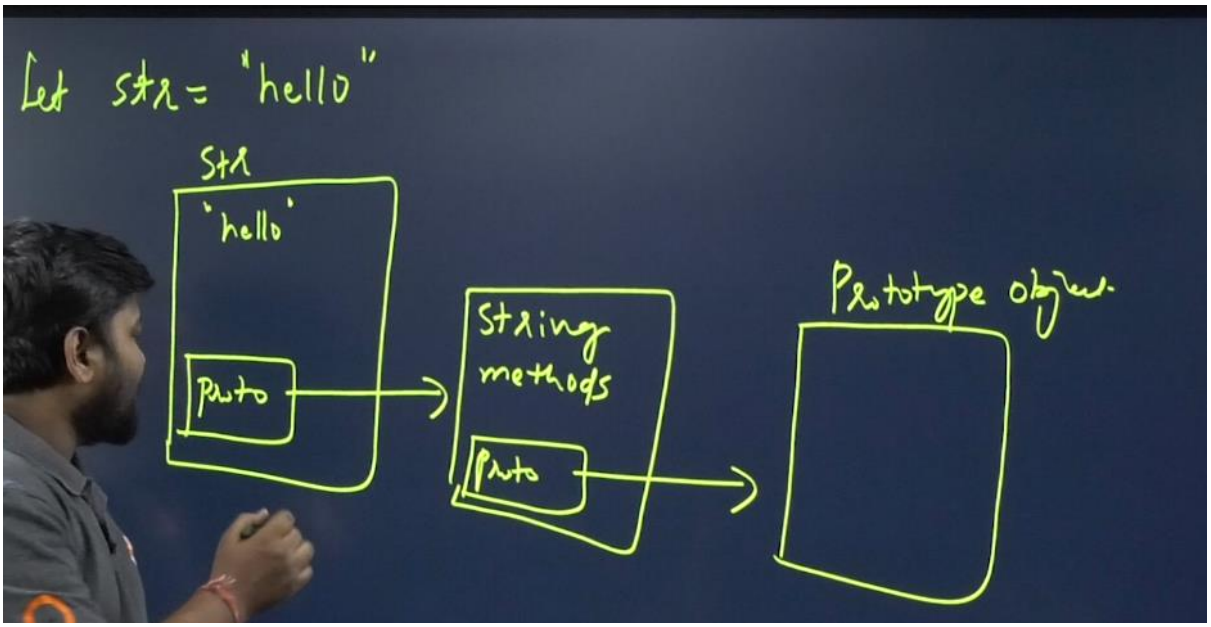
```
                                                    script.js:45
▶ {eName: 'san', isDeveloper: true, address: 'bangalore'}
                                                    script.js:46
▶ {eName: 'san', isDeveloper: true, address: 'bangalore'}
{"eName":"san","isDeveloper":true,"address":        script.js:50
"bangalore"}
                                                    script.js:55
▶ {eName: 'san', isDeveloper: true, address: 'bangalore',
  pin: 560083}
                                                    script.js:56
▶ {eName: 'san', isDeveloper: true, address: 'bangalore'}
```

Prototype



Prototype is an object that is attached to the all data types .






```
let arr = [10, 20];
console.log(arr);
console.log(arr.__proto__);
console.log(arr.__proto__.
__proto__);
console.log(arr.__proto__.
__proto__.__proto__);
console.log(Object.prototype)
console.log(Object.prototype ==
arr.__proto__.__proto__)

let str = "hello";
console.log(str.__proto__)
console.log(str.__proto__.
__proto__);
console.log(str.__proto__.
__proto__== Object.prototype);
```

```
let str = "hello";
console.log(str.__proto__)
console.log(str.__proto__.
__proto__);
console.log(str.__proto__.
__proto__ == Object.prototype);

let num = 123;
console.log(num.__proto__)
console.log(num.__proto__.
__proto__);
console.log(num.__proto__.
__proto__ == Object.prototype);
```

```
let num = 123;
console.log(num.__proto__)
console.log(num.__proto__.
__proto__);
console.log(num.__proto__.
__proto__ == Object.prototype);

console.log(Function.prototype.
__proto__)
console.log(Function.prototype.
__proto__ == Object.prototype);
```

Prototype Function

```
let emp = {
    eName : 'san'
}

console.log(emp.eName);



//  taking a function

function test1()
{
    console.log(this)
}
// ! it is pointing to window object

test1();

                    call() Method

  I want to point to current object , for that call() method
is used

function test1()
{
    console.log(this)
    console.log(this.eName)
}

test1.call(emp);
```

```
Default levels ▾ | No Issues

san                                          script.js:6
                                             script.js:26
▸ Window {window: Window, self: Window, document:
  document, name: '', location: Location, …}
undefined                                    script.js:27
▸ {eName: 'san'}                             script.js:26
san                                          script.js:27
⊗ ▸Uncaught Error: Minified React            content.js:2
error #299; visit https://reactjs.org/docs/error-de
coder.html?invariant=299 for the full message or
use the non-minified dev environment for full
errors and additional helpful warnings.
    at t.createRoot (content.js:2:192056)
    at content.js:2:369403
    at content.js:2:369475
    at content.js:2:369478
>
```

```
// ? ex 2 :

let name = {
    fName : 'san'
}

function test2(lName)
{
    return this.fName + lName;
}

console.log(test2.call(name," adhikary"))
```

apply() method

```
let name = {
    fName : 'san'
}

function test2(lName)
{
    return this.fName + lName;
}

console.log(test2.apply(name,[" adhikary"]))

//? similar to call() method , but here we have to
pass the value as array literal(within the square
brackets)
```

bind() method

```
let Values = {
    a : 10 ,
    b : 20
}

function add(c, d)
{
    return this.a + this.b + c + d
}

console.log(add.call(Values,30,40))

let result = add.bind(Values,10,20)

console.log(result)
```

```
// when we use bind() method it will return a bound function

console.log(result())

// if we call it like a function then it will give the proper op
```

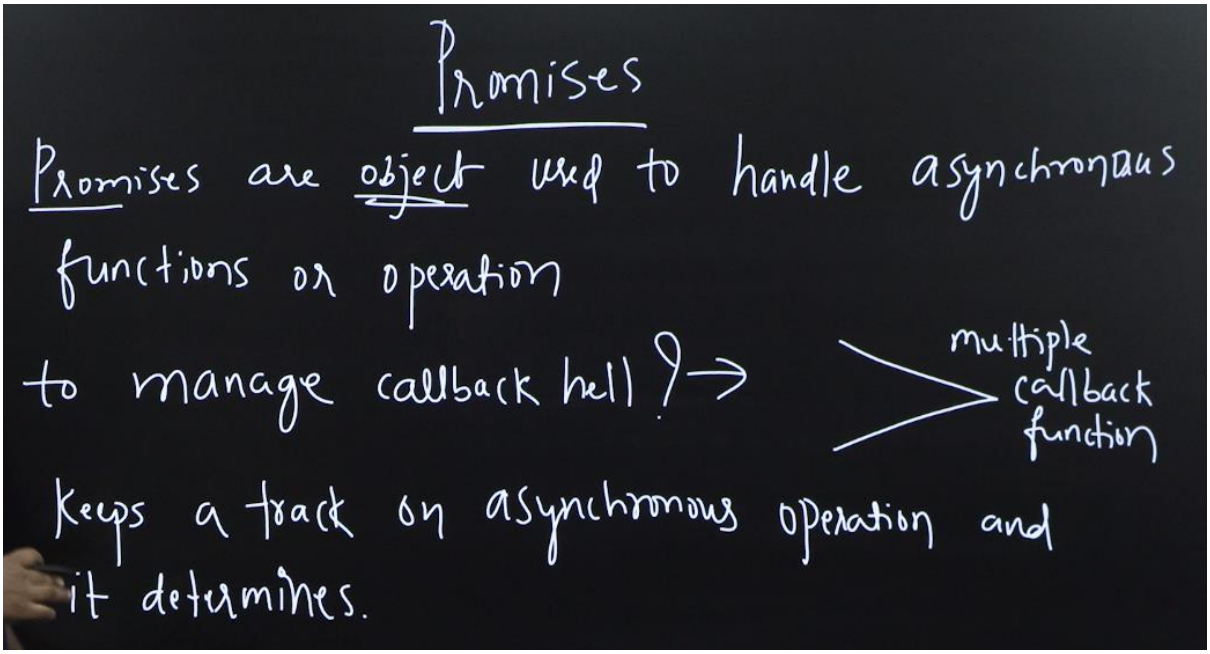
```
100                                   script.js:78
ƒ add(c, d)                           script.js:82
{
    return this.a + this.b + c + d
} ℹ
60                                    script.js:86
▶Uncaught Error: Minified React       content.js:2
```

```
 Note :  if we use arrow function , then it won't point to current object , it will indicate window object


let name = {
    fn : 'santanu'
}


let demo = ()=> console.log(this)

demo.call(name)
```

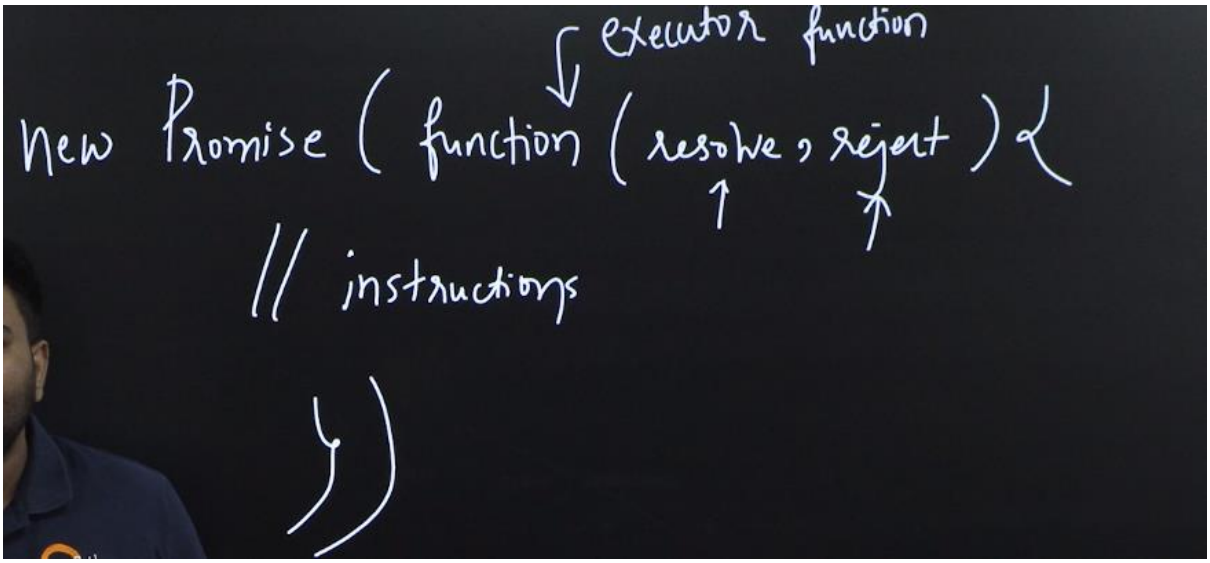
Synchronicity In JS

```
//!synchronous
// console.log(1);
// console.log(2);
// alert("hi");
// console.log(3);
// console.log(4);

//!Asynchronous
console.log(1);
console.log(2);
setTimeout(() => {
  alert("hello");
});
console.log(3);
console.log(4);
```

Promises



Syntax :

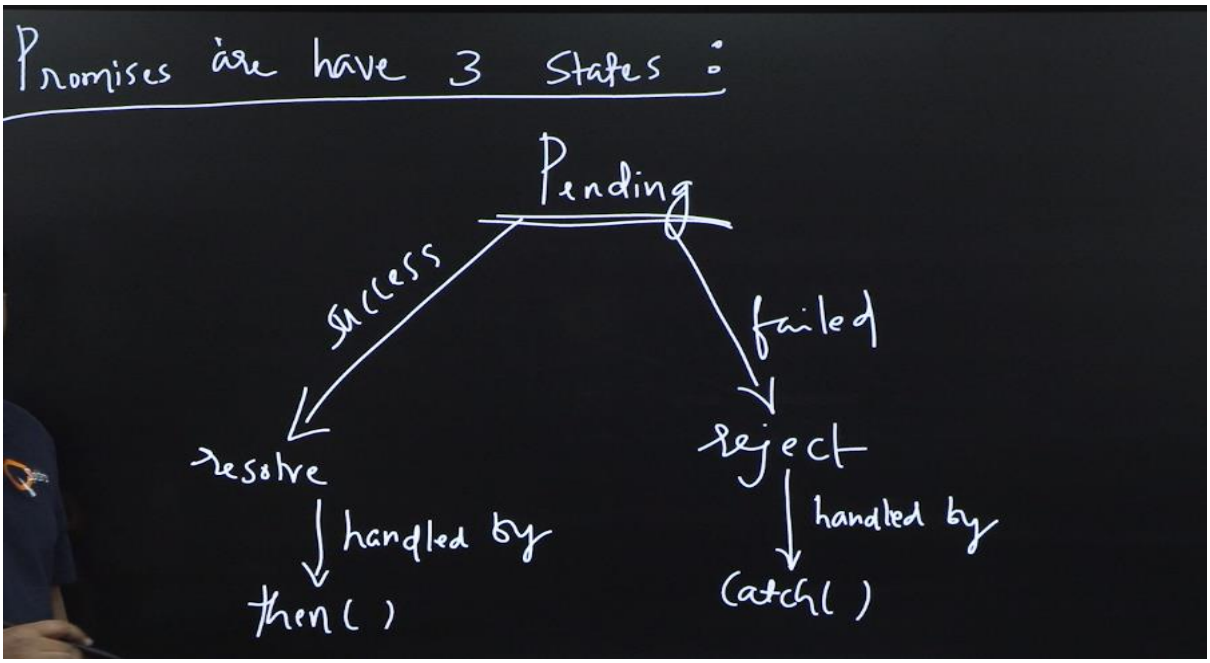


Ex:

```
new Promise ( (resolve, reject) => {
    resolve ('I'm resolved)
    reject ('I'm rejected)
  })
```

Promise instance methods:

1. then ( ) → resolved state
2. catch ( ) → rejected state
3. finally ( ) → ?



> JS App.js > ...

```js
new Promise((resolve, reject) => {
  resolve("i m resolved");
  reject("i m rejected");
})
  .then(result => {
    console.log(result);
  })
  .catch(error => {
    console.log(error);
  })
  .finally(
    console.log(
      "i will be executed
      irrespective of resolved and
      rejected states"
    )
  );
```

```
let p = new Promise(function (res,
rej) {
  let isRoomCleaned = false;
  if (isRoomCleaned) {
    res("yes. The room is
    cleaned");
  } else {
    rej("Idiot, clean the room");
  }
});
p.then(data => {
  console.log(data);
).catch(err => {
  console.log(err);
});
```

```
top ▼
Default levels ▼   No Issues
Idiot, clean the room
>
```

Promise Static Methods

```
//?promise static methods:
/*
1.Promise.any()
2.Promise.all()
3.Promise.allSettled()
4.Promise.race()
*/
```

Promise.any()

- **It will show the first resolved promise .**

```
let p1 = new Promise( (resolve, reject) =>{

  // resolve("p1 resolved");
   reject("p1 rejected")
}
  );

  let p2 = new Promise( (resolve, reject) =>{

   resolve("p2 resolved");
   reject("p2 rejected")
}
  );

  let p3 = new Promise( (resolve, reject) =>{

   resolve("p3 resolved");
   reject("p3 rejected")
}
  );


  Promise.any([p1,p2,p3]).then(
                    data=> console.log(data)
                    ).catch(
                      err=> console.log(err));
```

Output:

```
p2 resolved
```

Promise.all()

- **If all the promises are resolved then it will show all the resolved promises .**
- **if any of the promises is rejected then it will show the rejected promise .**

```
let p1 = new Promise( (resolve, reject) =>{

  resolve("p1 resolved");
   reject("p1 rejected")
}
  );

  let p2 = new Promise( (resolve, reject) =>{

   resolve("p2 resolved");
   reject("p2 rejected")
}
  );

  let p3 = new Promise( (resolve, reject) =>{

   resolve("p3 resolved");
   reject("p3 rejected")
}
  );


  Promise.all([p1,p2,p3]).then(
                        data=> console.log(data)
                        ).catch(
                          err=> console.log(err));
```

Output:

```
[ 'p1 resolved', 'p2 resolved', 'p3 resolved' ]
```

```javascript
let p1 = new Promise( (resolve, reject) =>{

  resolve("p1 resolved");
   reject("p1 rejected")
}
  );

  let p2 = new Promise( (resolve, reject) =>{

   resolve("p2 resolved");
   reject("p2 rejected")
}
  );

  let p3 = new Promise( (resolve, reject) =>{

  // resolve("p3 resolved");
   reject("p3 rejected")
}
  );


  Promise.all([p1,p2,p3]).then(
                        data=> console.log(data)
                        ).catch(
                          err=> console.log(err));
```

Output:

```
p3 rejected
```

Promise.allSettled()

- **It will give array of objects .**
- **If any promise is rejected then it will show the status as rejected and give the reason of reject .**

```
Promise.allSettled([p1,p2,p3]).then(
                    data=> console.log(data)
                    ).catch(
                      err=> console.log(err));
```

```
Output:

[
  { status: 'fulfilled', value: 'p1 resolved' },
  { status: 'fulfilled', value: 'p2 resolved' },
  { status: 'rejected', reason: 'p3 rejected' }
]
```

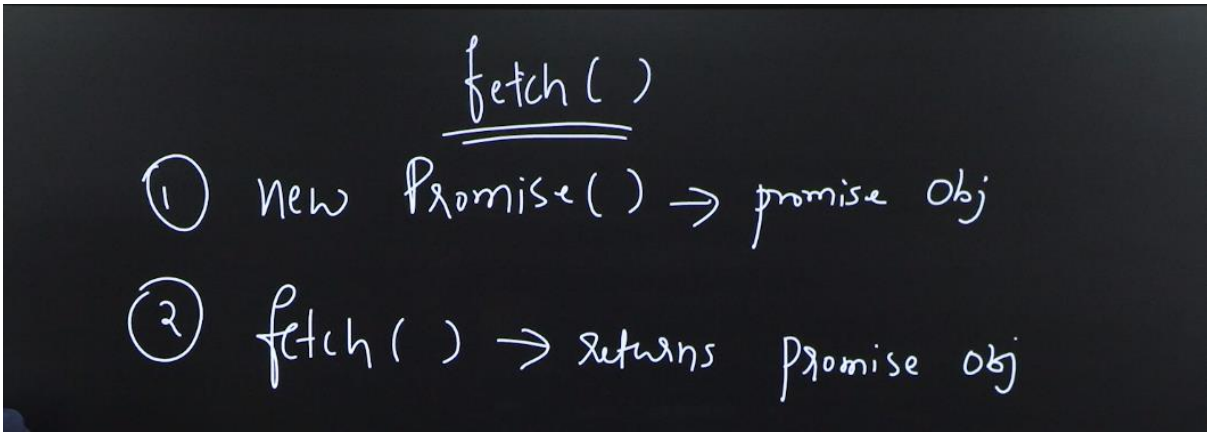
Promise.race()

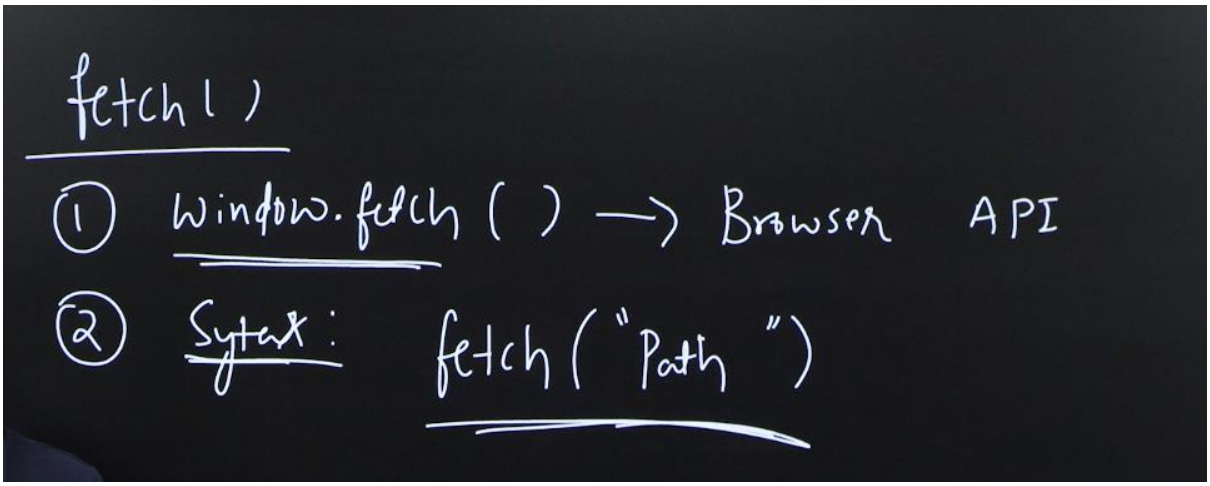
- **It will show the promise which has taken less time . it won't care about whether it is resolved or rejected .**

```
let p1 = new Promise( (resolve, reject) =>{

   setTimeout(function() {
     resolve("p1 resolved");
     reject("p1 rejected")
   }, 3000);
});

  let p2 = new Promise( (resolve, reject) =>{

   setTimeout(function() {
       resolve("p2 resolved");
       reject("p2 rejected")
   }, 1000);
} );

  let p3 = new Promise( (resolve, reject) =>{

   setTimeout(function() {
     resolve("p3 resolved");
     reject("p3 rejected")
   }, 2000);
} );

    Promise.race([p1,p2,p3]).then(
                        data=> console.log(data)
                        ).catch(
                          err=> console.log(err));
```

```
Output:

p2 resolved
```

Fetch Method





```
let p = window.fetch("https://api.
github.com/users");
console.log(p);
p.then(data => {
    console.log(data);
    let jsonData = data.json();
    console.log(jsonData);
    jsonData.then(finalData => {
        console.log(finalData);
        finalData.map(user => {
            console.log(user);
            console.log(user.login)
            console.log(user.id);
        })
    })
}).catch(err => {
  console.log(err);
);
```

Here promise chaining is created.

Async and Await

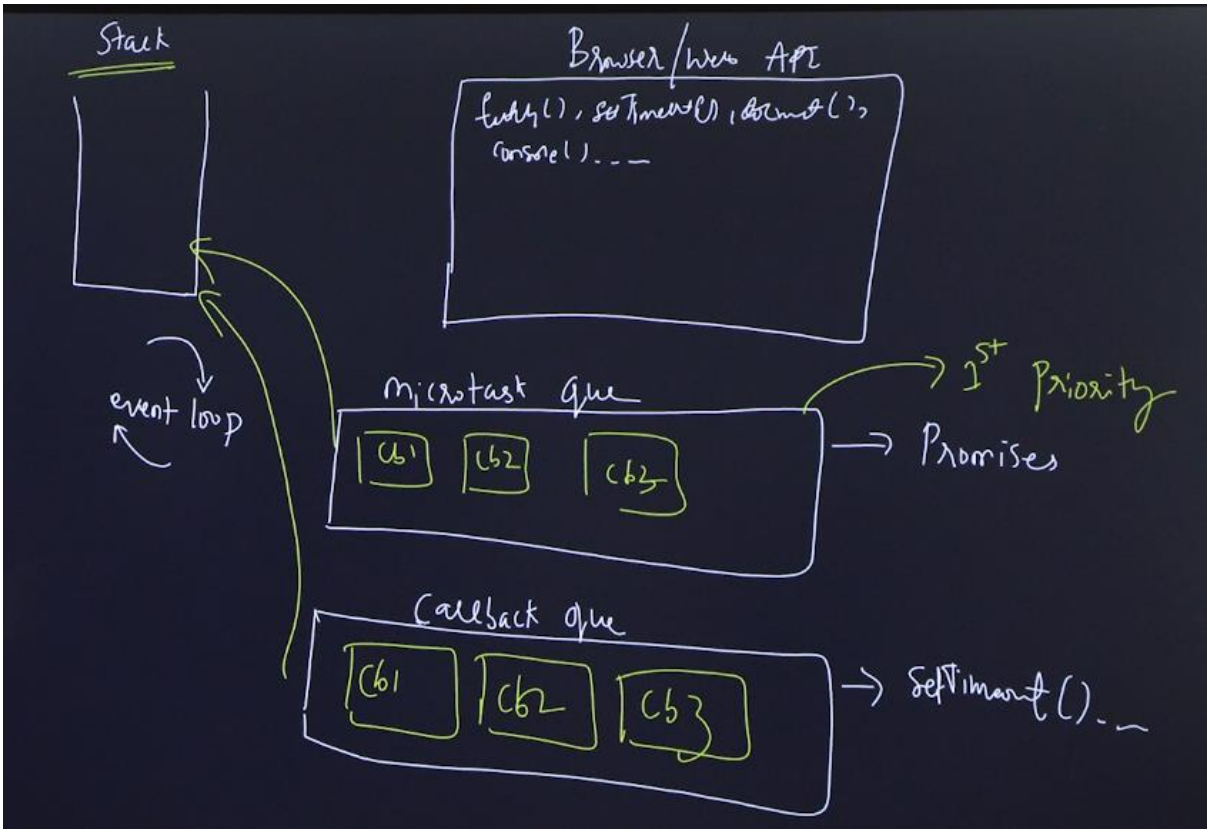
```
async function gitUsers() {
  let data = await fetch("https://
  api.github.com/users");
  console.log(data);
  let finalData = await data.json
  ();
    console.log(finalData);
    finalData.map(user => {
        console.log(user)
    })

}
itUsers();
```

For avoiding promise chaining we use async and await .

Error Handling in Async Await

```
async function git() {
  try {
    let data = await fetch
    ("https://api.hub.com/users");

    let finalData = await data.json
    ();
    console.log(finalData);
  } catch (error) {
    console.log(error);
  }
}
git();
```

Call Stack


Stack
Browser / Web API
fetch(), setTimeout(), setInterval(),
console()...
event loop
microtask que
cb1
cb2
cb3
1st Priority
Promises
Callback que
cb1
cb2
cb3
setTimeout()...

allstack > JS App.js > ...

```
console.log(1);
console.log(2);
setTimeout(cb1 => {
  console.log(3);
}, 3000);
fetch("").then(pcb1 => {
  console.log(4);
});
setTimeout(cb2 => {
  console.log(5);
}, 5000);
fetch("").then(pcb2 => {
  console.log(6);
});
new Promise((res, rej) => {
  res("resolved");
  rej("rejected");

  then(pcb3 => {
   console.log(pcb3);
```

Note : Before fetch method Promise() constructor will execute .

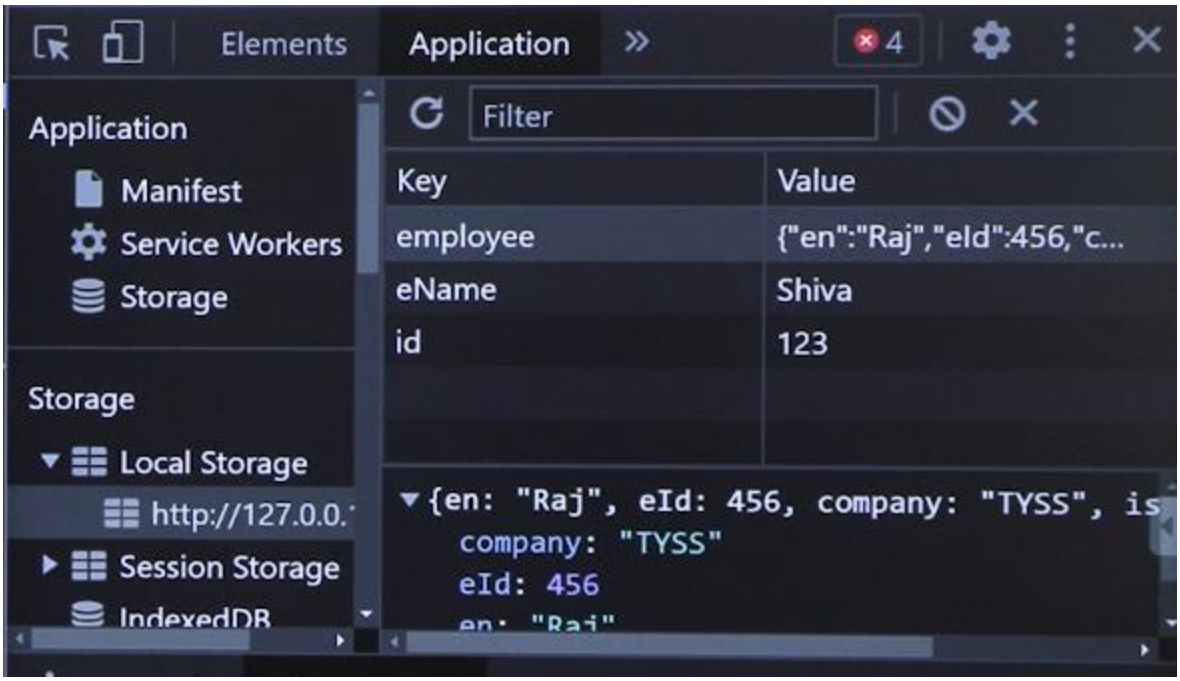
```
1                                                   App.js:1
2                                                   App.js:2
resolved                                            App.js:20
4                                                   App.js:7
6                                                   App.js:13
⚠ DevTools failed to load source map: Could not load content
for chrome-extension://fheoggkfdfchfphceeifdbepaooicaho/sou
rceMap/chrome/scripts/iframe_form_check.js.map: System
error: net::ERR_BLOCKED_BY_CLIENT
3                                                   App.js:4
5                                                   App.js:10
>
```

Local Storage

Storing data :

- **For storing the data in local storage we have to use setItem() method .**
- **It will take 2 arguments .**

```
localStorage.setItem("id", 123);

let emp = {
  en: "Raj",
  eId: 456,
  company: "TYSS",
  isDeveloper: true,
};

let strObj = JSON.stringify(emp);
console.log(strObj)
localStorage.setItem("employee",
strObj);
console.log(typeof strObj)
```



Getting data from localStorage :

- **For fetching the data form local storage we have to use getItem() method.**
- **It will take one argument .**

```
let id = localStorage.getItem
("id");
console.log(id);
```

```
let empObj = localStorage.getItem
("employee");
console.log(empObj);
console.log(empObj.company);
console.log(typeof empObj);
let emp1 = JSON.parse(empObj);
onsole.log(emp1);
nsole.log(typeof emp1);
sole.log(emp1.company)
```

```
{"en":"Raj","eId":456,"company":"TYSS","isDevel App.js:21
oper":true}
undefined                                      App.js:22
string                                         App.js:23
                                               App.js:25
▸ {en: 'Raj', eId: 456, company: 'TYSS', isDeveloper: tru
  e}
object                                         App.js:26
TYSS                                           App.js:27
Console   What's New ×   Issues
```

We have to use parse() method because we have to convert the string into object .

Removing data from the local storage

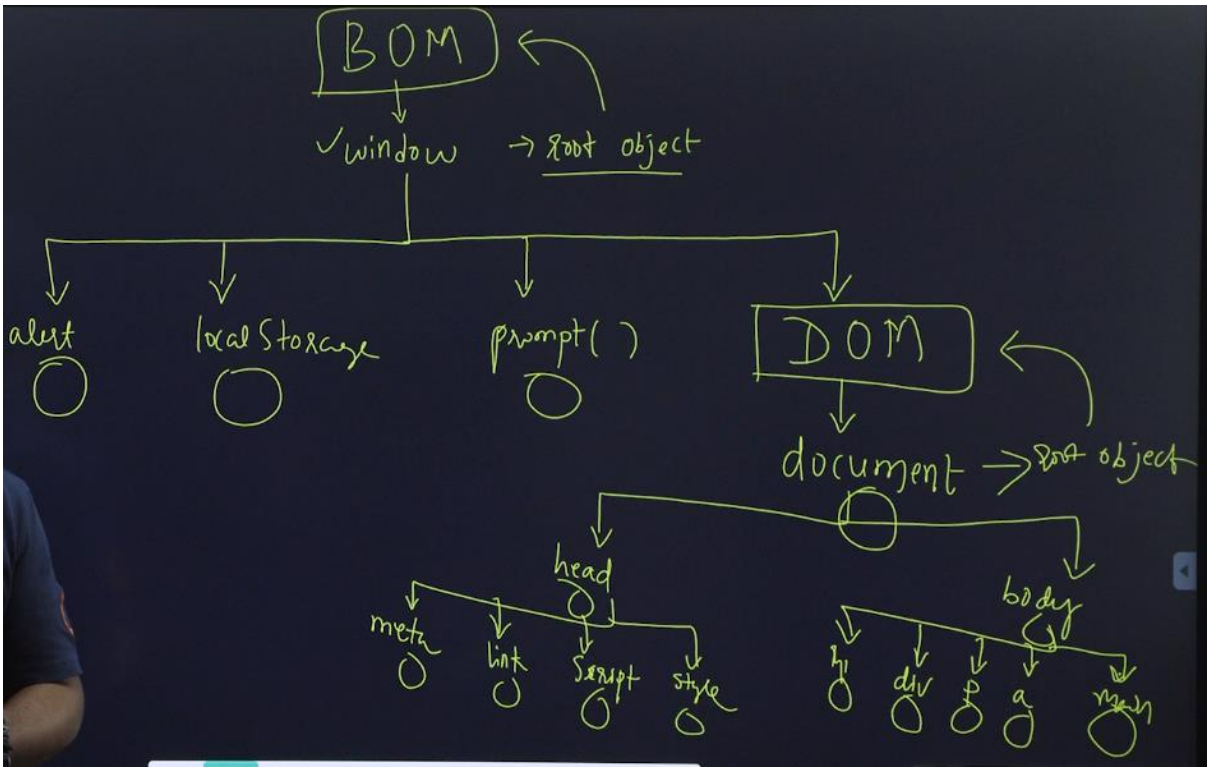
```
//!removeItem:to remove particular
key from the localStorage
localStorage.removeItem("eName");

/!clear: to remove/clear all the
s from the localStorage
lStorage.clear();
```

In the same way we can add, delete , fetch the data form sessionStorage .

But once the window is closed all the data will be removed from the sessionStorage . but the local storage data will be there .

BOM

BOM is an object provided by the browser which having the main global object window that is the root object .



Why do we need DOM ?

- **If we want to interact with the UI or manipulate the UI then we need DOM.**

DOM Methods

1. getElementById():

```
//?DOM method
//1. document.getElementById()
let h1 = document.getElementById
("demo");
console.log(h1);
```

```
top ▼   -url:http://127.0.0.1:5500/DOM   4 hidden
efault levels ▼   No Issues
<h1 id="demo">Hello JS</h1>                App.js:7
```

2. getElementsByClassName :

```
//2.document.getElementsByClassName
let elements=document.
getElementsByClassName("demo");

console.log(elements);
console.log(Array.isArray
(elements))
```

```
                                                App.js:12
▶HTMLCollection(4) [h1.demo, h2.demo, div.demo, p.demo]
false                                           App.js:13
⚠ DevTools failed to load source map: Could not load
```

Note :

- **HTML Collection is not array so we can't perform map() here .**
- **For that first we have to convert it into array by using Array.from() method.**

```
<h1 class="demo">Hello JS</h1>
<h2 class="demo">hi</h2>
<div class="demo">div</div>
<p class="demo">para</p>
```

```
8
9   //2.document.getElementsByClassName
10  let elements=document.
    getElementsByClassName("demo");
11
12  console.log(elements);
13  console.log(Array.isArray
    (elements))
14  let arrOfEle = Array.from
    (elements);
15  console.log(arrOfEle)
16  console.log(Array.isArray
    (arrOfEle))
17  arrOfEle.map(element => {
        console.log(element)
    })
```

```
<body>
  <h1 class="demo">Hello JS</h1>
  <h2 class="demo">hi</h2>
  <a href="">link2</a>
  <div class="demo">div</div>
  <p class="demo">para</p>

  <a href="">link</a>
  <a href="">link</a>
  <a href="">link</a>
  <a href="">link</a>
  <a href="">link</a>

  <script src="./App.js"></
  script>
</body>
```

3. getElementsbyTagName

```
21
22  //!3.document.getElementsByTagName
    (): returns HTMLCollections
23  let a = document.
    getElementsByTagName("a");
    console.log(a)
```

```
Default levels ▼   No Issues
▸HTMLCollection(6) [a, a, a, a, a, a]        App.js:24
⚠ DevTools failed to load source map: Could not load
content for chrome-extension://fheoggkfdfchfphceeifdbepao
```

4. getElementsByName()

```
<h1 name="demo">Hello JS</h1>
<h2 name="demo">hi</h2>
<a href="">link2</a>
<div name="demo">div</div>
<p name="demo">para</p>
```

```
//!4.document.getElementsByName()
:returns NodeList()
let tags = document.
getElementsByName("demo");
console.log(tags);
console.log(Array.isArray(tags))
let arrOfTags = Array.from(tags);
console.log(arrOfTags);
console.log(Array.isArray
(arrOfTags))
arrOfTags.map(tag => {
    console.log(tag)
})
```

```
Default levels ▼   No Issues
▸NodeList(4) [h1, h2, div, p]                App.js:28
⚠ DevTools failed to load source map: Could not load
content for chrome-extension://
```

5. document.querySelector :

```
//!5 document.
querySelector:returns the first
element that is a descendant of
node that matches selectors.
let a = document.querySelector
("#test");
console.log(a);
```

```
efault levels ▼   No Issues
<p id="test">para</p>                    App.js:38
```

Note : We can pass id, class , tag in the argument .

```
let a = document.querySelector(".
test")
console.log(a)
```

```
ault levels ▼   No Issues
<a href class="test">link</a>
DevTools failed to load source map: Could not l
```

```
let a = document.querySelector("a")
console.log(a)
```

```
<a href class="test">link</a>
DevTools failed to load source map: Cou
```

6. document.querySelectorAll() :

```
//!6.document.querySelectorAll()
:returns NodeList()
let tags = document.
querySelectorAll("#test");
console.log(tags)
```

```
Default levels ▾   No Issues

▾NodeList(3) [a#test, p#test, div#test]   App.js:42
  ▸0: a#test
  ▸1: p#test
  ▸2: div#test
   length: 3
  ▸[[Prototype]]: NodeList
```

```
let div = document.querySelector
("div");
console.log(div);

let divs = document.
querySelectorAll("div");
console.log(divs);

let divTags = document.
getElementsByTagName("div");
console.log(divTags);
```

```
                                   App.js:46
▸NodeList(10) [div, div, div, div, div, div, div, div,
div, div]
                                   App.js:51
▸HTMLCollection(10) [div, div, div, div, div, div, div,
div, div, div]
⚠ DevTools failed to load source map: Could not load
```

DOM Properties

document.innerHtml vs document.innerText :

```
DOM properties > JS App.js > ...
//!DOM properties
//?difference between innerHTML
and innerText

let div = document.querySelector
("div");
// div.innerHTML = "hello";
// div.innerText="hello"
div.innerText = `<h1>hello</h1>
<ul>
<li>1</li>
<li>2</li>
</ul>`;
```

127.0.0.1:5500/DOM/DOM%20properties

```
<h1>hello</h1>
<ul>
<li>1</li>
<li>2</li>
</ul>
```

```
div.innerHTML = `<h1>hello</h1>
<ul>
<li>1</li>
<li>2</li>
</ul>`;
```

firstChild , firstElementChild , lastChild, lastElementChild :

```
<body>
  <!-- <div></div> -->
  <ul>first <li>1</li>
    <li>2</li>
    <li>3</li>last</ul>
  <script src="./App.js"></
  script>
</body>
```

```
let ul = document.querySelector
("ul");
console.log(ul.firstChild);
console.log(ul.lastChild);
console.log(ul.firstElementChild);
console.log(ul.firstElementChild.
innerHTML);
console.log(ul.lastElementChild)
console.log(ul.lastElementChild.
innerText);
```

```
"first "
"last"
▶<li> ··· </li>
1
```

```
console.log(ul.children);
console.log(document.body.children)
```

```
▶ HTMLCollection(3) [li, li, li]        App.js:21
▶ HTMLCollection(2) [ul, script]        App.js:22
```

Document.createElement()

```
//!7. document.createElement();
let table = document.createElement
("table");
let tr1 = document.createElement
("tr");
let tr2 = document.createElement
("tr");
tr1.innerHTML = `<td>1</td>
<td>2</td>
<td>3</td>`;
tr2.innerHTML = `<td>4</td>
<td>5</td>
<td>6</td>`;
document.body.appendChild(table);
table.appendChild(tr1);
table.appendChild(tr2);
```

Events

1. onclick :

```
//!DOM events
//?onclick
let btn = document.querySelector
("button");
console.log(btn);
btn.addEventListener("click", e =>
{
  e.preventDefault();
  let inputTags = document.
  querySelectorAll("input");
  console.log(inputTags);
  let inputs = Array.from
  (inputTags);
  console.log(inputs);
  inputs.map(inputVal => {
    console.log(inputVal.value);
  });
});
```

```
//?2.by using onclick as a property
let btn = document.querySelector
("button");
btn.onclick = function (e) {
  e.preventDefault();
  let inputTags = document.
  querySelectorAll("input");
  console.log(inputTags);
  let inputs = Array.from
  (inputTags);
  console.log(inputs);
  inputs.map(inputVal => {
    console.log(inputVal.value);
  });
};
```

```
//?3. by giving onClick as atr in
button tag

function clickMe() {
  console.log("clicked");
}
```

```
</form>

<button onclick="clickMe()
">LOGIN</button>
```

2. onsubmit

```
//! onsubmit
let form = document.querySelector
("form");
console.log(form);
form.addEventListener("submit", e
=> {
  e.preventDefault();
  let inputTags = document.
  querySelectorAll("input");
  console.log(inputTags);
  let inputs = Array.from
  (inputTags);
  console.log(inputs);
  inputs.map(input => {
    console.log(input.value);
  });
);
```

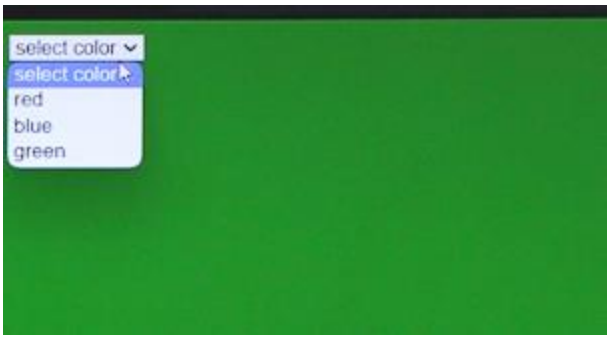
Note :

onsubmit is used only for form , when we are using buttons then we can use onclick .

3. onchange :

```
//!onchange
let select = document.querySelector
("select");
select.addEventListener("change",
e => {
  document.body.style.
  backgroundColor = e.target.value;
});
```

```
<body>
  <select>
    <option
    value="initial">select
    color</option>
    <option value="red">red</
    option>
    <option value="blue">blue</
    option>
    <option value="green">green</
    option>
  </select>

  <script src="./App.js"></
  script>
</body>
```



4. keyup and keydown :

```
//!keyup and keydown
let input = document.querySelector
("input");
console.log(input);
input.addEventListener("keyup", ()
=> {
  console.log("keyup triggered");
});
input.addEventListener("keydown",
() => {
  console.log("keydown triggered");
});
```

```
//!keyup and keydown
let input = document.querySelector
("input");
console.log(input);
input.addEventListener("keyup", ()
=> {
  console.log("keyup triggered");
  let div = document.querySelector
  ("div");
  div.innerHTML = input.value.
  toUpperCase();
});
input.addEventListener("keydown",
() => {
  console.log("keydown triggered");
});
```

Shiva | 123 | SUBMIT

Elements | Console | Sources » 4

top ▼ | -url:http://127.0.0.1:5500/DOM | 4 hidden

Default levels ▼ | No Issues

for chrome-extension://fheoggkfdfchfphceeifdbepaooicaho/sourceMap/chrome/scripts/iframe_form_check.js.map: System error: net::ERR_BLOCKED_BY_CLIENT

⚠ DevTools failed to load source map: Could not load content for chrome-extension://fheoggkfdfchfphceeifdbepaooicaho/sourceMap/chrome/scripts/iframe_form_detection.js.map: System error: net::ERR_BLOCKED_BY_CLIENT

▸ NodeList(2) [input, input] App.js:8

▸ (2) [input, input] App.js:10

Shiva App.js:12

123 App.js:12

```
input.addEventListener("keydown",
() => {
  console.log("keydown triggered");
  document.body.style.background =
  `#f${Math.round(Math.random(10)
  * 1000)}`;
});
```

5. mouseover and mouseout :

```
//!mouseover and mouseout
let btn = document.querySelector
("button");
btn.addEventListener("mouseover",
() => {
  //   alert("hello");
  document.body.style.background =
  "red";
});
btn.addEventListener("mouseout",
() => {
  document.body.style.background =
  "green";
});
```

This Keyword

```
hisKeyword.js
console.log(this);
console.log(window);
console.log(this == window);
```

```
thisKeyword.js:1
▶ Window {window: Window, self: Window, document:
  document, name: '', location: Location, …}
thisKeyword.js:2
▶ Window {window: Window, self: Window, document:
  document, name: '', location: Location, …}
true                                  thisKeyword.js:3
```

```
//?this keyword in regular function with and without "use strict" mode
function test() {
  console.log(this);
}
test();

function demo() {
  "use strict";
  console.log(this); //?undefined
}
demo();
```

```
thisKeyword.js:7
▶ Window {window: Window, self: Window, document:
  document, name: '', location: Location, …}
undefined                                thisKeyword.js:13
>
```

```
//?this keyword in arrow function
with and without "use strict" mode

let arrFun1 = () => {
  console.log(this);
};
arrFun1();
let arrFun2 = () => {
  "use strict";
  console.log(this);//? no change
};
arrFun2();
```

If we use arrow function then this keyword won't care about "use strict" mode , it will print window object .

```
                                        thisKeyword.js:20
▶ Window {window: Window, self: Window, document:
  document, name: '', location: Location, …}
                                        thisKeyword.js:25
▶ Window {window: Window, self: Window, document:
  document, name: '', location: Location, …}
>
```

```
29  //?this keyword inside objects
30  let obj1 = {
31    k1: function () {
32      return this;
33    },
34    k2: () => {
35      return this;
36    },
37  };
38  console.log(obj1);
    console.log(obj1.k1());
    console.log(obj1.k2());
```

If we use regular function in the object and use this keyword it will point to the current object but if we use arrow function and use this keyword it will point to the window object .

```
▶ {k1: ƒ, k2: ƒ}                        thisKeyword.js:38
▶ {k1: ƒ, k2: ƒ}                        thisKeyword.js:39
                                        thisKeyword.js:40
▶ Window {window: Window, self: Window, document:
  document, name: '', location: Location, …}
```

```
60  //?this keyword inside DOM events
61  let btn = document.querySelector
    ("button");
62  btn.addEventListener("click", ()
    => {
63    console.log(this); //window obj
64  });
65  btn.addEventListener("click",
    function () {
66    console.log(this);//points
      towards the current DOM elements
67  });
```

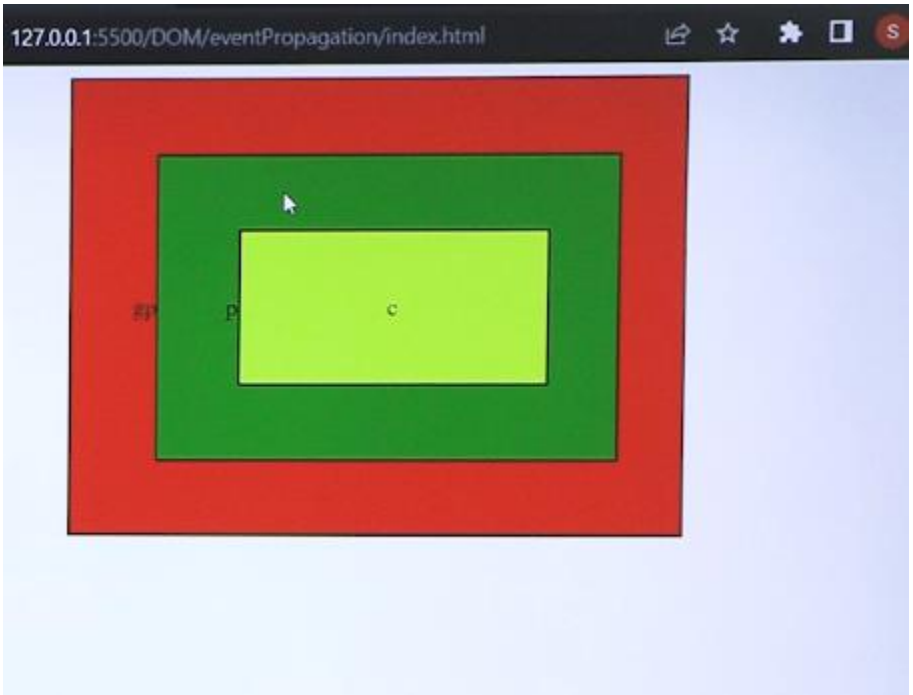
```
                                    thisKeyword.js:63
▶ Window {window: Window, self: Window, document:
  document, name: '', location: Location, …}
  <button>clickMe</button>          thisKeyword.js:66
```

Event Propagation

```
eventPropagation > JS App.js
//!event propagation
/*
1.capturing phase
2.target phase
3.bubbling phase
*/
```

```
let gp = document.getElementById
("gp");
gp.addEventListener("click", e => {
  console.log(e);
  gp.style.background = "red";
  console.log(`Grand parent
  executed at the phase ${e.
  eventPhase}`);
});

let p = document.getElementById
("p");
p.addEventListener("click", e => {
  console.log(e);
  p.style.background = "green";
  console.log(`parent executed at
  the phase ${e.eventPhase}`);
});
```

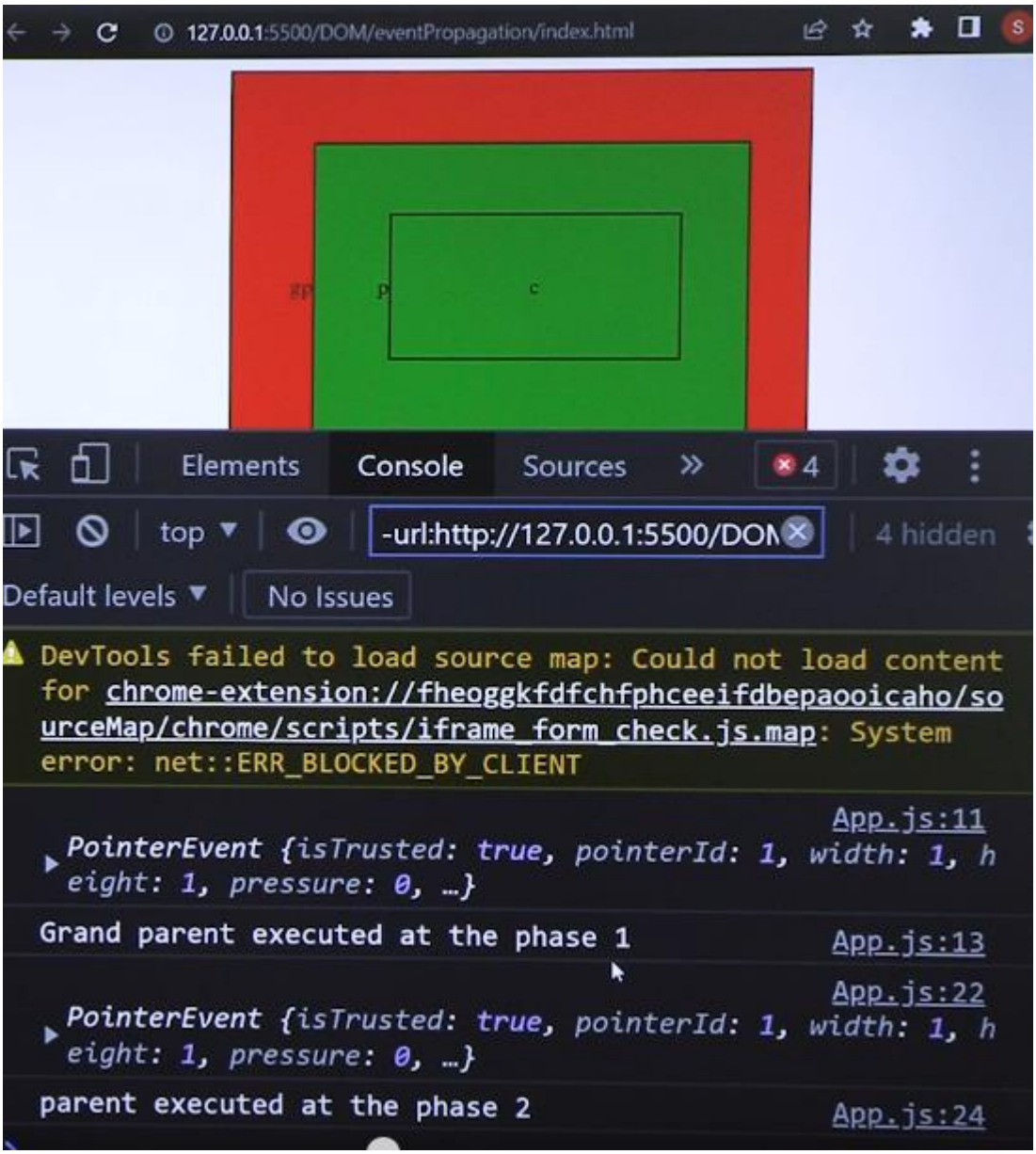
```
let c = document.getElementById
("c");
c.addEventListener("click", e => {
  console.log(e);
  c.style.background = "yellow";
  console.log(`child executed at
  the phase ${e.eventPhase}`);
});
```



By default the value is false for that it is executing during bubbling phase (3).

If we make it true , then it will execute during capturing phase (1).

```
let c = document.getElementById
("c");
c.addEventListener(
  "click",
  e => {
    console.log(e);
    c.style.background = "yellow";
    console.log(`child executed at
    the phase ${e.eventPhase}`);
  },
  true
);
```



If we give true means we are stopping the propagation during capturing phase .

If we give false it will stop the propagation during bubbling phase .

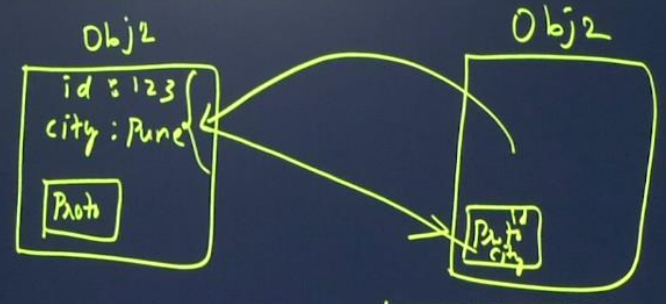
```
let p = document.getElementById
("p");
p.addEventListener(
  "click",
  e => {
    e.stopPropagation();
    console.log(e);
    p.style.background = "green";
    console.log(`parent executed
    at the phase ${e.eventPhase}`);
  },
  false
);
```

For stopping propagation we have to use e.stopPropagation();

OOP's in JS

1. Inheritance
2. Abstraction
3. Polymorphism
4. Encapsulation

1 Inheritance (prototype)



clg (obj2.id) // 123

$\boxed{\text{Obj2.__Proto__} = \text{Obj 1}}$

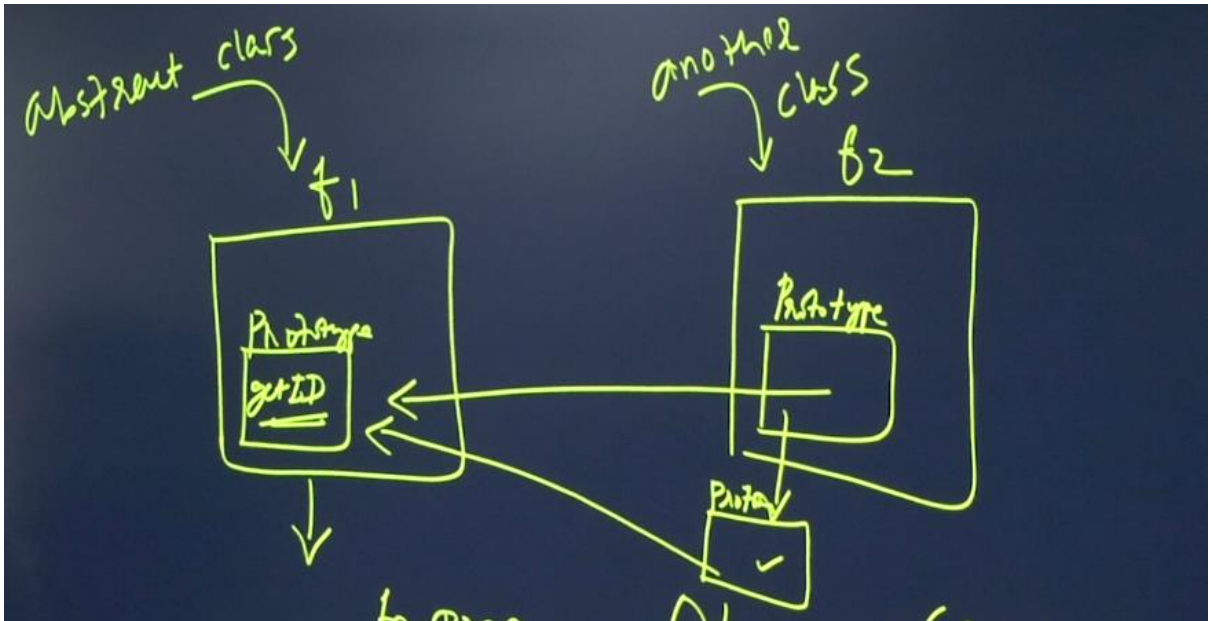
Inheritance

```
Inheritance using prototype

let ob1 = {
    id : 1,
    city: "bangalore",
    company:"TYSS"
}

let ob2 ={
   id:2
}

console.log(ob1)
console.log(ob2)
console.log(ob1.__proto__ == ob2.__proto__)

ob2.__proto__ = ob1;

console.log(ob2.city);
console.log(ob2.id);

console.log(ob1.__proto__ == ob2.__proto__)
```

```
▶ {id: 1, city: 'bangalore', company: 'TYSS'}      script.js:15
▶ {id: 2}                                          script.js:16
true                                               script.js:17
bangalore                                          script.js:21
2                                                  script.js:22
false                                              script.js:24
```

Abstraction in JS

- restricted class
- We can not create an instance / object
- By using another class one can access the abstract class.




```
f2.prototype = Object.create(f1.prototype)
let obj = new f2(123)
(lg(obj.getID)) → 123
```

Abstraction

1. restricted Class
2. we can't create an instance / object .
3. By using another class one can access the abstract class by using prototype object .
4. In JS we don't have any abstract keyword .
5. Through Constructor function we can achive the Abstraction Class

creating object using constructor function

```
function f1(id)
{
    this.id = id;
    throw new Error("can't create an instance for abstraction class");

}

// let obj = new f1()   //? it will throw the error
```

adding method to abstraction class using prototype object .

```
f1.prototype.getId = function()
{
    return `the id is : ${this.id}`;
};
console.log(f1.prototype)
```

creating another object

```
function f2(id){
    this.id = id ;
}

console.log(f2.prototype);

f2.prototype = Object.create(f1.prototype) ;

let ob1 = new f2(333);
console.log(ob1);

console.log(ob1.getId());
```

```
▶ {getId: ƒ, constructor: ƒ}        script.js:28
▶ {constructor: ƒ}                  script.js:36
▶ ƒ2 {id: 333}                      script.js:41
the id is : 333                     script.js:43
```

Signup and Login Page

Signup.js

```
let form = document.querySelector("form");

form.addEventListener("submit", (e) =>{

  e.preventDefault();

  let un = document.getElementById("uName").value;
  let pn = document.getElementById("pNum").value;

  console.log(un);
  console.log(pn);

  if(un == "" && pn == "")
    alert("both user name and phone number can not be empty");
  else if(un == "" )
    alert(" user name  can not be empty");
  else if(pn == "" )
    alert(" phone number  can not be empty");
  else if(un !== "" && pn !== "")
   {
       localStorage.setItem("user",un);
       localStorage.setItem("phone",pn);
       open("./login.html");
   }
   else
   alert("Registration Failed ... ");




})
```

Login.js

```
let form = document.querySelector("form");

form.addEventListener("submit",(e)=>{
    e.preventDefault();
  let  un = document.getElementById("un").value ;
  let  pn = document.getElementById("pn").value ;

  let un1 = localStorage.getItem("user");
  let pn1 = localStorage.getItem("phone");

  if(un == "" && pn == "")
    alert("both user name and phone number can not be empty");
  else if(un == "" )
    alert(" user name  can not be empty");
  else if(pn == "" )
    alert(" phone number  can not be empty");
  else if(un !==un1 || pn !==pn1 )
      alert("Wrong Credentials");
  else if(un ==un1 && pn ==pn1 )
  {
    alert("Successfully Logedin ");
    open("./home.html");

  }





})
```

Signup.html

```html
<body>
    <form action="">
        <input type="text"  id="uName" placeholder="Enter your name">
        <input type="text"  id="pNum" placeholder="Enter your phone number">
        <button>REGISTER</button>

    </form>


    <script src="./signup.js"></script>
</body>
```

Login.html

```html
<body>

    <form action="">
        <input type="text"  id="un" placeholder="Enter your name">
        <input type="text"  id="pn" placeholder="Enter your phone number">
        <button>LOGIN</button>
    </form>
    <script src="./login.js"></script>
</body>
```

Home.html

```html
<body>
    <h1></h1>

    <script src="./home.js"></script>
</body>
```

Home.js

```js
let h1 = document.querySelector("h1");

let userName = localStorage.getItem("user");

h1.innerHTML = `Logged in as ${userName} `;
```