CITY OF RALLIES

Delhi, India's capital territory, is a massive metropolitan area in the country's north. In Old Delhi, a neighborhood dating to the 1600s, stands the imposing Mughal-era Red Fort, a symbol of India, and the sprawling Jama Masjid mosque, whose courtyard accommodates 25,000 people. Nearby is Chandni Chowk, a vibrant bazaar filled with food carts, sweets shops and spice stalls.

JavaScript

13 :- Java script LANGUAGE :- 5/1/2023

Introduction to Java script :-

(1) Java script is scripting language, which is used to frontend and Backend language.

(2) Java script ~~and~~ come under the front end language.

(3) Java script initial first name is mocha 1st name

front End language	Backend language
Java script ↓ node JS React JS	Javascript node JS frame work.

Invented
(A) Brandon Eich
1995.

(4) first name is mocha

(5) after it will live script.

(6) after it will become Java script.

Java script :-

"Js is a scripting language that enables you to create dynamically updating content, control multimedia, image run the functionalities."

Introduction :-

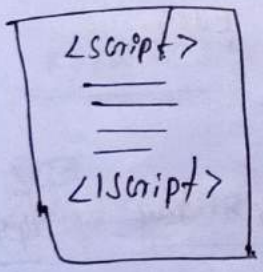
1. Java script we as front end language. as well as backend language.
2. In the front end we used it as ~~[redacted]~~, React JS.
3. In the back end we used it as node JS.
4. Java script was invented by Brendan EICH in the year of 1995.
5. The netscape who has standardized browser ~~day by~~ they hired a programmer called Brandon EICH he invented the scripting language called Js within 10 days
6. The first name of Js was mocha.
7. later named it has live script.
8. They didn't get any hype by keeping live script. all that time Java was Booming. They replace live script as Java script.
9. Java script was used we as functionality for the websites.

89%

1. almost, 60 to 70% websites are using Javascript.
2. Java script doesn't need any development tool.
3. It is run on by having browser.
4. Java script run outside the browser by using node JS.

Embedded with C++ :

Syntax of Javascript :

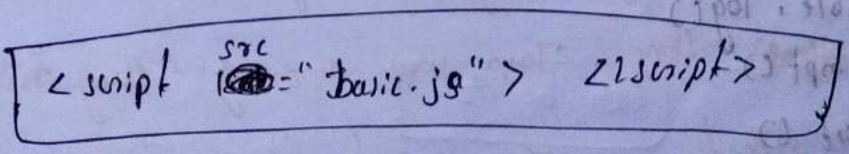


← Scripting language.

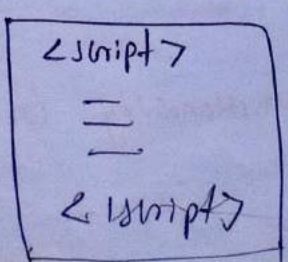
Two types of writing a JS code.

1. by creating external JS like with extension .js
2. by taking script Tag exactly before the body close inside script Tag we write JS code.

1]



2]



① Every browser has Javascript Engine to run the Javascript code.

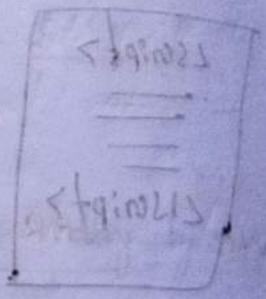
② Chrome has V8 Engine Javascript.

③ Firefox has spyder monkey, spyder monkey is Engine of firefox.

④ Internet explorer has Javascript core.

KEYWORDS :-

① In Java script we have 3 keywords.



1] LET

2] CONST

3] VAR

:- OUTPUT METHODS OF JS :-

1) Document.write();
2] Document.writeln()
3) console.log()
4] prompt()
5] Alert()
6] confirm()

console.log() :-

(1) console.log() is a developer output.

prompt() & Alert() :-

(2) prompt() & alert() are the pop up messages output.

(3) prompt() is take input from user

KEYWORDS :-

Steps	VAR	LET	CONST
Declaration →	var a;	let a; ✓	const a; ✓
Initialization →	a=12; ✓	→ a=12 ✓	a=12 ✗
W & I →	var a=12 ✓	let a=12 ✗	const a=12 ✓
R.D, →	var a; ✓	let b=12 ✓	const a ✗
R.I →	a=14 ✓	let a ✗	a=12; ✗
R.D & R.I	var a=14 ✓	a=14 ✓	
		let a=;	const a=12 ✗
			const 2=14 ✓

Keywords :-

1) VAR :- var is a keyword it is a variable which is used to store the data, named memory locations.

steps :-

syntax :- `var a;` // variable Declaration

examples :-

`a=12` // variable Initilization

`var a=12` // variable Declaration & Initilization both.

`a=14;` // variable Re initilization

2) LET :- LET is a keyword it is used to store the data The syntax of Let is bellow:

syntax :-

`let a;` // variable Declaration

examples :-

`a=12` // variable Initilization

`let b=12` // variable Declaration & Initilization in One line.

`a=15` // variable Re initilization

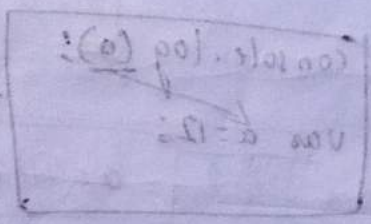
Note ; variable Declaration & initilization we did not use same variable name we will use another variable name.

3) CONST : const is a keyword it is used to store data. The syntax of const is bellow:

Syntax :

```
const  a = 14      // Variable Declaration & initilization
```

④ Note : variable Declaration & initilization at one line it is possible.



7/1/2023

Hoisting :-

"Hoisting is a process of utilizing the variable before Declaration and initialization is called as Hoisting.

Note:- we can Hoisting only VAR keyword.

```
var a = 12;
```
→ global variable / scope.

```
console.log(a);
var a = 12;
```
→ Hoisting

1] VAR

```
console.log(b)
var b = 100;
```

note: we can't Hoisting Let and const keyword.

why because Let & const are Local variables or scope.

Hoisting:

① Utilizing the variable before Declaration and initilization is called as Hoisting.

② we can achive Hoisting VAR keyword because VAR is a global scope

syntax:

```
console.log(a);
var a=100;
```

↓

global scope variable.

example:

```
console.log(b)
var b=500;
```

Note:

① we can't Achive Hoisting WITH LET KEYWORD. because a LET KEYWORD is a Local scope / variable.

Example:

```
console.log(b)
let b=100; // U.D Undelined Reference Error.
```

if you try to Hoisting we get uncaught Reference.

② we can't Hoisting CONST KEYWORD / VARIABLE.

② : If you try to Hoisting with CONST KEYWORD. We will get uncaught Reference error.

Ex:

```
console.log(d)
const d=17;
```

~~TDZ :- (Temproal Dead Zone) :-~~

~~" The time taken blw the console.log() stmt and Declaration & initialization stmt.~~

TDZ :- (Temporal Dead Zone) :-

```
{console.log(b);
let b=12;  //TDZ
}
```

When we try to Hoisting the time taken b/w the console.log() stmt and Declaration & initialization stmt.

② we can achive this TDZ only in LET And CONST Keyword

② we can't achive TDZ with VAR Keyword, because it is we get them undefined as an output.

Global Execution context :-

9/1/2023

GEC :- Global Execution context

~~Java script Engine~~

① "Every browser has a Javascript Engine_ it takes the code to convert from Javascript code to machine understandable code."

② Java script Engine takes Js code, scan for two times and split into two ~~times~~ phase.

1] variables phase 2] Execution or functional phase.

1] Variables phase :-

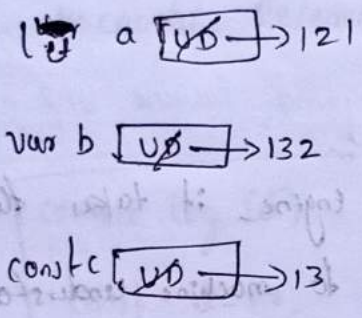
① All the variables are undefined.

2] Execution or functional phase :-

① Execution or functional phase all the variables are initilized with some values.

② All the console.log() stmt present in execution or functional phase.

Variable phase



Execution / functional phase

Console.log(b)

```
a=121;
b=132;
console.log(a);
c=13;
Console.log(b);
Console.log(c);
```

Type casting :-

converting one type of data to another type is called type casting.

we have two type casting :-

1) Implict type casting.
2) Explict type casting.

1) Implicit type casting :-

converting one type of data to another type of data implicitly / internally is called as Implicit type casting.

example :-

```
console.log(10+10); //20
console.log(10+"10"); 1010
let a = 100;
console.log(a typeof a); // 100 'number'
console.log(a+"hi"); // 100 hi
console.log("10"+"10"); //1010
let b = "Uma shankar";
console.log(b, typeof b); Uma shankar string
```

typeof is a keyword which type of data it indicates

Explicity type casting :-

we as a developer converting one type of data to another type of data forcefully is called as explicity type casting.

it's converting number to string

Example :-

```
let r = 101;
console.log (r, typeof r);
```

// number.

```
let r = string (101);
console.log (r, typeof r);
```

// string
// 101 string.

note :-

converting string to number it shows NAN

```
let r = string (101);
console.log (r, typeof r); // 101 string

let r1 = Number ("ABC");
console.log (r1, typeof (r1) // ABC Number

let r3 = string (true);
console.log (r3, typeof r3) true string.
```

Operators

"Operator are symbols which we used to perform some operations on operators."

Types of operators

1) Arithamelic operators → +, -, *, /, ++, --
2) Assignment operators → =, +=, *=, %=.
3) comparision operator → ==, !=, <

1] Arithmetic operator

+, -, *, /, %

2] Relational operators

<, >, <=, >=, ==, !=, ≤, ≥,

3] Logical operator

1) NOT → 1 i/p → 1 o/p
2) AND → 2 i/p → 1 o/p
3) OR → 2 i/p → 1 o/p

NOT

0	1
1	0

AND

0	0	0
0	1	0
1	0	0
1	1	1

OR

0	0	0
0	1	1
1	0	1
1	1	1

Examples on operators:-

1] Arithmetic operations:-

```
Let a = 10;
Let b = 5;
console.log("a+b is", a+b); //15
console.log("a-b is", a-b); //5
console.log("a*b is", a*b); //50
console.log("a/b is", a/b); 2
```

2] Relational operators:-

```
Let a = 10;
Let b = 20;
console.log("a<b is", a<b); //10
console.log("b>a is", b>a); //20
console.log("a<=b is", a<=b); //false
console.log("a>=b is", a>=b); //false
console.log("a==b is", a==b); //false.
console.log("a!=b is", a!=b); //false.
```

```
console.log("++a is", ++a); //11
console.log("++b is", ++b); //21
console.log("a++ is", a++); //11
console.log("b++ is", b++); //21
```

```
x = 4;
x += 2;
x = x+2;
x = x+2;
Document.write(x); //10
```

```
x**=4
console.log(x);
```

```
x1=5
x=x15;
console.log(x);
```

13/1/2023

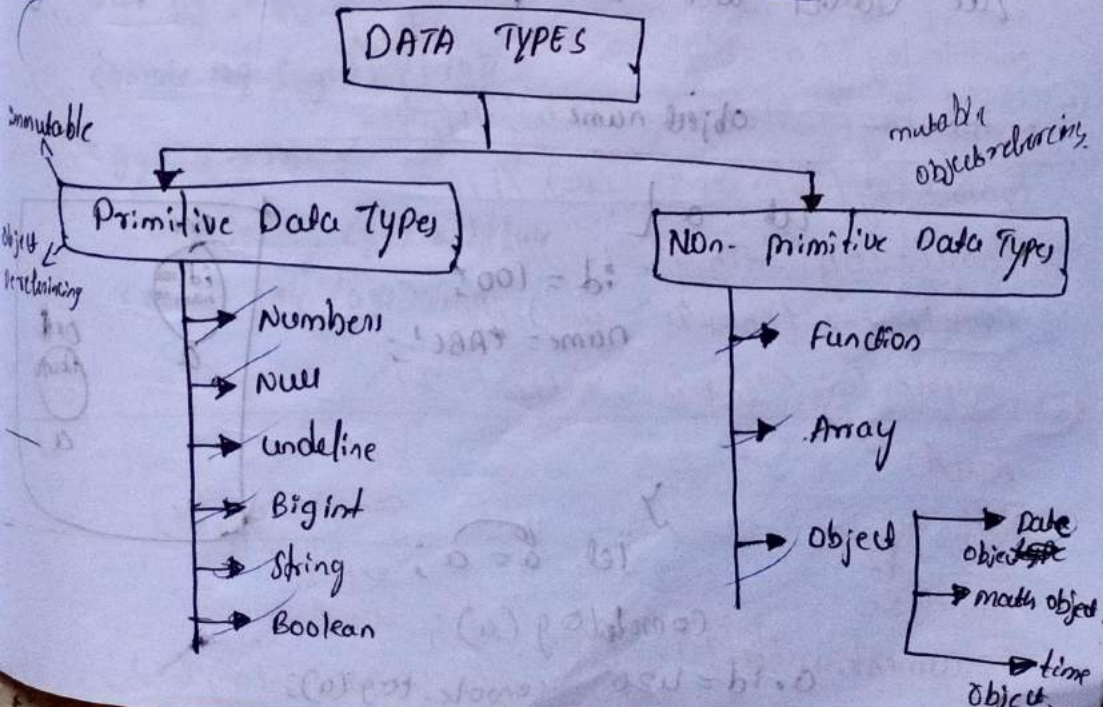
DATA TYPE'S :-

"Data type is nothing but, which type of data the data specifies the values would be stored. and it is manipulated and memory allocation."

In Data types. There are two Data Types
1) Primitive and 2) Non-primitive Data type.

1) Primitive Data type :- primitive Data type is immutable and object Dereferencing. [values can't be altered].

2) Non-primitive Data type :- Non-primitive Data type is also called object referencing and it is also called as mutable. [values can't altered].



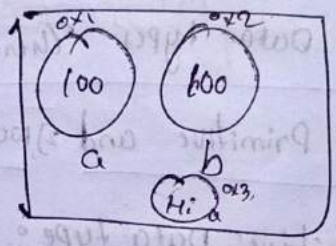
Note:

Immutable :

(1) Immutable is when we want alter the value of an variable it doesn't updating the existing the memory.

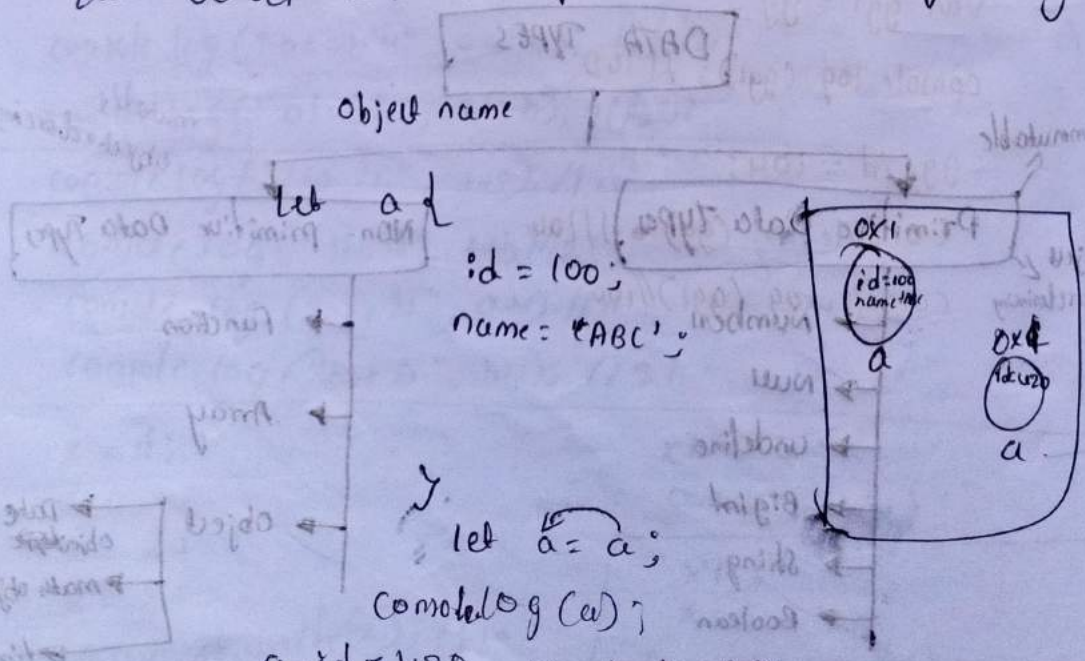
(2) : it creates another memory space.

```
let a = 100
b = a;
let a = "Hi";
console.log (b);
```



: mutable :

(1) : if you want to alter the values in the non-primitive Data type it won't create another memory space. The values will be updated in the existing memory.



//immutable

```
var bb=10;
  cc = bb; // c <-> 10
var bb =20;
var bb=80;
console.log(bb);   //30.
```

//mutable

```
Var gg {
      id = 100;
      name:'abc';
    }
console.log(gg);   //100
var gg1 = gg;
console.log(gg1); //100
gg.id = 104;
Console.log(gg) = //104
console.log(gg1)//100
```

Primitive Data types :-

1) Number :-

(A) Number is integer type of data.

(A) in javascript all decimal integer number value considers it has only a number.

Examples :

(1) let a = 100;
console.log(a); // 100

(2) let b = 10.20;
console.log(b); // 10.2

(3) let c = 10/5;
console.log(c); // 2

(4) let d = string(100)
console.log(d, typeof d)
// 100 string.

2) Null :-

(A) Null is primitive data types & it is a Bug (error) because type of null is object.

(A) object is non-primitive data type of & null doesn't comes any memory space into the database.

3) Undefined :-

(A) Undefined is primitive data type, it is a variable Declaration part is there, & There is no initialization of variable is called as Undefined Variable.

for ex :

```
let b; // undefined
```

6) BigInt: BigInt is primitive type. it is integer type of data type. which we used the ranges of integer we can store more integer values in a variable. we can use BigInt datatype

ex:

① let a = 100; // integer type of data
console.log(a); // 100

② let a = BigInt (1000000000);
console.log(a); // 1000000000n

③ var v = BigInt (20000000000);
consol.log(v, typeof v);

7) string:

① string is a primitive data type. string is a anything we can written in a double quote & single quote and back tick is called as string.

for ex:

"NAGARAJA", 'NAGARAJA' ~NAGARAJA~

"nagaraja", 'nagaraja', ~nagaraja~

ex:

① var v1 = "NAGARAJA";
console.log(v1); // NAGARAJA.

② let b = 'nagaraja';
Console.log(b); nagaraja

③ let c = ~nagaraja~;
console.log(c);
// nagaraja.

var d = "nagaraja";

console.log(d, typeof d); // 'nagaraja' string.

(R) we can't convert string to number. it shows Nan.

var c = Number("Hi");

Console.log(c, typeof c); // Nan number

Interpolation:-

(R) Interpolation is a fetch the variable data.

(R) Interpolation representing in ${} ;

ex: let a = "motera";
let b = "v k";
let c = 200;
console.log(`${a} player is ${b} he scored ${c}`);

o/p
motera player v k he scored 200

String methods:-

1) console.log(b.length); // length of string
2) console.log(b.repeat(3)); // repeat.
3) console.log(b.toUpperCase());
4) Console.log(b.toLowerCase());
5) Console.log(b.indexOf());
6) Console.log(b.includes('d'));
7) console.log(b.startsWith('M'));
8) console.log(b.EndsWith('d'));
9) console.log(b.concat('bad'));
10) console.log(b.padStart(15,'@'));
11) console.log(b.padEnd(15,'?'));
12) Console.log(b.split("").reverse().join(""));
13) Console.log(b.substring(2,5));

Non primitive Data Type

18/1/2023

1) function
2) Array
3) object.

1) function :- ~~A function doesn't have any name function that is called as function.~~

A function is nothing but a block of stmt which perform some operations is called as function.

Function Types :-

1) Anonymous function.
2) Named function.
3) Function with expression.
4) First class function.
5) Arrow function.
6) Higher order function.
7) Nested function.
8) IIFE (Immediate Invoke function Expression).
9) Callback function.

Non-primitive Data Type's :

1] Function 2] ARRAY 3] OBJECT

1] Function :-

(K) Function is we to execute set of stmt's or instruction. it called as whenever it is called invoked.

(K) In Javascript we have 9 function.

1] Anonamous Function.
2] Named Function.
3] Function with expression.
4] First class Function.
5] Arrow function.
6] Higher order function.
7] Nested function.
8] IIFE (Immediate Invoke function Expression).
9] callback Function.

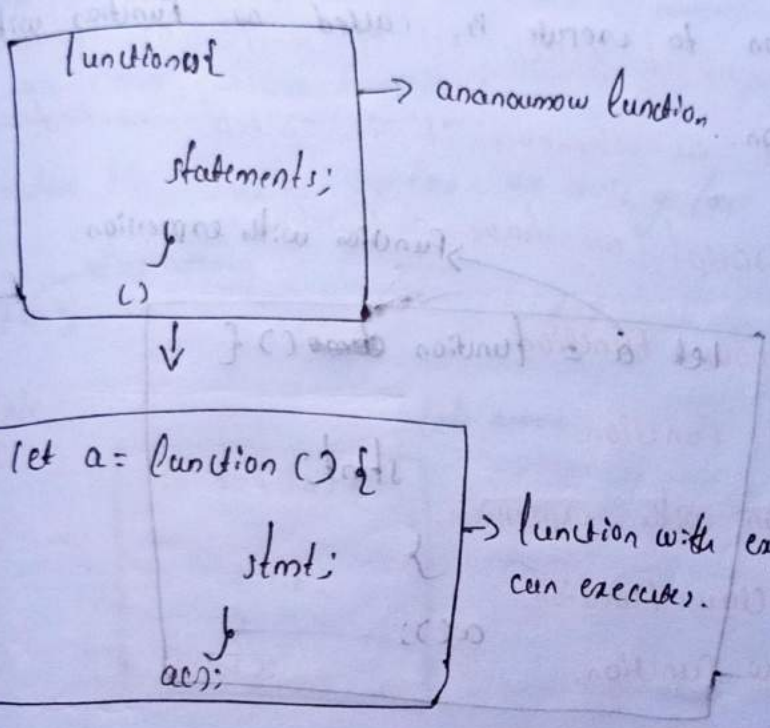
1] Anonamous function :-

(K) A function doesn't have a function name is called as Anonamous function.

(K) we can't execute anonamous function

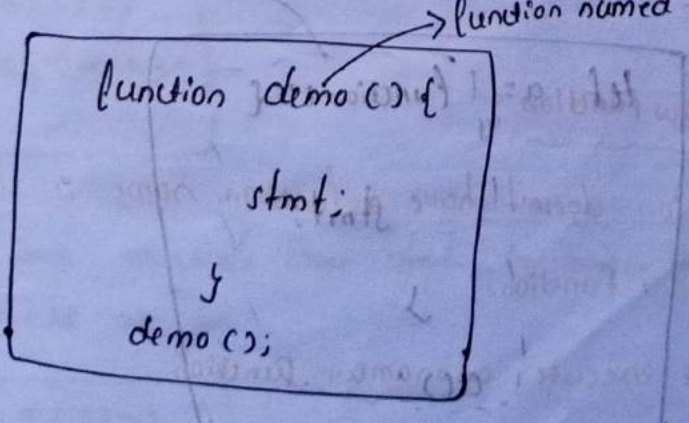
(K) To make that execute we are using function with expression.

ample :-



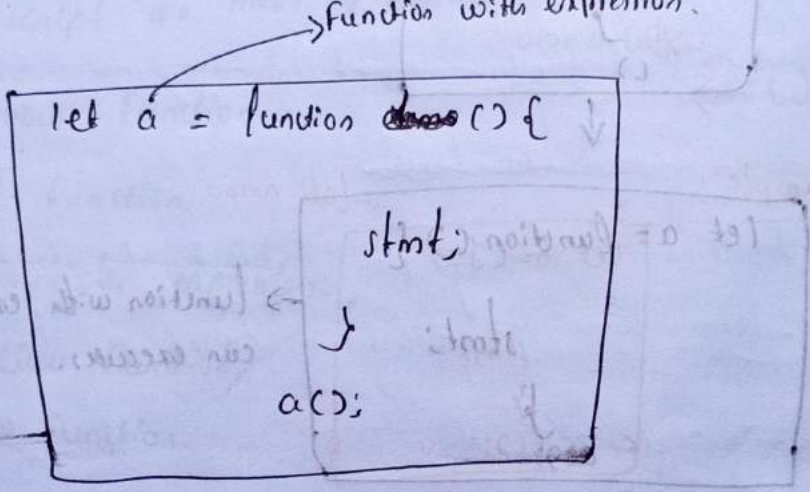
i) Named function :- A function which a function name is casted as Named function.

ample



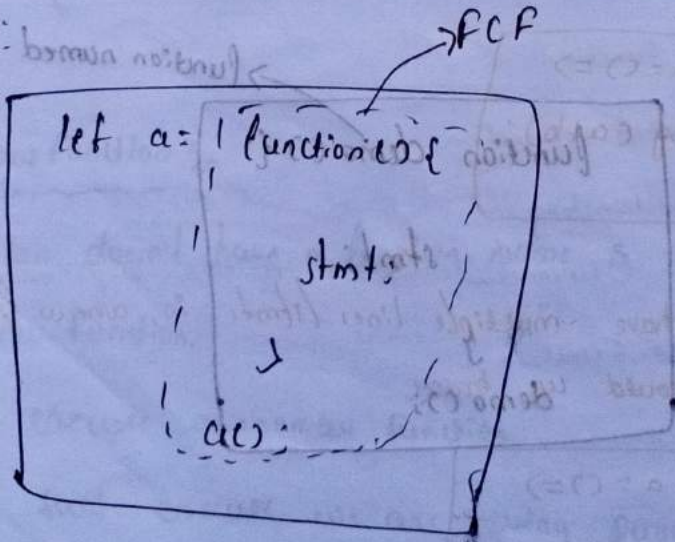
3] Function with expression :- A function which has an expression to execute is called as function with expression.

Example:



4] First class function :- A function passing as a value to a variable / expression is called as first class function.

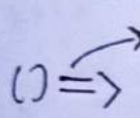
example:



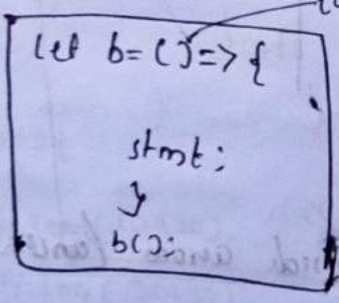
1) Arrow Function :- A function having a fat arrow is called as Arrow function.

2) Why we need arrow function :-

1) to reduce the function syntax we will go for arrow function.



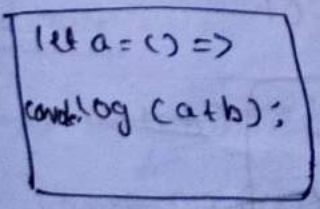
example :



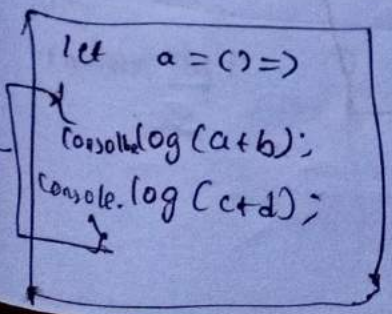
Arrow function rules :-

1) when we have single stmt there is no need to write braces. execute code.

eg :



2) When we have multiple lines / stmts in arrow function must & should use braces.



3] When we have return keyword in arrow function must a should use log stmt out side the arrow function.

ex:

```
let a=()=>{

    console.log(2+3)

    return

}

console.log(a);
```

4] When we have log stmt inside arrow function no need to a log stmt outside arrow function.

ex:

```
let a=()=>{

    console.log(a)

}

a();
```

① First rule :-

```
let k1 = () =>

    console.log (10+20);

k1()
```

O/P

30

② Second rule :

```
let k2 = () => {
    console.log (10+20);
    console.log (20+30);
    console.log (30+50);
}
console.log(k2());
```

O/P

30
50
80

③ Third rule :-

~~let k3 = (a, b) => {~~

~~console.log (10+20);~~
~~return a+b;~~

~~}~~

~~console.log~~

```
let k3 = (a, b) => {

    return 10+20;

}
console.log (k3());
```

O/P

30

4] let ku = (a, b) => {

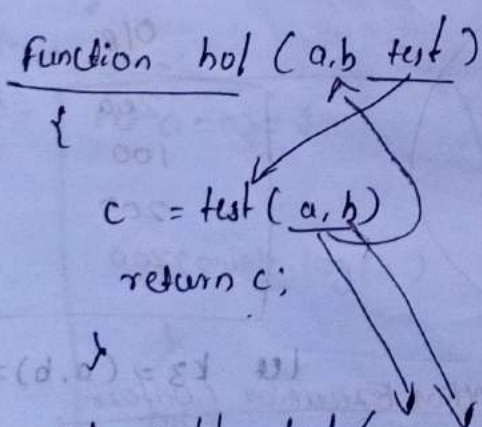
```
console.log(50+50);    //100.
}
ku();
```

6] Higher order function :

A function which takes another function as an Argument is called as Higher order function.

Example:

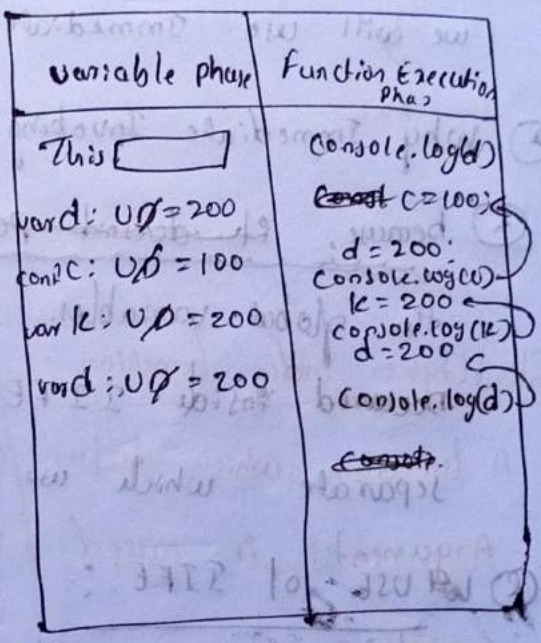
```
function hol(a,b, test)
{
    c = test(a,b)
    return c;
}
let add = hol(20, 30, function(a,b){
{
    return (a+b);
})
console.log(add());
```



o/p
50

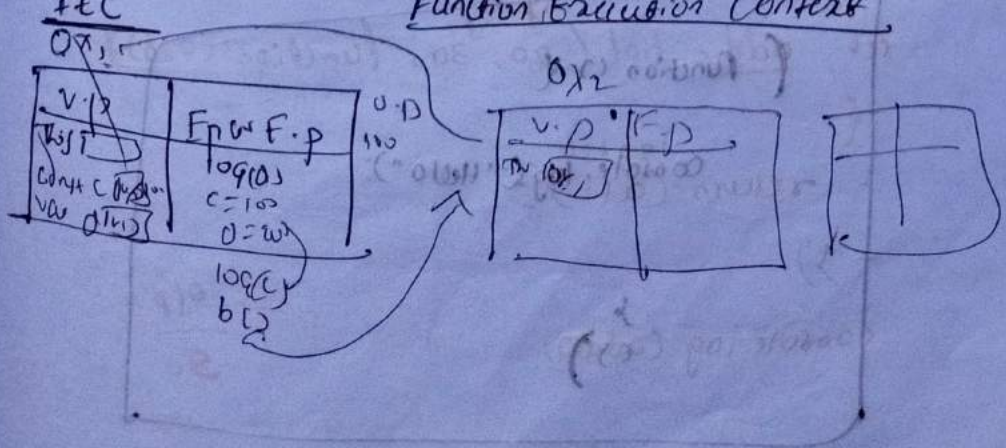
GEC:

```
console.log(d);
const c=100;
var d=200;
console.log(c);

function b(){
    var k=200;
    console.log(k);
}
b();

function b2(){
    var d=200;
    console.log(d);
}
b2();
```



O/P

200
100
200
200



(8) Immediate Invoking Function Expression :- (IIFE)

(*) To execute the function, which doesn't have function name we will use Immediate Invoking function

(*) Why Immediate Invoking function Expression?

(*) because, ~~the doesn't polute~~ IIFF doesn't polute the global variables. means, whatever the variables, declared inside IIFE it total different or separate which we declared ~~the~~ outside the IIFE.

(*) USE of IIFE :-

(*) When we have set of codes, the current code is effecting by previous code, then we will use IIFE by invoking that code. (IIFE).

Ex:

[IIFE]

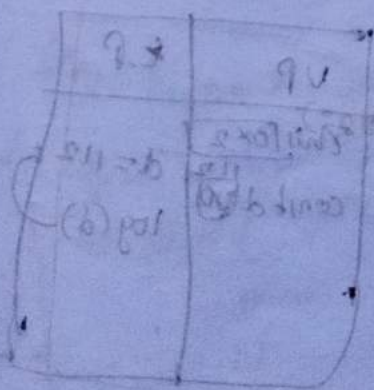
```
( function () {
    console.log("Hello");
}
())
```

Syntax of IIFE :-

```
(function()
{
  ---
  ---
})()
```

Example :

```
let a = 'UMA';
console.log(a);

(function()
{
  let a = 'saroja';
  console.log(a);
})();
```

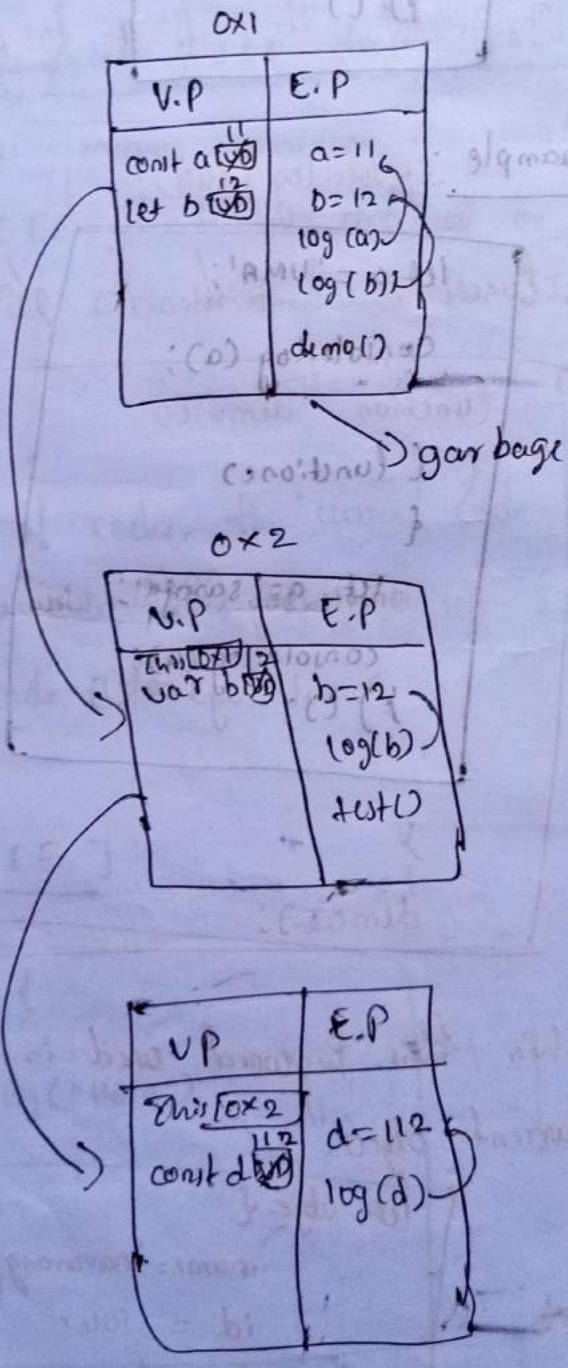


Function Execution Context :-

(*) Function Execution context is each & every functions how it works internally. That is called Function Execution Context.

Example :-

```
const a = 11;
let b = 12;
Console.log(a);
Console.log(b);
function demo(){
  var b = 12;
  console.log(b);
}
demo();
function test(){
  const d = 112;
  console.log(d);
}
test();
```



THIS. KEYWORD

23/1/2023

① This is a keyword ~~is~~ refers to an object, which belongs to keyword.

② it differ values depends on where it present.

③ In general, it points to the global object.

```
console.log (this);
```

④ in, functions : it points to global object.

```
function demo ()
{
    console.log ("the sum of two numbers", 2+2);
    console.log (this);
}
demo();
```

⑤ when this. keyword used in object : it points current object.

```
let obj {
    name: "navarang bar";
    id = 101;
    sum = function ()
    {
        console.log (3+7);
        console.log (this);
    }
}
obj.sum ();
```

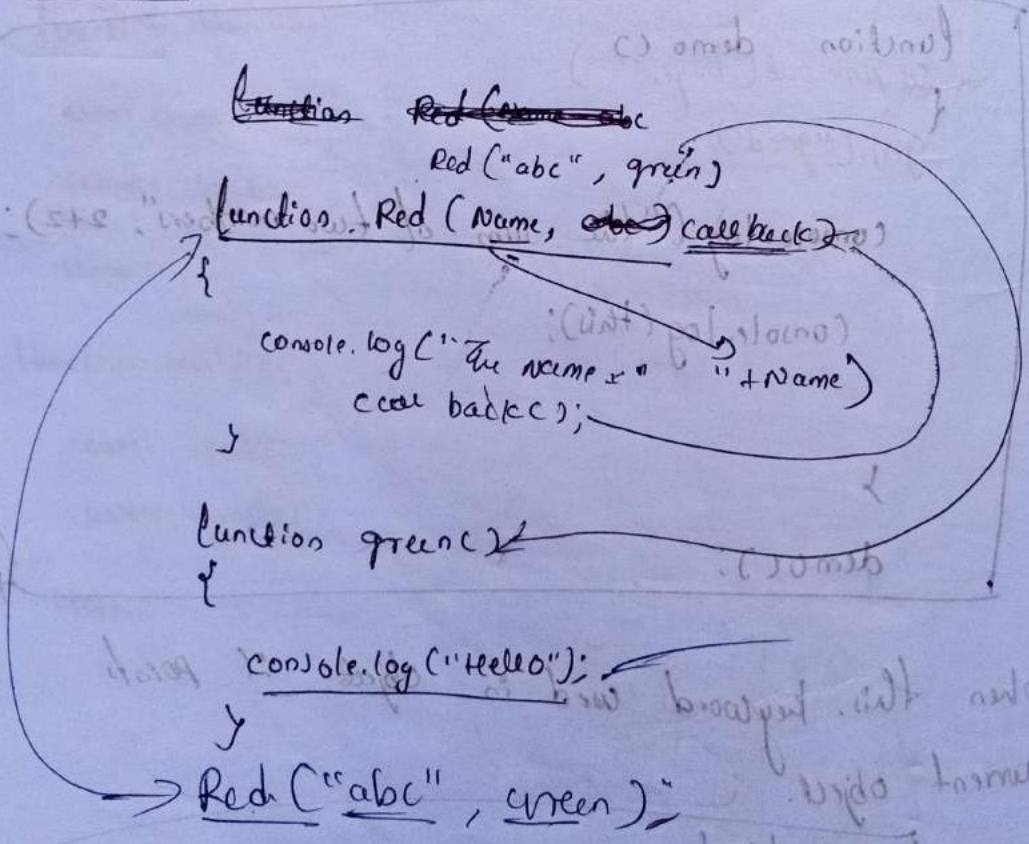
⑨ Callback Function :-

" A function passing as an argument to another function is called as callback function. "

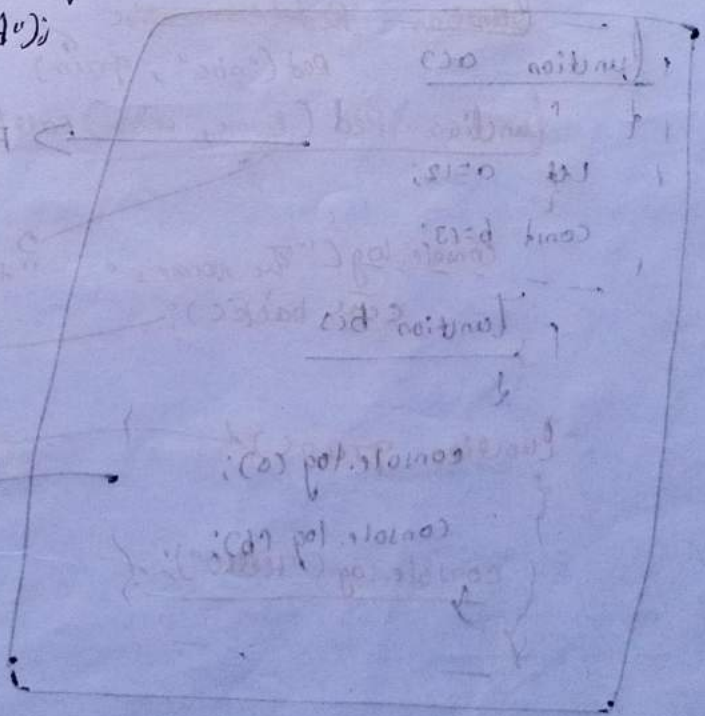
① callback function is used to build relationship b/w two functions.

② JS is basically synchronized in nature. We can make it Asynchronies by using callback function.

Example :-



```
function boys()
{

  console.log ("hi boys");

}

function girls(name)
{
  console.log ("girls are" + " " +name);
}

  setTimeout( boys, 5000)
  girls("good");
```



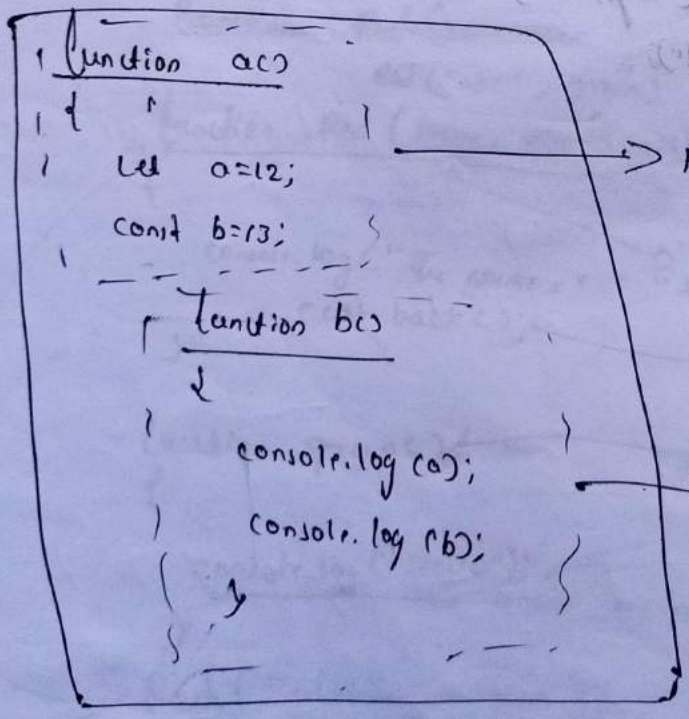
⑦ Nested function :-

① A function inside another function is called as Nested function.

② Ability of Javascript Engine to search for varaibles and outer ~~function is not a~~ scope or global scope is not availble in function is called as lexical scope.

③ We can achive lexical scope in Nested function.

Example :-



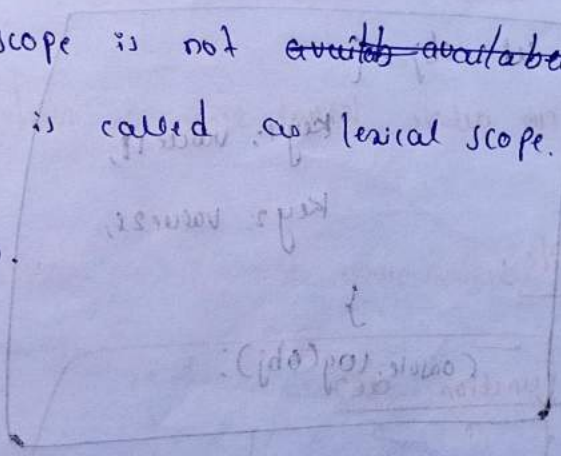
CLOSER :-

- holds the address of previous GEC.
- A closer is a function which is access to parent scope. even, the parent scope is deleted.

lexical scope :-

- ability of JS ~~it will~~ Engine, it will search variable over scope, global scope is not ~~available~~ available function is called as lexical scope.

Ex: nested function.



26/1/2023

OBJECT'S

① objects associates with properties.

② object is associates with properties and every properties consisting of key and value pair.

syntax of object :-

```
let obj = {
        key1: value1,
        key2: value2,
    }
console.log(obj);
```

③ Every properties separated with coma,

④ we can create object in two ways.

1] using literals.

2] using constructor new operator through.

1] using literals : By using literals through keywords.

```
let obj
{
    id : 104;
    name : "xyz".
}console.log (obj);
```

key → id : 104; → value

key → name : "xyz". → value

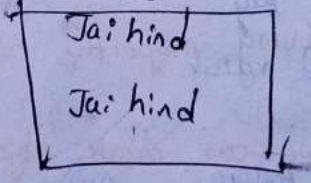
To fetch value of an object we have two ways.

① [ ] ⟶ Array notation.

② . ⟶ dot

```
let obj4
{
    id: 104,
    Num: "Jai hind",

}
console.log(obj4[Name]); //through array notation.
console.log(obj4.name); //through . dot.
```

o/p



Empty object :

```
let EObj = {

}
```

30/11/2023

OBJECT METHODS :-

1. Assign
2. Freeze
3. Entries
4. Key
5. value.

1] Assign :- Assign is a object method. it Targets the object value and update the object values.

Example : // assign

```
let a = {
        id: 110,
        name: "Xyz"
    }
let b = {
        id: 120,
        name: "abc"
    }
b.name = "pqr";
b.id = 840;
let c = object.assign(a,b);
Console.log(c);
```

— 2] Freeze object : Freeze object method stocks all the modified values then, after freezing the current object.

3] Entries : Entries object method converts object into arrays.

a] Keys : if we want fetch only keys in an object we will go for key object.

b] values : if we want fetch only values in an object we will go for obj values.

Object Types :-

1] MATH OBJECT
2] DATE OBJECT → Date objects is type of object : I show Date
3] TIME OBJECT → Time OBJECT it Display Time. getTime(), getHours

```
let dt = new Date();
console.log(dt, type of dt);  console.log(dt, getTime
"  (dt, getDay());  "  (dt, getHour
```

1] MATH OBJECT :- MATH OBJECT is Type of OBJECT.
it finds correct values of maths.

```
let a = 9.9;
console.log(Math.ceil(a));
```
o/p 10.

```
let b = 99.9
console.log(Math.floor(b));
```
o/p 99.-

Max & min Number.

```
let num1 = 100;
let num2 = 200;
console.log(Math.max(num1, num2));
console.log(Math.min(num1, num2));
console.log(Math.pow(2,3));
console.log(Math.round(num1));
```

Array Methods :-

1] Array.isArray() :-

(*) if you want to check a whether given variable is array / not.

(*) The output in the form boolean. (True / False).

Example

False.

```
let a={
    id:101
}
console.log(Array.isArray(a));
```

True

```
let array1=[100, 200, 300];
console.log(Array.isArray(array1));
```

1] PUSH()

2] UNSHIFT()

3] POP()

4] SHIFT()

1] PUSH() :- push() method adding the elements in array from behind / last position of array.

(*) push method affects the original length of the array.

(*) we can pass more than one values in a push method.

ex.

```
let parray = [100, 200, 300];
parray.push("NAGU");
console.log(parray);
```

2) UNSHIFT :- unshift method adding the elements in array starting from 0th array.

① unshift method it will effects the original length of array.

② we can pass more than one values unshift().

Ex:

```
let usarray = [500, 100, 400, "Hello"];
usarray.unshift(1, "Nagu");
console.log(usarray);
```

O/P

[1, "Nagu", 500, 100, 400, "Hello"]

3) pop (): pop method removes the elements from the behind of an array.

① it will effect the original length of array.

② we can't pass the values in pop() method.

③ It removes values only one at a time.

Ex:

```
let plarray = [150, 200, "web"];
PlArray.pop()
console.log(PlArray);
```

O/P

[150 200]

4) shift() :- shift method it will remove the element from ~~the~~ front of an array.

(1) it will effect length of array.

(2) we can't pass values for the shift():

Ex:

```
let sArray = ["gomu", "pomu", "somu"];
sArray.shift();
console.log(sArray);
```

SLICE METHOD :-

(1) slice method except two values.

(2) it includes first values, it excludes last values.

(3) it will not effect original length of array.

ex:

```
let slArray = [1, 2, 3, 4, 5];
console.log(slArray.slice(0, 4));
console.log(slArray);
```

0 1 2 3 4

o/p

1 2 3 4

SPLICE METHOD :

① Splice method excepts Three values.

② First value indicates Index of an array.

③ second value indicates. how many elements want to remove from an array.

④ Third value indicates elements add to an array.

⑤ it will effects original length of array.

Example :

```
let spliArray = [1, 2, 3, 4];
spliArray.splice(2, 2, "Nagu");
console.log(spliArray);
```

O/P

[1 2 "Nagu"]

JOIN METHOD :-

it converts array to string

Example :

```
let jArray = [20, 30, 40, 50]
console.log(jArray.join("+"))
```

[20+30+40+50]

Concate an Array ()

```
let ar= [11, 21, 31, 41];
let ar1= ["a", "b", "c", "d"];
let ar2= ["A", "B", "!", "A"];
Console.log (ar.concat(ar1.ar2))
```

3/2/2023

MAP, FILTER, REDUCE :-

"It is a method which changes entire elements of an Array with every element that passes the test implemented by provided by the function."

i) FILTER :-

(a) FILTER is the method. filter() method is use to filter the Arrays.

(1)

```
let a= [100, 200, 300, 400, 500];
let b = a.filter ( n=>{
                    return n> 200;
                  });
console.log(b);
```

O/P

```
300
400
500
```

(b) n is a value that starting count first element of Array.

(c) n :+ , n> 200 :+ comparing all elements from Array.

(2)

```
var a= [100, 200, 300, 400, 500];
var b= a.filter ( n=>{
                    return n< 300;
                  });
console.log(b);
```

(3)

```
let arr1 = [10, 20, 30, 40, 50];
let arr2 = arr1.filter ( v=>{
                    return v>20;
                  });
console.log (arr2);
```

O/P

```
30
40
50
```

2] REDUCE :-

① REDUCE :- a method. it changes entire method into single value.

② Reduce accepts two arguments accumulator, & value.

1] accumulator :- Pass as an Argument to the reduce function.

2] values :- it take every elements of an Array & perform the operation.

Example :-

reduce() :-

①

```
let red = [50, 60, 70, 80];
let red1 = reduce ((accu, value) => {
                    return acc+val;
                });
console.log(red1);
```

O/P

260

②

```
let reduce1 = [10, 20, 30];
let reduce2 = reduce1.reduce((accu, value) => {
                    return accu + value;
                });
console.log(reduce2)
```

O/P

60

3] MAP():-

(A) MAP() is a method that changes entire elements of array by adding specific number for every elements of an array.

(B) It add every elements of an array.

Example:-

(1)

```
let n = [10, 20, 30, 40];
let m = n.map( p => {
        return p + 20;
    });
console.log(m);
```

O/P

~~10~~
30
40
50
60

(2)

```
let arr = [10, 20, 30, 40, 50];
let arr_1 = arr.map( c => {
        return p - 10;
    });
console.log(arr_1);
```

O/P

0
10
20
30
40.

6/2/23

BOM (Browser object Model) :-

(1) BOM : It stands for Browser object Model. It is an object which has ability interact between JS (Javascript) and the browser.

(2) BOM inside we will see window object.

1] Screen → console.log(window) → screen.
2] Navigator
3] History
4] ~~Document~~ Document
5] Location.

(3) above, all properties under comes window object.

(4) All the global variables, global object, method and properties belongs to the member of window object.

1] Screen :- The screen object contains the infrastructure about the screen.

```
window.screen()
```

2] Navigator :- It contain Infrastructure about the visitor screen.

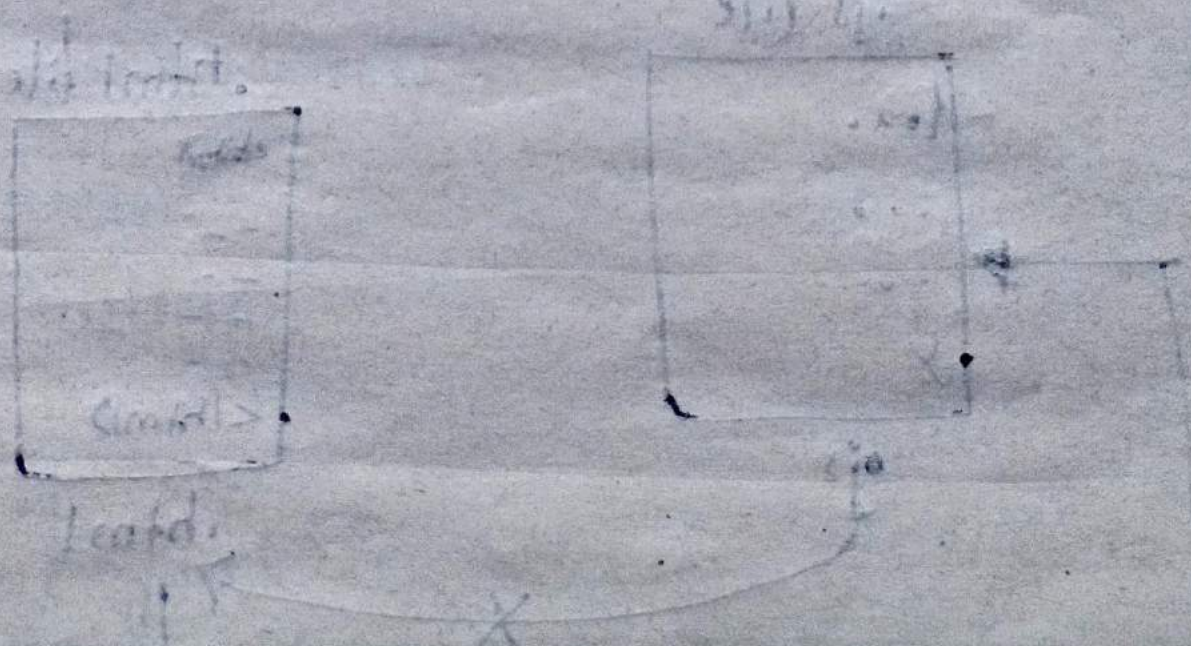
```
window.navigator.cookieEnabled;
```
op / True

3] History :- History is an object which contain URL visited by the user

```
window.history.back();
window.history.forward();
```
← previous History
→ front page

5] Location :- location is an object which contains information about the current url.

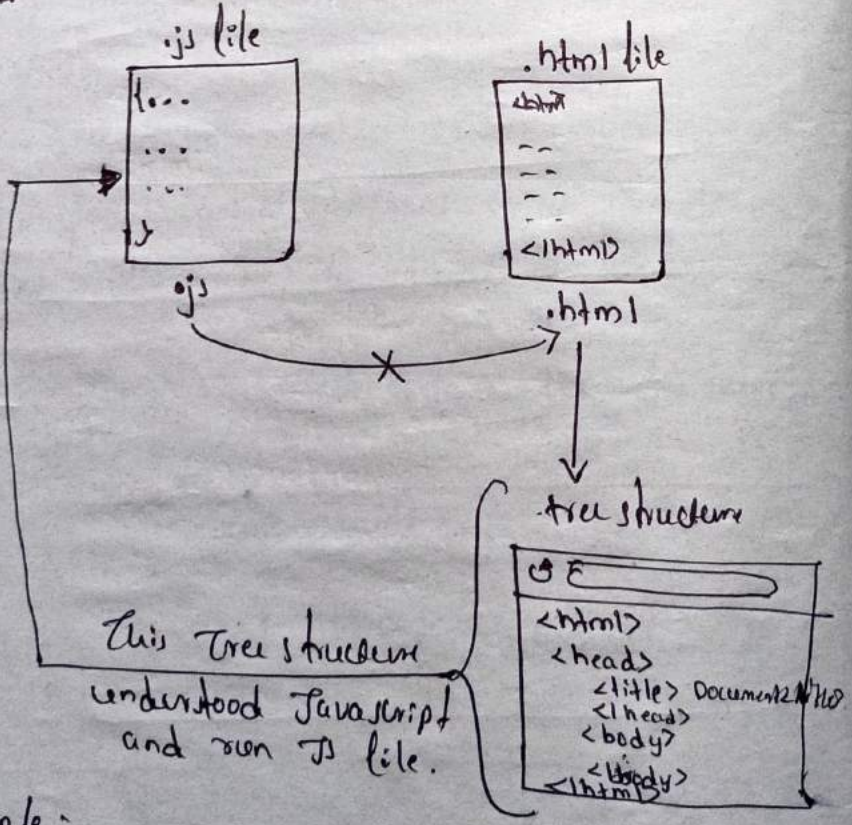
```
window.location.href = "http://www.instagram.com";
```



DOM (Document Object Model)

- DOM : it stands for Document Object Model. it an object. : t presents in window section.
- DOM is one of the Javascript concept.
- Javascript (JS) doesn't understood HTML like
- DOM is in the form of Tree structure.
- Browser will create DOM.

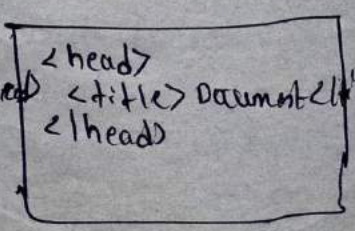
DOM:



Example :

```
let n= document.head;
```

:t Display <head> Tag to close of </head> in the form of document.



Document Tag:

① document.head;

② document.title;

③ document.body;

④ document;

DOM METHOD'S

1] getElementById();

2] getElementByClassName();

3] getElementByTagName();

4] querySelector

5] querySelectorAll

1] getElementById():-

① getElementById is one of the DOM method.

② getElementById which Targets a particular element, that elements Targeted by Id.

example:

.html file

```
<html>
<head>
    <title> sample </title>
</head>
<body>
<h1 id='test'> Hello </h1>

</body>
</html>
```

.js file

```
let t= document.getElementById
        Id ('test');
t.style.backgroundColor="blue";
t.style.color="orange";
t.style.textAlign="center";
console.log (t);
```

③ .innerHTML you can change text.

2) getElementByClassName :-

① getElementByClassName is Dom method. it's Target the className inside class all the elements can make style it through getElementByClassName.

② it's Target class name & make it style.

.html file

```
<html>
<body>
<p class="demo">
Hi, This is getElemntByClassName one
of the DOM method
</p>
</body>
</html>
```

.js file

```
let b= document.getElementByClassName('demo');
b[0].style.backgroundColor='orange';
b[0].style.color="black";
b[0].style.TextAlign="center";
console.log(b);
```

3] getElementByTagName() :-

① getElementByTagName is one of the DOM method.

② getElementByTagName which is used to targets by taking html elements tag name, it returns the answers in html collection.

.html file

```
<html>
<body>
<h1 id="tt"> Hello Targets
</h1>

</body>
</html>
```

.js file

```
var tar = document.getElementByTagName
          ('tt');

console.log(tar);
```

4] query selector :-

(*) query selector is DOM method. which is used to
:1 Targets the html first element and make
:1 Elements style.

.html file

```
<html>
<body>
<h1> Hello... </h1>


</body>
</html>
```

.js file

```
var se = document.querySelector
console.log (se);
```

5) query selector All :-

(*) query selector All is a DOM method, which is used to it Targets all HTML elements and make elements style it.

.html file

```
<html>
<body>
<h1> Hello.. </h1>
<p> Hi This is paragraph </p>
<h5> Hello This hs </h5>

</body>
</html>
```

.js file

```
var all = document.querySelectorAll

console.log(all);
```

document.getElementsByName:

(*) document.getElementsByName is Dom method it Targets the name in the form Tag input method & <input type="text", name="firstname"> and make it style.

.html file.

```
<html>
<body>
<form>
<input type="text", name="firstname">
<input type "text" name="lastname">

</form>
</body>
</html>
```

.js file

```
var f_name=document.getElementsByName
('firstname');
f_name.style.backgroundColor="Red";
f_name.style.color="black";
console.log(f_name);
```

Create Elements method :-

1) Create Elements is a method it is used to Creating html elements.

2) append add the values in Html body.

.js file

①

```
var head = document.createElement('h2');
document.body.append(head);
head.innerHTML = "Headding Tag2";
console.log(head);
```

.js file

```
var p = document.createElement('p');
document.body.append(p);
p.innerHTML = "Hi This is paragraph";
console.log(p);
```

DOM EVENTS

1] Onclick Event.

2] prevent default event.

3] key Events ⟶ ① key down ② key up ③ keypress.

1] onclick Event :-

① Onclick event is a one of the Dom Event.

② Onclick it's job is when we click on button it will changes the entire web page. style, image, background...

① .html file

```
<html>
<body>
<h1> onclick Event </h1>
<button onclick="sc()"> submit </button>

</body>
</html>
```

.js file

```
var v= document.querySelector('button')
v. onclick=()=>{
document. body. style. background="red";
}
```

Examples

.html file

```
<html>
<body>
<h1> onclick event </h1>
<div>
<button onclick="demo()"> submit </button>
</div>

</body>
</html>
```

.js file

```
var s = document.queryselector('button');
s.addEventListner('click',()=>{
var st = document.queryselector('div');
st.innerHTML = "Hello world...!";
st.style.color="red";
st.style.height="400px";
st.style.width="500px";
st.style.transition="5s";
st.style.fontsize="100px";

})
```

2] Prevent default Event :-

① Prevent default Event is one of the DOM event.

② Prevent default Event is used to fetch the username & password Data & it's fetch the login details and display it.

③ Prevent default it fetches the Data through. method

```
document.getElementById().value;
```

.value

Example :-

.html file

```
<html>
<body>
<form>
<label> Username:
<input type="text" name="Uname" Id="un">
</label>
<label> password
<input type="text" name="Password" Id="pw">
</label>
<button> Submit </button>
</form>
</body>
</html>
```

.js file.

```
var data = document.getElementById query selector ('form')
data.addEventListner ('submit', (e) => {
let uname = document.getElementById ('un').value;
let password = document.getElementById ('pw').value;
e.preventDefault();
console.log (uname, password);
})
```

3] Key Event :-

1] Key down
2] Key up
3] Keypress.

1] Key down :- key down is one of the key event, it is calculating the number of keys user entering from keyboard.

.html file

```
Enter Text
<input type="text" name="name" id="Enter">
```

.js file

```
let text1 = document.getElementById('enter');
text1.addEventListener('Keydown', ()=>{
    console.log("keydown");
})
```

2] keyup :- keyup is a keyEvent. keyup is it's calculating the values from keyboard. when we press the key it starts counting the key value.

.html file

```
Enter your Text :
<input type="text" name="name" id="enter">
```

.js file

```
let key = document.getElementById('enter');
key.addEventListner ('keyup', () => {

    console.log ("keyup");

})
```

3] keypress :- keypress is the one of the keyevent it work same as keydown.

.html file

```
Enter your Text :
<input type="text" name="name"
 id="enter">
```

.js file

```
let keyp = document.getElementById('enter')
keyp.addEventListner('keypress', () => {
    console.log("key press");
})
```

if you want background color different colors.

```
let radcol = Math.floor(math.random() * 1000000).to
                                        String(16);
console.log(radcol);
document.body.style.backgroundColor = '#${radcol}'
});
```

mouseOver() :-

(R) mouse over method is a whenever we move on curs to the text field / button. It will changes style.

.html file

```
<button> submit </button>
```

.js file

```
let m= document.querySelector('button');
m.addEventListner ('mouseover', ()=>{
m.style.height ="100px";
 m.style.width ="200px";
 m.style.color ="red";
});
```

16/2/23

-: STORAGE :-

1) " Storage is a process of collecting data and store to storage devices".

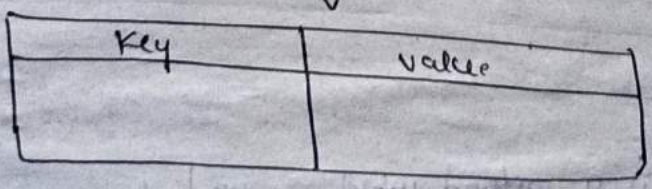
Storage has two storage types :-

1] Local storage
2] session storage

1] Local storage :-

(1) Local storage is type of storage. through this storage we can store the Data in the Browser.

console > —> application



(2) This storage capacity is 5MB.

(3) Local storage is store data where the Data is valid until wer can delete that.

(4) The local storage is store data permenaly.

store data ~ some have methods :-

1] setItem() :
2] getItem() :
3] removeItem() :
4] clear() ;

1] setItem() :-

(R) setItem is a method which is wed to store data in both storage.

ex:

key — value

var data = Localstorage.setItem ("Id", 1);
var data1 = Local storage.setItem ("Name", "Nagu");

2] getItem() :-

(R) getItem() is a method which is wed to fetching the ~~data~~ data we we getItem().

var data = localstorage.getItem (1);
console.log (data);

3] removeItem() :-

(R) removeItem() is a method which is wed to remove the data.

var del = local storage.removeItem (1);
console.log (del);

4] clear() :-

Clear() method is a it removed all data from storage.

var clear = local storage.clear();
Console.log (clear);

2] Session storage :-

(1) Session storage is a storage type where the data is valid until the session expires.

(2) We can store the data through the called setItem().

(3) In the storage Local/session storage the data is store in terms of keys and values.

(4) We can fetch the value through the method called get method.

(5) We can remove the particular pair (key value) through the method called removeItem.

(6) If you want clear all the data in storage you can clear through clear method.

(1) store data

```
var s = sessionStorage.setItem("Id", 12);
```

(2) get data

```
var data= sessionstorag.getItem(12);
console.log(data);
```

(3) remove data

```
var data1=sessionStorage.removeItem(12);
console.log(data1);
```

(4) clear data

```
var data2 = sessionstorage.clear();
```

-:o: PROMISE :o:-

(1) Promise is one of the Javascript object.

(2) Promise is a process of linking b/w produce code and consume code.

(3) Produce code and consume code this are two type of promise.

1] Produce code :- Produce code is type of promise produce code it takes some time to execute the code.

2] Consume code :- Consume code is type of promise consume code it wait for the Result.

(4) Promise checks the Asynchronous operations is successfully completed or not.

Synchronous :- it is process step by step run.

Asynchronous :- it is also process it not step by step run it execute at time.

Promise has Three states :-

1] Pending
2] Rejected
3] fullfield.

Syntax of promise :-

```
let promise = new Promise (function (resolve, reject) {
    ____;
    ____;
    ____;
    ____;
})
```

(1) promise construction accept a function of an arguments, that function accept two function, that is resolve and reject.

(2) resolve :- resolve function when promise is return successfully then resolve function will call.

(3) reject :- reject function, when promise is fails then reject function will call.

24/2/2023

JSON

(1) JSON is one of the Javascript concept.

(2) JSON it stands for Javascript object Notation

(3) JSON is used to standard text based format for representing structured data based on Javascript object syntax.

(4) JSON is perfect for storing temporary data.
for example: temporary data can be user generated data such as submitted form on web site.

Data format:

object can storing the data formed.

Key : Value

Syntax:

Javascript

```
let object {
    id: 11,
    name: "Nagu"
}
object();
```

JSON The key & value.

```
{
  "Id": 104,
  "Name": "Abc",
  "place": "Bangalore"
}
```

JavaScript object Notation have two methods :-

1] stringyly ();
2] Parse ();

1] stringyly() :-

(*) stringyly() method is wed to, when the client is sends the Request to server, The server will accept the Request & process the Request The data form will be in within double quoat both key & values.

2] Parse() :-

(*) Parse(): method is wed to, when the server inside data fetching by client that data form is object data format. That is parse() method.

Example of stringyly method :-

① stringyly() :-

```
let a = {
    Id: 11,
    name: "nagu"
  }
console.log(a); //object
let str = JSON.stringyly(a);
console.log(str);
```

Example of parse() method

② Parse () :-

① same object return back to the Data by using parse ():

```
let a = {
    Id: 11;
    nam: "nagu".
  }
  Console.log(a); //object.
let str = JSON.parse(str);
Console.log(str);
```