JavaScript

Variable :- which store value in it like container. It always remember value and we can use calculation easily.

Data - Types in JS

Primitive Types

- Number
- Boolean
- String
- undefined
- Null
- Bigint
- Symbol

Type of :- This is a method which tells the type of data.

- 1 Numbers :- Number is the data type in script.
  - Positive (14) & negative (-14)
  - Integer (45, -50)
  - floating numbers - with decimal (4.6, -8.9)

limits in variables :- > There is limit to number, and then it will keep the nearest int round of value.

* Operations in JavaScript :->

a = 10
b = 20

- Modulo (remainder operator)
  12 % 5 = 2
- Exponention (Power Operator)

7. Modulo :- (%) works as detection odd and even number

-> odd number modula allways gives some value and give reminder

n%.2 == (n)

-> even number modulo always give 0.

12%.2 = 0

(12) is even number.

Exponentiation 2**3 -> 2^3 => (8)

a**b => a^b

Next Project (after this class) we can make calculation using java script.

NaN in (JS) (Not a Number)

0/0 -> NaN

type of NAN -> 'Number'

NaN + 1 -> NaN

Nan - 1 -> NaN

NaN * 1 -> NaN

NaN ÷ 1 -> NaN

NAN % 1 -> NaN

Operator Precedence :->

General order of Solving an expression.

()

**

*, /, %.

BODMAS

let keyword :-

Syntax of declaring variables

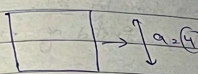
```
let age = 23;
age = age+1;
```

→ Proper Syntax:-

```
let cgPa;
cgpa = 8.9;
```

```
let num1 = 1;
let num2 = 2;
let num3 = 3;
let Sum = num1 + num2 + num3;
```

let → declare value of variable just a syntax.



Calculate area

```
let a = 4;
area = a**2
```

→ 4x4 = 16

let → we declare at the very initial position

↓

if we need to modify the value then just change it no need to declare again using let

```
let num1 = 1;
let num2 = 2;
let Sum;
Sum = num1 + num2;
```

Sum = 3;

(X) Const → Constant → not able to change

```
Const newnum = 5;
newnum+1; → give error not can be change due to (Const)
```

Const keyword → value of constants can't be changed with re-assignment if they can't be re-declared

```
const yeqr = 2025;

year = 2026 // Error :>
year => year ++ // Error -> Const cannot change its value as it is already declare as constant

Const P1 = 3.14;
```

$\pi = 3.14$ or $22/7$

Var keyword :> old style using (old syntax of writing variable)

work done by var was before now done by let.
So we majorly use let nowadays instead of (var)

Comments in Script

```
let sum = 24; // this is the comment of same of type
```

Unary operator :> Single operator

```
age = age+1 ;
age += 1;
age -= 1;
age *= 1;
```

age ++ // increment age -- // decrement
++age // same --age // same.

Pre increment (Change, then use)

```
let age = 10;
let newage = ++age;
```

Post increment. (Use, then change)

```
let age=10;
let newAge = age++;
```

Q

```
let num = 5;          →  (5)
let newNum = num++;   →  (6)
New Num = ++num;      →  (7)
```

Identifiers Rule:-

- all script variable must be identified with unique name
  A-Z 0-9 $ _ #
- Hyphen (-) is Not allowed else error.
- No space should be there. max price #error 404
- Name must begin with letter.
- Name can also begin with $, _ valid (_age, $age)
- Name not begin with number.
- Names are case sensitive. (Pus P) (a vs A)
- Reserved keyword should not be variable name.

Should be meaning ful name there which describe identity.

* Camel case :- way of writing identifiers.

Camelcase
Snake case
Pascal case

→ Just like write fn name in java
myfunction

↑
also Camel Case

let firstName

↑ small ↓ Capital

Boolean in JS: It represent the truth value. (T/F)

```
let age = 13;
is this person is adult just cheack.
let isAdult = false; // person is not adult.

let age = 25;
let isAdult = true;
```

typeof isAdult => 'Boolean'

true and false comes with small letter.

we can change the variable type inside script

Java Does not allowed that where as script does.

```
let a = 5;
typeof a = 'number'
a = true;
type of a => 'boolean'
```

What is typeScript ?

Static typed, where JS is dynamic typed.

Designed by Microsoft

fixed ↙ ↘ Change

String in JS :-

Strings are text or sequence of character.

```
let name = " Tony Stark";
let role = "IRONMAN";
let char = "a";
let name = "kuf$a";
let empty = "";
```

```
let Sentence = "this is 'apple'";
Sentence →
"this is 'apple'"
```

Sigle & double comes with double and double with sigle both cannot comes at same time it give error.

String Indexing :- let name = "Tony Stark";

name → "Tony Stark"

T	O	N	Y		S	T	A	R	K
0	1	2	3	4	5	6	7	8	9

name[0] → 'T'

name[1] → 'O'

name[4] → ' '

Index start from 0

⇒ also known as zero based indexing

Java script follows same rule of Java followed by 0-1 indexing same as array concept.

we can access the element via indexing.

name[0] → 'T'

name[5] → 'S' so-on

name.length → 10

↓

Property of size

Null & undefined:-

undefined :- A variable that has not been assigned a value is of type undefined.

```
let a;
undefined
```

(No initialize)

Null :- The null value represent the intentional absence of any object value.

To be explicitly assigned.

```
let a = null;
undefined;
> a
< null
```

> let name = "Shaunaat";
< undefined
> type of name
< string
> name.length
< 8

String + Number → String

"kush" + 123 → "kush123"

* Space always got counted in script both they are differ Empty vs Space

"" (vs) " "

→ length → 0 length → 1

Javascript → class → 2

* Console window

1. console.log() :- To write(log) a message on the console

```
Console.log("Hii");
```

* Linking js file:

```
<script src = "app.js"></script>
```

Script add functionality → Before action our HTML icon should be loadded that's why it should always include before </body> end tag.

In console we can avoid (;) where as in (JS) file we cannot (;) ⇐ nessesary for end line.

* Template literals :- They are used to add embeded expression in a string.

```
let a = 5;
let b = 10;
Console.log(`your pay ${a+b} rupee`);
```

* Comparison operator

- Compare value, not type

  ==

  "123" == 123

  → true (by value)

vs

- Compare both type & val

  ===

  "123" === 123

  → false

IV vs 'iv'
true

0 === '0'
false

Strict comparison

> 0 == ' '
< true
> 0 == ' ' false
< true

0 === false
< false

* Comparison for non-number

> 'a' > 'A'
< 'true'

Search javascript unicode.

→ 'p' < 'b' according to unicode
50 < 70
true

cheack online available Pdf.

unicode.org/charts/pdf/U0000.pdf

So to link

a < b < c < d - - - - z

A < B < C < D < - - - - Z

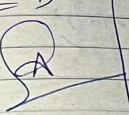
Conditional Statement

if else

if statement!-

```
console.log("first line get execute lets start if-else");
let age = 18;
if (age >= 18){
    console.log("you can vote");
    console.log("you can drive");
}
console.log("line get ended");
```

Any →



first line get executed lets start if-else ...
you can vote.
you can drive;
line get ended.

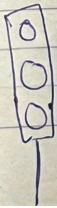
① what if age <= 13

you are

first and same.

line get ended

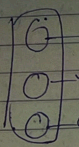
② what functionality does a traffic light do.



Red → stop
Yell → Be ready
Green → lets go

```
let color1 = "Red";
let color2 = "Yellow";
let color3 = "Green";
```

→ String type data.




```
Console.log("lets run game");
if (color1 == Red) {
    Console.log("You should stop");
} if (color2 == Yellow) {
    Console.log("You should give Yield")
} if (color3 == Green) {
    Console.log("You have way to go");
} Console.log("game end");
```

String Methods :-

Methods → action the things to perform action.

format

```
StringName.method()
```

```
let str = "kush";
```

String Method.

Str.trim()

→

Wide spaces will be remove by this fn and will return new str.

Array in Java

Javascript Array vs

JavaScript Array vs Arrays in Java

Feature	JavaScript Array	Java array	Support	Wy/How
Type System	Dynamicly, can hold all typed inside arr	Statically typed cannot hold all. one at time	X in Java	int[] → cannot add str.
length	Dynamic can grow/shrink	fixed after creation	X in Java	Java fix size
Indexing	0-based	0-based	both	Doly
Mutability	Mutable	Mutable	both	element can be change after creation
Type Safety	No	yes	* in Java ✓ x in JS X	flexible -
Performance	less than	Java array are faster and memory efficient	✓	✓

Arrays are Mutable.

let msg = "Hello";

Shift and unshift Concept:

* Shift() → remove the first element and shift the rest of element
* unshift() → add a new element at the start

let arr = ["A", "C", "F"];

↓ to

["B", "C", "F"]

shift(arr);

console.log(arr) → ["C", "F"]

unshift(arr) → ("B") Must Pass value

arrays.unshift(element1, element2, element3, ...n);

arr.unshift("B");

Console.log(arr);

A B C → B C F

Remove A → shift()

Add F to end → then add or Runshift

arr.unshift("B");

Array Method

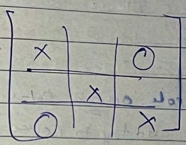
index of :- return the index of the element which is present in array container, must have same name, same camell letter due to sensetive language (JS)

```
array.indexOf("object");
```

Array Const :-

We can perform operation but we can not transform completely this array due to const keyword.

Tic Tac Game :-



Black space → represent (null) value.

```
let arr = [ ['X', null, 'O'], [ null, 'X', null],
            ['O', null, 'X'] ];
```

Loop → To Run thing more than moments
same thing like repetitions.

use syntax.

same as for loop

```
for (let i=0; i<n; i++) {
    console.log(i);
}
```

Print Table of (5)

5 X 1 => 5

5 X 2 => 10

5 X 10 => 50

```
for(let i=1; i<10; i++)
```

11-12 DP attempt one code on bjb Solve (3) questions
look at back notes and try to cover again, with DFS

DOM Events

Q what are Dom Event in JS?

Dom Events are the signal sent when something is happens in the browser - like a user clicking, typing or hovering.

Event	→	when it happen.
Click	→	User click on a element
mouseover	→	Mouse goes over an element.
mouseout	→	Mouse leaves an element.
keydown	→	key is pressed.
keyup	→	key is released.
submit	→	form is submitted
change	→	input value is changed (select, text)
load	→	Page finished loading.

DOM → event as a security guard read.

How → it checked at a door like a security to pass some basic information.

↳ Dom concept — Real world Analogy.

element . add Event Listner() — assign a guard to safety door

Click, Submit → kind of behaviors guard is working on (display a bag, swiping a card)

Call back function — The guard reaction to the event (sound alarm, let person in, log event)

event.preventDefault() — The guard block someone from doing something (like stoping from reloading page)

→ Achieve what to want need to better health & wealth spiritually.

$E=mc^2$

Here we are going to create a button and once we clicked on that button i want to change the background color of it.

Jsy →

```
let btn = document.querySelector("button");
let p = document.querySelector("p");
let h1 = document.querySelector("h1");
let h2 = document.querySelector("h2");
```

→ add new event listner

```
btn.addEventListner("click", function() {
    Console.log("it got clicked");
    btn.~~document~~.style.
    background.color = "blue";
});
```

for this thing i will say ; have to create the same function and just add
• notation in front of it . instead . using this
great concept from here.

```
function Change color (){
    console . dir ( this . inner text) ;
        this. style . background color = "Blue";
        }
```

→ (call the f^n here)

btn. add event listner ("click", change color);
p. add event listner ("click", change color

Yes we can use it in the linear form,

document. query Selector ("button"). add eventListner ("click",
Changecolor);

// where Change color f^n, define color,

```
function change color (){
    console. let red = math.floor(Math.random()* 255);
        let green = "
        let yellow "
        blue = "
}
```

0.57 — remove dec — val in b/w

i need to bring a fn for Color which have some fix valuy as a string So.

let random color = `rgba (${red}, ${green}, ${blue}`);

Keyword functions in java script

```
let inp = document.querySelector("input");
inp.addEventListener("keydown", function() {
    consok.log("key was pressed");
});
```

Keyboard Event Constructor

Syntax :-

new KeyboardEvent(type)

new KeyboardEvent(type, option)

type :- A string with name of event is case sensitive and browser set it to keydown, keyup or keypress

Options :- An object that, in addition of the property defines in UIEvent(), can have following Properties

key (optional)

A string, defaulting to "", that set to the value of keyboardEvent.key

→ 3-July-2025

Today mission is to owner manual hidden access system.

→ This is the system where secret key combo works.

- View booked appointments.
- View conversations.
- Stay hidden from normal users.

Phase 1 → Define the structure and flow

Step 1 → Owner visit normal website.
step 2 → Press Ctrl + Shift + O
step 3 → localStorage stored is owner = true.
Step 4 → hidden dashboard is shown.

Logic

if owner presses a secret combo, and the system knows which location they are on (Main street or olivia), then only show that location's appointment and messages.

Logic

```
const System_Location = "Main St";
```

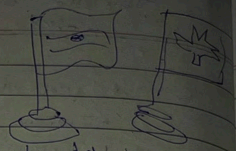
Plan → filter appointments by locations.

logic document.addeventlistner ("keydown", function (event){

```
if ( event.ctrlKey && event.shiftKey && event.key == 'O')
    { // logic to show admin section }
});
```

Call Stack LIFO

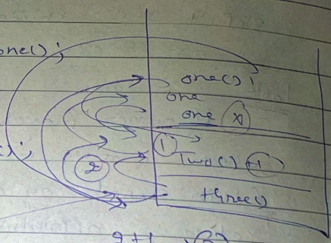
Visualize the call stack



Breakpoints :- it works when d. debugging - finding error -> we will see this concept in Backend. during backend.

```
function one(){
    return 1;
}
function two(){
    return one() + one();
}
function three(){
    return two() + one();
}
three(); // call stack
```

Breakpoint here



Call stack is empty now.

JS is single threaded language.

it execute one piece of code at one time. on a single call stack and in a specific order.

what is an API. Application Programing Interface.

In a web development, an API is usually a URL endpoint. endPoint you can send a

Chrome write its code in C++.
So Browser is getting out work for its own and its multithread language.

Asynchronise Nature?

→ Things can happen in the background, and we don't have the wait for them.

→ Instead of waiting for a slow task (like an API call) JS continues execution after code once that slow task finish, a callback get triggered.

Q why is JS Asynchronous?

→ Because JS is single-threaded, it can only do one thing at a time.

→ callback Hell →
Concept → code → solution

→ it is when we have many nested callbacks, one inside other → making the code.
! Hard to read, hard to maintain, Error-Prone.

How async, await and Promises helps to reduce ~~error or~~ Hell callbacks.

* Promises:- The Promises Object represents the eventual completion (or failure) of an asynchronous operation and its resulting value.