JavaScript

Variable :- Switch store value in it the computer 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 (19) & negative (-19)
  - 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)

* Modulo :- works as detection odd and even number

-> add number modulo always gives remainder and give remainder

$$n\%2 = 0 \text{ or } 1$$

-> even number modulo always give 0

$$12\%2 = 0$$

(12) is even number.

Exponentiation $2**3$ or 2^3 => (8)

next Project (after this class) $a**b => a^b$

We can make calculation using javascript

NaN in (JS) (Not a Number)

$0/0 \rightarrow$ 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 = 21 ; → Proper Syntax

age = age+1 ;

let CgPa ;

cgpa = 6.7 ;

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 * a

⇒ 4 x 4 = 16

let should declare it 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 ;

let 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 & they can't be re-declared)

const year = 2018;

year = 2026 // Error

year = year + 1 // Error -> Const cannot change its value as it is already declared as Constant

const PI = 3.14; PI = 3.14 or 22/7

Var keyword :- old style using (old syntax of work 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 as same as JS

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 = 20;

let newAge = age++;

Q let num = 5; → (5)

let newNum = num++; → (6)

NewNum = ++num; → (7)

IDentifiers Rule:-

- all script variable must be identified with unique name

  A-Z 0-9 $ #?

- Hyphen (-) is Not allowed give error.
- No space should be there. max Price #error yo
- Name must begin with letter.
- Name can also begin with $, _ valid(_age, $age)
- Name not begin with number.
- Names are case sensitive. (P vs p) (a vs A)
- Reserved keyword should not be variable name.
- Should be meaningful name then which describe identity.

* Camel case :- way of writing identifiers.

Camelcase → just like write fn name in Java myFunction

Snake case

Pascal Case

↑ as/er Camel Case

let firstName
↑ ↓
small Latin

Boolean type: a Represent the true value. (T/F)

let age = 15;
// Here Person is adult or not check.
let isAdult = false; // Person is not adult.

let age = 25;
let isAdult = true;

typeof isAdult is 'Boolean'

true and false comes with small letter.

We can change the variable type inside script.
Java Does not allowed that whereas JS/TS does.

let a = 5;
typeof a = 'number'
a = true;
typeof a is 'boolean'

What is TypeScript ?
Static typed, where JS is dynamic typed.
Designed by Microsoft
fixed
change

String in JS :-

Strings are basically sequence of characters.

```
let name = "Tony Stark";
let role = "Avengers";
let char = "a";
let name = "kutta";
let empty = "";
```

let Sentence = "this is 'apple'";

Sentence →

"this is 'apple'"

Single gote comes with double and double with single both cannot comes at same time in one string.

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] → ' ' → also known as zero based indexing

Index start from 0

JavaScript 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
```

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

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

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

String + Number → String

"kush" + 123 → "kush123"

✗ Space always get counted in script both thing are differ Empty vs Space

" " (vs) " - "

→ length 0 length → 1

Javascript → Class-02

* Console window
  - 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 loaded that's why it should always include before </body> end tag.

In console we can avoid (;) where main (JS) file we cannot (;) is necessary for end line.

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

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

* Comparison operator
  - Compare value, Not type
  - Compare both type & val

==	vs	===
"123" == 123 → true (type value)		"123" === 123 → false

N vs '0'	0 == '0'	Strict comparison
5 > '3'	false	
< true		
2 == '2' false	0 == false	
< true	< false	

* Comparison for non-number

'a' > 'A' Search Javascript unicode.
< true

→ 'P' < 'p' according to unicode
50 < 70 Check online available pdf.
true

Unicode.org/charts/PDF/U0000.pdf

*Go to link

a < b < c < d - - - z

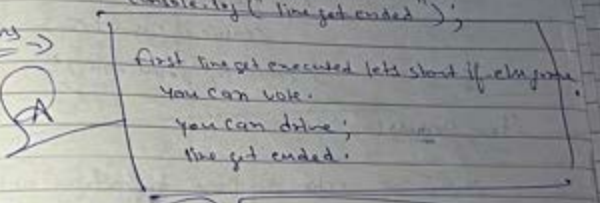
A < B < C < D < - - - Z

Conditional Statement

if else

if statement :-

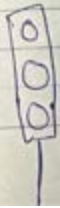
```
console.log("first line get execute bfr start if-else");
let age = 18;
if (age >= 18){
    console.log("you can vote");
    console.log("you can drive");
}
console.log("line get ended");
```



① what if age <= 13 then ans

first and line get ended same.

② Qs what functionality does a traffic light do.

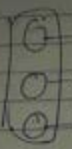


→ Red → Stop
Yell → Be ready
Green → lets go

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

→ String called data.

```
Console.log("lets run game");
if (color1 == Red) {
    Console.log("you stop");
} if (color2 == Yellow) {
    Console.log("you should be Ready")
} if (color3 == Green) {
    Console.log("you can go")
} Console.log("game end");
```



String Methods:

Methods → action the things to perform action.

format

const StringName.method()

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

String Method

Str.trim()

→

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

Array

Array in Java

Javascript Array vs

JavaScript Array vs Arrays in Java

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

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 unshift

arr.unshift("B");

Array Method

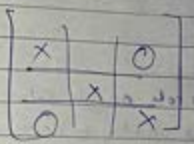
Index of :- return the index of the element which is passed to array container, most have same name, same camel letter due to function language (JS)

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

Array Const :-

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

Tic Tac Game :-



Blank space → represent (null) value.

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

Loop → To Run this more than once
same thing like repetitions
use syntax.

same as for loop

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

Print Table of (5)

5 × 1 = 5
5 × 2 = 10

for(int i=1; i<10; i++)

5 × 10 = 50

11-12 Of attend and took online Solve (3) questions
Look at back notes and try to cover again with RFI

Dom Events

Q what are Dom Event in JS?

A Dom events are the signal that when something is happens to the browser - like a user clicking, typing or hovering.

Event → when it happen.

Click → when 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 mind.

How → It checked at a door give a security to pass some basic information.

↳ Dom concept real world analogy

↓

element . add Event Listener() assign a guard to watch door

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

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

Event.PreventDefault() — The guard block someone from doing something (like stopping form 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.

Eg →

```
let btn = document.querySelector("button");
let p = document.querySelector("p");
let h1 = document.querySelector("h1");
let h3 = document.querySelector("h3");
```

→ add new event listner — — —

```
btn.addEventListener("click", function(){
    console.log("it got clicked");
    btn.style.
    backgroundColor = "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 changecolor() {
    console.log("hello inner world");
    this.style.backgroundcolor = "blue";
}
```

(call the fn below)

```
btn.addeventlistner("click", changecolor);
p.addeventlistner("click", changecolor);
```

Yes we can use it in the linear form.

```
document.querySelector("button").addEventListner("click",
() => changecolor(), changecolor);
```

```
function changecolor() {
    let red = math.floor(Math.random() * 255);
    let green = ;
    let blue = ;
}
```

0.5

val in b/w

i need to bring a fn for color which have been fix value as a string so.

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

Keyword functions in java script

let inp = document.querySelector("#id");
inp.addEventListener("keydown", function (e) {
console.log("key was pressed");
});

Keyboard Event Constructor

Syntax :-

new KeyboardEvent(type)
new KeyboardEvent(type, option)

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

options :- An object that, in addition of the property defined in UIEvent(), can have following properties

key (optional)

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

3-July-2015

Today mission is to owren 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 as 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.addEventListener("keydown", function (event){

if (event.ctrlKey && event.shiftKey && event.key === "O")

{ // logic to show Admin section }

});

Breakpoints :- It stores when i debugging finding errors . 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 Programming Interface.

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

[illegible] code is C++.
So Browser is written out with C++ and its multithread language.

Asynchronise feature?
→ Things can happen in the background, and we don't have the wait for them.
→ Instead of waiting for a time task (like an API call) JS continues executing other code even that time 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 callback Hell callbacks.

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