(Hindi)

New classes — 7th February-2023

Java Script - class 1

Javascript Basics :-
↳ logic and functionality.

what is Javascript ?

→ Light weight Programming Languages and Scripting Languages use to implements the Behaviours of the website.

History → Netscape navigator founded Javascript (1994)
Firstly it was called mocha, then Livescript then Javascript.

what can we do with Js :-

→ we can create web app/mobile application, Network apps.
→ CLI tools
→ Games.

Created in 1995 (Netscape) - in 10 days developed "Scripting Language"

Client side Scripting languages Execute on web browers.
The Js Engine/Environment help to run Js code) in Chrome is called V8 - firbox - Spider monkey.
(DONT GO IN DEPTH)

(Server Side). — To run Javascript outside the browers a c++ program added Js.
and NODE is Invented (by Ryan Dehl)

To Run Javascript

- Client Side → browers
- Server Side → NODE

① What is Server?

→ A Computer which gives back data to client's Computer when client searches Something.

→ To Run in Browser :-

→ Inspect in browser and go to 'Console'
& then you Can add :-

→ To Run in IDE :- (Code editor)

→ ① VS Code → Install
② Node.js → Install

Adding JS in Code

use <script> tag in HTML document.

Ex:- <script> → to point or last.
Console.log("Namaste Duniya")
</script> ~ used for client side scripting.

② we Can add script tag inside.
→ Head tag
→ body tag.

Q) Best practice?
→ Best practice is to add in body tag below of all the HTML Codes add script in last Body tag.

why?
→ It will create lag if added above in the body tag or in Head tag, first script will run and unable to parse that will error or delay in execution.

~~Comment~~

* Comment in Javascript (//) | Ctrl+/ select: // |

→ No significance in execution.

* External JS

Due to seperation of Concern, we will use External fill for Javascript.
we create Javascript fill the use Extension (.js)

* Linking <script src="index.js"> </script>

To run JS file using NODE :-

① VS Code → view (top bar)
② open terminal.
③ then make sure you are inside your working folder.
④ Command → Node index.js

~~Imp~~ Variables

Named memory location is called variables.

Creating variable in JS :- (var, let & Const)

As it is dynamic typed languages we no need to tell which data type to use, it automatically detects from the value.

Ex :- let a = 5 (numbers)
(let Keywords) let name = "Niraj" (string)
let status = true; (boolean)
let b = 12.5 (float)
↳ variable name

Var keywords

```
var a = 12;
var Name = "Niraj"
```

Let vs var

difference is of scope → Block, global.

→ let is a block scope variable.

```
{
  let a = 5; (only be used inside this block)
}
```

Eg:-

```
if (true)
{
  let a = 5;
}
Console.log(a); (error)
```

Now

→ var is a global scope variable. (Can change before in the code document)

→ let → redeclaration not allowed.

→ var → redeclaration is allowed.

Const variable → Fixed value of variable Can't be changed.

```
Const a = 5
a = 6 (reassign not allowed)
```

No redeclaration.

Variable Naming :- Rules

1. Cannot be a reserved keywords (let if x)
2. Meaningful
3. Cannot start with number (16x)
4. No space use '_'
5. CamelCase (FirstName)

Primitive types :- defined data types.

→ String – ("Niraj")
→ Numbers – (1, 2, 3, 4, 1.23, 1.54)
→ Boolean – true and false
→ Undefind – let a; not defined
→ null → empty variable (defined empty)

Note:- you can better Practice the code please used space my code.

Dynamic typing :- changing data type in js.

```
let a=5;
a = "Niraj";
Console.log(a); ↳ Print (Niraj)
```

Reference types (data types)

① objects (multiple variable linked/variables)
② Arrays – (List of similar items js)
③ Functions.

① objects :- (top level entity for multiple linked variables)

```
let person = {
    firstName = 'Niraj'
    age = 22
    };
```

Properties.

To Access :-
- dot Notation Notation (person.age)
- Bracket Notation (person['age'])

② Arrays :- ① used to Contain a list of items

```
let names = ['Love', 'Niraj', 'Sangam'];
```

0 1 2 — Indexes.

To Access indexes

names[1] – Niraj
names[0] → Love
names[3] ?
names[3] = Ramesh; // value added.

names[1]; 2 // updation overwrite.

ECMA
- ECMA is a standard of Javascript ECMA is an organizations which every each add updates in Javascript.
- ES6 → (Launched in 2015)

operators
① Arithmetic - (+, -, *, /, %, **) → powers
② Assignment - (=, +=, -=, *=, /=, %=)
③ Comparision - (>, <, >=, <=, ===, !==)
④ Ternary (Condition) (Cond^{n} ? val1 : val2)
⑤ Logical (AND, OR, NOT)
⑥ Bitwise (Bitwise AND, Bitwise OR)

Pre/post → Increment/decrement operators.

(++x;) → pre-increment
↙ firstly increment the value
Second, use the value

Ex:- let x=10
Console.log(++x);
↓
11

eg:- x++ → post increment operator
↘ first use the value second increment the value.

let a=6
Console.log(a++)
↓
6

Assignment - x = x+5
also (x+=5)

- x = x+3
also (x*=3)

Comparision
↳ Answer will always be in True or false

=.

=== (Strict equality) !== not equal

Equality

== (loose equality) | === (strict equality) } == v/s ===

== → Loose equality, value is same or not.

let num = 1
let str = '1' == give true;

=== → Strict equality, value + data is same or not

let num = 1;
let str = '1' === give false

Ternary operator :-

Cond^n ? val1 : val2

true ↑ false

Ex :- age = 27
let status = (age >= 18) ? 'valid' : 'Conf';

@ Logical operator

And (Cond^n1 && Cond^n2 && Cond^n3)

if any Condition is false the entire false All Conditions have to be true.

OR (Cond^n1 || Cond^n2 || Cond^n3)

any Condition is true then true and false, then only false.

Not ! { true → false, false → true }

② with Non Boolean (logical operators)

(true || false) → True

(true || true) → true

(false || false) → false

Now,

(false || 'Love') → Love

(true || 1 || 5) → 1

(Concept of falsy and truthy)

falsy → undefined, null, 0, false, "", NaN

Truthy → anything that is not falsy

(false || 'love') → 'love' → truthy

* Short Circuiting Concept in OR:

(false || 1 || 5)

↓

finds truthy then stop execution and prints (1)

* Bitwise operator :-

Bits → 0 (false)

→ 1 (true)

Bitwise AND → &

Bitwise OR → |

&

A	B	O/P
0	0	0
0	1	0
1	0	0
1	1	1

(1)

A	B	O/P
0	0	0
0	1	1
1	0	1
1	1	1

Operator precedence.

let c= a+b*d/c;

which operator first ?

[use brackets to resolve problem of precidency

let c=(a+((b*d)/c))]

Control statements :-

① If-else] two ways.
② Switch]

(i) if-else :- (if-elseif-else)

Single { if (condition) {
  - -
}

Can be multiple { else if (condition) {
  - -
}

Single { else {
  - -
}

② Switch-Case:

logic { input 1 → A
  2 → B
  3 → C

Syntax switch Case:-

```
switch (expression) {
 Case1: - -
 break ;
 Case2: - - -
 break ;
 Case3 - -
 break - -
 default - -
}
```

break after executing the case breaks the control statement and will not execute further.

Loops :- (Repetition of task)

① for loop
② while loop
③ Do-while loop
④ what is an Infinite loop ?
⑤ for-in-loop.
⑥ for-of-loop.

① for loop

```
for (let i=0; i<5; i++;
{
 Console.log(i);
}
```

i=0 → initialisation
i<5 → Condition
i++ → updatation

(0 1 2 3 4) print

② while loop

```
let i=0;
while (condition)
{ - - -
  i++; ← updation
}
```

③ Do-while loop :-

```
let i=0
do
{
  - - -
  i++
} while(i<10);
```

This execute atleast one time condition is true or not it execute one time atleast.

(Nitesh Kumar Tiwari)

Note :-

To Run - open your terminal and put Command -
- pwd - check path
- node index.js