

(Nirf)

Java Script - class 2

New class

[7th February - 2023]

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 Unscript then Javascript.

What can we do with JS?

- we can create web app/mobile application,
- Network apps: / created in 1995 (Netscape) - in Today
- CLI tools / developed "Scripting language"
- Games.

client side scripting languages execute on web browsers.

The JS Engine/Environment (loop to run JS code) in

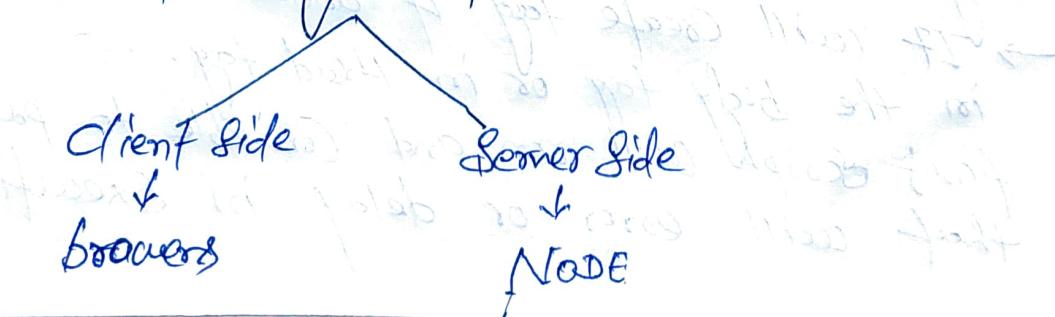
Chromo is Called JS - firbox - spider monkey.

(DON'T DO
IN DEPTH)

(Server Side), - To run Javascript outside the browser
a diff program added JS.

and NODE is invented (by Ryan Dahl)

To Run Javascript



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' of & 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 look.

Console.log("Name Dunya")
<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 tag if added above

in the body tag or in Head tag,
first script will run and invisible to page
that will error or delay in execution.

⑥ Comment

* Comment in javascript (//)

→ No significance in execution.

Ctrl + Shift + C

* External js

Due to Separation of Concern, we will use External file for javascript.
we create javascript file 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

Top

Variables

Named memory location is called variables

Creating variable in js :- (var, let & const)

As it is dynamic typed language 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 = "Viraj" (String)

let status = true; (boolean)

let b = 12.5 (float)

↳ variable
name

Var keywords

Var a = 12;

Var Name = "Vijay Kumar";

Let vs Var

Block

difference is of scope

→ 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 here in the code document)

→ let → redefinition not allowed

→ var → redefinition is allowed

→ fixed value of variable

Can't be changed.

Const a = 5

a = 6 (modification not allowed)

No reassignment.

Variable Naming :- Rules

① Cannot be a reserved keywords (let if +)

② Meaningful

③ Cannot start with number (1b+)

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
- undefined - let a; not defined
- null → empty variable (defined empty)

Note:- you can never
prefix the code please
use space by code.

Dynamic typing:- changing data type in JS.

```
let a = 5;
a = "Niraj";
console.log(a);
```

Point Niraj

Reference types (datatypes)

- ① objects (multiple variable linked/variable)
- ② arrays - (list of similar items of)
- ③ functions.

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

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

properties.

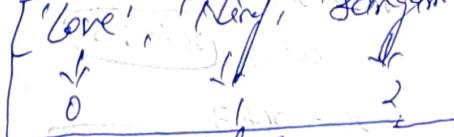
To Access :-

+ dot notation (person.age)

+ bracket notation (person[age])

② arrays :- ① used to contain a list of items.

```
let names = ['Core', 'Niraj', 'Ramesh'];
```



To Access

index

names[1] - Niraj

names[0] → Core

names[3] ?

names[3] = Ramesh; // value added.

names[1]; // update the element.

ECMA - ECMA is a standard of Javascript. ECMA is an organization which every year add changes in Javascript.
ES6 → (Launched in 2015)

operators

- ① Arithmetic - (+, -, *, /, %, **) → powers
- ② Assignment - (=, +=, -=, *=, /=, %=)
- ③ Comparison - (>, <, >=, <=, ==, !=)
- ④ Ternary (Condition) (Conditions ? val1 : val2)
- ⑤ Logical (AND, OR, NOT)
- ⑥ Bitwise (Bitwise AND, Bitwise OR)

Pre/post → Inrement/decrement operators.

$++x;$ → pre-increment

firstly increment the value
second, use the value

Ex:- let $x = 10$

Console.log($++x$);

\downarrow II

e.g:- $x++$ → post increment operator
let $a = 6$
Console.log($a++$)
 \downarrow I

first use the value second
increment the value.

Assignment -

$n = x + 5$
also $(x + 5)$

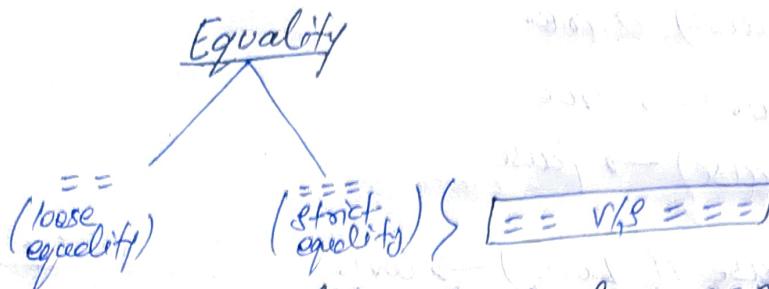
- $x = x + 3$

also $(x + 3)$

Comparison

→ Answer will always be in
True or False

$=$ \equiv \neq \neq (strict equality) \neq (not-equal)



$== \rightarrow$ loose equality, value is same or not.

let num = 2

let str = '2' $\quad ==$ gives true;

$== \rightarrow$ strict equality, value + data is same or not.

let num = 2;

let str = '2' $\quad ==$ gives false

Ternary operator:

Condⁿ ? val1 : val2

Ex:- age = 27

let status = (age >= 18) ? 'adult' : 'child'

② Logical operator

And (Condⁿ¹ && Condⁿ² && Condⁿ³)

if any Condition is false

the entire false All Conditions have to be true.

OR (Condⁿ¹ || Condⁿ² || Condⁿ³)

any Condition is free

then true and false, then only false.

(Not) !

{ True \rightarrow false }
{ false \rightarrow True }

Q with Non Boolean (logical operator)

(true || false) \rightarrow true

(true || true) \rightarrow true

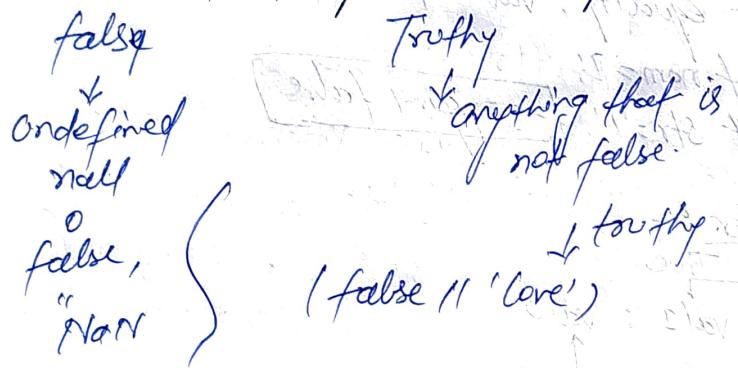
(false || false) \rightarrow false

Now,

(false || 'Love') \rightarrow Love

(true || 1 || 5) \rightarrow 1

/Concept of falsy and truthy)



* Short Circuiting Concept in OR

(false || 1 || 5) = 1

↓
finds truthy then stop exception
and points 1

* Bitwise operator :-

Bits \rightarrow 0 / false
1 / true

Bitwise AND \rightarrow &

Bitwise OR \rightarrow |

A	B	O/P
0	0	0
0	1	0
1	0	0
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 precedence]

Let $c = (a + ((b * d) / c))$

Control statements :-

- ① If-else] two ways.
- ② Switch

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

Single if Condition { (or)

Can be multiple { else if Condition } {

Single { else }

- ② Switch-Case:

logic { input $\rightarrow A$
 $\rightarrow B$
 $\rightarrow C$

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 $i = 0$; $i < 5$;

initialization

if;

update/fin

Condition

Console.log(i);

}

0 1 2 3 4 print

Syntax Switch Case:-

Switch (expression) of

Case1: --

break ;

break after executing
the case breaks

Case2: --

break ;
the Control

Case3: --

break ;
statement and all

break --

not execute

default: --

default

② while loop

let $i = 0;$

while (condition)

{
 if
 i++;
}
 ↑ update i

③ do-while loop :-

let $i = 0$

do {
 // statements
 i++
}

if

 & while ($i < 10$);

{ This executes atleast one time Condition
is true or Not if execute one time
atleast. } (Niraj Kumar (Lever))

Logic part of H:

Note :-

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