Turwash's Notes

7th February 2023

JavaScript - class 1

Javascript Basics :-

↳ Logic and Functionality.

What is JavaScript ?

→ Light weight programming language & Scripting language use to implement the Behaviour of the website

History → Netscape navigator Founded Javascript (1994) Firsty it was called Mocha., Then LiveScript then JavaScript

What can we do with JS :-

→ We can create web app / mobile app / network apps

→ CLI tools

→ Games

Client Side Scripting language executes on web browser.

The JS Engine (environment help to run JS code) in Chrome is called V8

Firefox → Spider Monkey

DONT GO IN DEPTH

:)

Server side,
To run JavaScript outside the Browser
a C++ program added with JS
and NODE is invented (by Ryan Dahl)

$$\begin{tikzcd}[column sep=small] & \text{To run JavaScript} \arrow[dl, dash] \arrow[dr, dash] & \\ \text{Client side} \arrow[d] & & \text{Server side} \arrow[d] \\ \text{Browser} & & \text{NODE} \end{tikzcd}$$

Q) what is Server?
→ A computer which gives back data to client's computer when client searches something.

→ To Run in Browser :-
→ 'Inspect' in browser & go to 'Console' & then you can code ;

→ To Run in IDE :- (code editor)
→ 1) VS code → Install
2) Node.js → Install

<u>Adding JS in Code</u>

use <Script> tag in HTML document.

ex:-
```
<Script>
    console.log("Namaste Duniya");
</Script>
```
→ to print or lock.

used for client side Scripting

✓ we can add Script tag inside

$$\begin{tikzcd} & \text{Head tag} \\ {} \arrow[ur] \arrow[dr] & \\ & \text{Body tag} \end{tikzcd}$$

Q) Best practice?

→ Best practice is to add in Body tag below of all the HTML codes. add Script in last of Body tag.

Why?

→ It will create Bug if added above in the body tag or in Head tag, First script will run & can't able to parse that will cause error or Delay in execution.

✓ Comment in Javascript using Forward Slashes (//)

→ No significance in execution.

External JS

Due to Separation of concern, we will use external File for Javascript.

we create javascript File we use extension $\boxed{.js}$

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 Variable

✓ 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 ~~(int)~~ (number)
let name = "Tuwash" (String)
let Status = true; (boolean)
let b = 12.5 (Float)
↑
Variable name

Let keyword

Var keyword

var a = 12,
var name = "Tuwash"

Let v/s var

Block

✓ difference is of Scope

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
(anywhere 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  (updation not
              allowed)
```

No redeclaration

Variable Naming

Rules

- ↳ cannot be a reserved keyword (let if X)
- ↳ meaningful
- ↳ Cannot Start with number (1b X)
- ↳ no space use '_'
- ↳ camelCase (firstName)

primitive Types → defined data types

→ String → ("Turwash")
→ Number → (1, 2, 3, 4, 1.23, 5.64)
→ Boolean → true or false.
→ Undefined → let a; not defined
→ null → empty variable (defined empty)

Dynamic typing.

↳ changing data type in JS

```
let a = 5;
  a = 'Turwash';
 console.log (a);
```

↳ Turwash printed

Reference Types (datatypes)

(1) Objects (multiple variables linked)
(2) Arrays (list of similar ~~datatypes~~ items (js))
(3) Functions

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

✓ let person = {
    firstName = 'Turwash',
    age = 24
    };

→ properties

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

(2) Arrays :-

↳ used to contain a list of items

✓ let names = ['love', 'rahul', 'sangram'];

0, 1, 2 — Indexes.

To Access :-
↓
indexes

names[1] → rahul
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 organisation which every year add updates in JavaScript.
ES6 → launched in 2015

Operators

(power)

1) Arithmetic (+, −, *, /, %, **)
2) Assignment (= , +=, −=, *= , /= , %=)
3) Comparison (>, < , >= , <=, ===, !==)
4) Ternary (condition) (condⁿ ? val 1 : val 2)
5) ~~Bitwise~~ Logical (AND, OR, NOT)
(6) Bitwise (Bitwise AND, Bitwise OR)

✓ pre/post → Increment/decrement Operator

++x ; → pre-increment

Firstly increment the value
Second, use the value

ex:- let x = 10
console.log (++x);

$\downarrow$

11

eg:-
let a = 6
Console.log(a++)

$\downarrow$

6

x ++ → post increment Operator

First use the value
Second increment the value

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'   == gives true;

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

let num = 1
let str = '1'   === gives False;

Ternary Operator :-

$$\begin{tikzcd} \text{cond}^n \arrow[r, bend left, "\text{True}"] \arrow[rr, bend right, "\text{False}"'] & \text{? val 1} & \text{: val 2 ;} \end{tikzcd}$$

ex:- age = 27

let status = (age >=18)? 'vote' : 'cant';

Logical operator

And

$$\underbrace{\text{(cond}^n 1 \ \&\& \ \text{Cond}^n 2 \ \&\& \ \text{cond}^n 3 \text{)}}$$

if any condition is False
the entire False
All conditions have to be true

OR

$$\underbrace{\text{(cond}^n 1 \ || \ \text{cond}^n 2 \ || \ \text{cond}^n 3 \text{)}}$$

any condition is true
then True
all False then only
False.

NOT !

True → False

False → True

With Non Booleans (Logical Operator)

(true || false) → true

(true || true) → true

(false || false) → false

Now,

(falsy || 'Love') → Love

(true || 1 || 5) → 1

Concept of Falsy & Truthy.

Falsy
↓
undefined
null
0
false
' '
NaN

Truthy
↓
anything that is not Falsy

truthy ✓
↓
(false || 'Love')

* Short Circuiting Concept in OR

(false || 1 || 5)
↓
Finds truthy then stop execution and prints $\boxed{1}$

Bitwise Operator

$$\begin{tikzcd} & 0 \text{ (False)} \\ \text{Bits} \arrow[ur] \arrow[dr] & \\ & 1 \text{ (true)} \end{tikzcd}$$

Bitwise AND → &

Bitwise OR → |

&

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

Control Statements :-

1) If - else
2) Switch

} two ways.

① if - else :- (if - elseif - else)

```
single {   if (condition) {
                 — —
           }
can be  {  else if (condition) {
multiple {      —— –
           }
single  {  else  {
                 —— —.
           }
```

② Switch - case :-

```
logic {  input 1 → A
               2 → B
               3 → C
```

Syntax Switch Case :-

```
Switch (expression) {
    Case 1 : ― ―
      break;
    Case 2 : ― ―
      break;
      :
      :
    default : ― ―
}
```

break, after executing the case breaks the Control Statement & 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

$$\underbrace{\text{for (let i = 0}}_{\text{initialisation}} \;;\; \underbrace{\text{i < 5}}_{\text{condition}} \;;\; \underbrace{\text{i++}}_{\text{updation}} \text{)}$$

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

0 1 2 3 4

2) <u>While Loop</u>

```
let i = 0;  ← initialization
while (condition)
   {
      _ _ _

      i++;  ← updation.
   }
```

3) <u>Do-while Loop :-</u>

```
let i = 0

do
 {
     _  _  _

     i++;
 } while (i<10);
```

This executes atleast one Time condition is True or Not it executes one Time atleast.