JavaScript

→ Javascript converts the static webpage into dinamic/interactive webpage.

Resources → 1) Visual Studio code.
2) Bable extension

file ⇒ file-name.JS

To run Javascript file :- node file.JS

• install the node.JS (Backend - plateform)

Client side program
1) Javascript
2) HTML
3) CSS

Server side program
1) python or
2) Javascript or
3) Java

Var var-name = 'value' ; ← variable declaration

→ Datatypes in Javascript :

1) undefined
2) Boolean
3) number
4) string
5) Biglint.
6) symbol

→ To check type we have : typeof() function.

→ number + string ⇒ string. → concatination
→ number - string ⇒ Substraction
→ string + string ⇒ concatination
→ string - string ⇒ NaN ⇒ not a number error
→ true + true ⇒ 1+1 = 2
→ false + true ⇒ 1
→ true = 1 / false = 0
→ false - true = -1

• To check is the number or not :
→ isNaN( ) ⇒ true/false

NaN ⇒ not a number

var x = 5 ;
var y = 5 ;

• console.log(" is both x & y are equal " + x==y);
↳ false ☹ → it is true but showing false

• to overcom we use atma script :
→ Console.log(`is both x and y are equal :- ${x==y}`)
↳ true ☺

→ select ctrl+D ← to edit same text.

- 5 falsy value in JS: 0, "", false, null, nan

Leap - year
- exactly divided by 4 & 2
- not needdivided by 100 &
- 400 ✓ divide

if true ⇒ It is leap-year

Operators:

1] Assignment operator ⇒ e =
2] Arithematical ⇒ (+, -, *, ÷, %) ⇒ (++, --) increment/dec
3] Comparison operato
4] Logical operator. (&, ||, !) → (and, or, not)
5] String operator (+) ⇒ concatenation
6) cunditional (ternery) operator.
7) Exponenthation operator (3**2) ⇒ $3^2 = 9$

[what is Diffrence between "==" & "===" ?]

```
var a = 5;
var b = "5";
```

→ console.log(a==b) ⇒ True. only chaking values are equal
→ console.log(a===b) ⇒ false value & type check ✓

→ Control statements & loops:

1) If... Else:

Syntax

```
① If(condition){
        operation 1
   }
   else{
        operation 2
   }
```

2] Ternery operater:
→ it takes 3 operands

② ternery operater

If true If false
↑ ↑

variable_name = (condition) ? value1 : value2

→ Short of If else

Using if/else

```
age = 14
if(age>18){
    you can vote
}
else{
    you can't vote
}
```

Using ternery operato

```
var age = 17;
(age>18) ? eligible :
    not-eligi
```

3) Switch case :-

```
switch (area) {
    case 'circle' :
        operation  break
    case : 'triangle' :
        operation 2  break
    Defaut :
        Operation
}
```

4) while Loop :-

```
var num = 0 ;
while(num < 10) {
    consol.log(num) ;
    num ++i
}
```

=> Block-scope

5) Do-while. Loop :

```
do {
    consol.log (num) ;
    num ++
} while (num<10) ;
```

6) For Loop :-

```
for (initializer ; condition ; iteration ) {
    // code
}
```

Functions :-

→ Block of code designed to perform perticular task

function functionName(parameter1, parameter2) {
  // code
}

- what is the difference between parameter & arguments :

function sum(a, b) { → parameters ⇒ function defination के time पे जो var pass करते हैं उसे parameter बोलते हैं !
  total = (a + b)
}

calling
sum(10, 20) → Arguments ⇒ calling time पे जो parameter पास करते हैं उसे Argument बोलते हैं !

ECMA Script :-

→ Javascript ⇒ 1996
→ Acmg Script ⇒ 1997

New Added features :

✓. Let & const	• Rest Operator	✓. Arrow function
✓. Template strings	• Destructuring.	• Spread Operator
✓. Default Arguments	• object properties	

1) Let vs const vs VAR

var ⇒ function scope
Let & const ⇒ Block Scope

Template string
without using concatination of string we can use ${var}

```
console.log('Hellow this ${number} is below 9');
```

3] Default Arguments :-

→ Default function parameters allow named parameters to be initialized with default values if no value or undefined is passed.

function mul(a,b) (1 2)	but if :	To tackle this scenario
{ return a*b; }	function mul(a,b) (1 2) { return a*b; }	function mul(a,b=5) { return a*b; }
console.log(mult(5,3)) (1 2);	console.log(mult(5)) (1);	console.log(mul(3)); (default value)
⇒ O/P : 15	⇒ O/P : error	⇒ O/P ⇒ 15
Defination → 2 parameter passed	→ 2 parameter required	→ 2 para required
calling → 2 Argument passed ✓	→ 1 Argument passed ✗	→ 1 Argument passed ✓

4] Arrow function :-

Traditional method of function:	Arrow method of function:
function sum() { (<= Defination)	Const sum = () ⇒ {
let a=5; b=6; ⇒	let a=5; b=6;
let sum = a+b;	sum = a+b;
return sum;	return sum;
}	}
sum() ← calling	sum() ← calling
	Const sum = () ⇒ '{a+b}';

Array

Var Array-name = [val1, val2, val3 ...]

→ Values can be of any Data type

Like Arr = ['Gaurav', 2, 13.2];

Var Names = ['Gaurav', 'Pratik', 'Anurag', 'Yash'];

Index : [0] [1] [2] [3]

Lower Boundary — Upper Boundary

- to calculate length of Array => array.len();

Strings :-

- Escape character
- finding a string in string
- Searching for string.
- Extracting string parts.
- Replacing string content
- other useful Method

Date & Time

→ Javascript Date objects represent a single moment in time in a platform-independent format.
Date object contain a number that represents milliseconds since 1 January 1970 UTC.

- There are 4 ways to create Date objects.

1) new Date()
2) new Date (year, month, day, hours, minutes, seconds, milliseconds)
3) new Date (milliseconds)
4) new Date (date string)

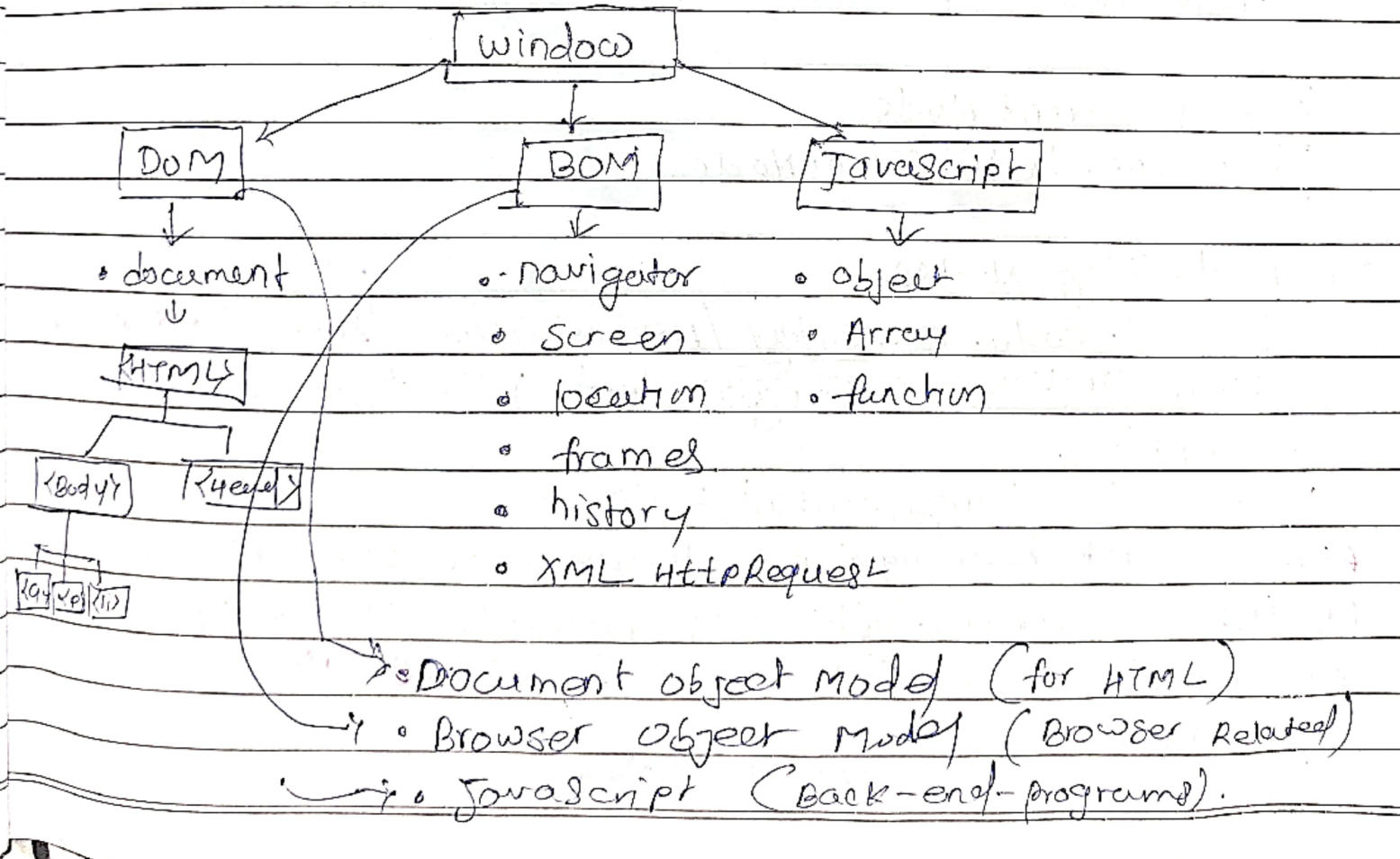
```
1) let currDate = new Date()
   console.log (currDate);
2) console.log (new Date());
3) console.log (new Date().toLocaleString()); // 9/11/2019, 1:28:07 PM
4) console.log (new Date().toString()); // Wed Sep 11 2019 13:28:07
                                           GMT+0530
```

Browser objet mod
windows.history.back();

Window vs Document

window	document
1: window is the main container or the global object and any operations related to entire browser window can be a part of window object.	1: whereas the DOM is the child of window object.
2: all the members like objects, methods or properties. If they are the part of window object then we do not refer the window object.	2: where in DOM we need to refer the document, If we want to use the document object, methods or properties.

window → parent
document → child



Dom Navigation

• inspect → console

⇒ Root element of DOM ⇒ <HTML>... </HTML>

• to find what we have writen into head tag
  → document.head

• Body
  → document.body

• to find childs of Body.
  • document.body.childNodes ⇒ It includes all enter, tab, spaces
  • document.body.childres ⇒ it gives only valid childs

• to check element have childs or not?
  document.body.hasChildNodes.

• to access childrens :-
  document.body.firstElementChild.

• const childTwo = document.querySelector('.child-two');
  childTwo.style.color = 'yellow'

• to find parentNode
  document.body.parentNode

• to find sibling Nodes.
  document.body.nextSibling / previousSibling. / previousElementSibli
  <button Onclick = "changeContent()"> click me </button>
  <script>
    const changecontent = () => {
for id ⇒ const headingchang = document.getElementById('id')
for class ⇒ document.getElementsByClassName('para')};
By tag ⇒ document.getElementByTagName('p');
  </script>

getElementsByName

querySelector() :-

document.querySelector('#id') /'.class' <= Access the element.

→ query selector only deals with first passed element. ('.para)

.para ✓ changes in first

.para ✗ no change.

Events in JavaScript :-

✓ 4 ways of writing Events.

✓ What is Event Object?

- Mouse Event
- Keyboard Event.
- Input events.

4 ways to writing Events :-

1. using inline events alert();
2. By calling a function();
3. Using Inline events (HTML onclick = "" & element.onclick);
4. using Event Listeners (addEventListener & IE is attachEvent);

Inside HTML

1] <href ='#' class "button_su_inner" onclick = "alert ('I am owesome, but NO one use me')">.

2] Second way

```
<script>
const callingfunction = () => {
    alert ('most common way of writting functions');
}
```

3] Third way :-

```
Const thirdway = document.getElementById('third way');   ← selecting id
thirdway.onclick = function() {
    alert ("third way of writing");
```

3 Arguments are
1. कौन Event को use करना है
2. कौन function को use करना है
3. true or false (event bubbling / event catching)

Calling using 4
4] Event Listener

```
const fourthway = document.querySelector('#fourthway');   ← selecting
                  ↳ takes 3 Arguments → function define under function
fourthway.addEventListener('click', () => {
        alert("I Love this way of writing");
});
```

↳ we can use normal function call.

- 4th way can be used multiple time.
- so we use this way most.
- it is trending & New technology.

5] Event object:
Event object is the parent object of the event object.
ex: MouseEvent, FocusEvent, KeyboardEvent, etc.

```
<script>
const fourthway = document.getElementById('fourthway');
const checkEvent = () => {
    alert('hum dekh the hai event object');
    → console.log(event);   ∈ total activity दर्शाये गा. clicked / double clicked
    → console.log(event.target)  ⇒ जहाँ click किया है बताये गा
    → console.log(event.type)  ⇒ mouse clicked, key Pressed, etc.
}
</script>
```

Mouse Events

The mouseEvent object Events that occur when the mouse interacts with html documents belongs to the mouse Event Object.

- onmousedown ← जब mouse click करेंगे.
- onmouseup ← जब mouse release करेंगे.

Example :-

```
<p id = "myp" onmousedown = "mouseDown()" onmouseup = "mouseUp()" >
    Paragrap
</p>

<script>
function mouseDown(){
    document.getElementById("myp").style.color = "Red";
}
function mouseUp(){
    document.getElementById("myp").style.color = "Blue";
}
</script>
```

- mouseenter ← mouse enter होगा element पे.
- mouseleave ← mouse Exit होगा element से.

```
<script>
const mEnter = document.getElementById('box');
mEnter.addEventListener('mouseenter', () => {
    console.log('Mouse enter Brow');
    mEnter.style.backgroundColor = 'Red';
});

mEnter.addEventListener('mouseleave', () => {
    console.log('mouse leave Bro');
    mEnter.style.backgroundColor = "green";
});
</script>
```

innerHTML ⇒ [ ]

this Area is inner HTML

Keyboard Events :- Events that occurs when user press the key on keyBoard.

Belongs to KeyboardEvent object.

- onkeypress
- onkeyDown
- onkeyUp.