Javascript

→ JS is a Programming language, we use it to give instructions to the computer.

Input (code) → Computer → output

Instructions

* alert("Abnq College");

* Our 1st JS code

→ console.log is used to log (Print) a message to the console.

console.log("Harsh");

* ~~Declaration~~

html ⇄ browser

↑↓

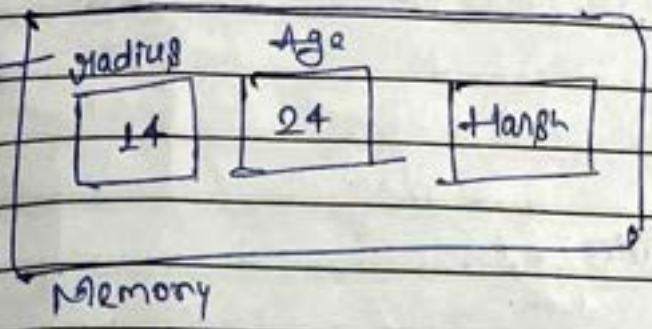
JS

Connect b/w Html and JS :-

(File name) ⇒ <script src="first.js"></

* Variables in JS

→ Variables are Containers of data.



Age = 24
name = "Harsh"

↳ In Programming we call Storing.

Code
=
```
full name = "Harsh";
Console.log (full name);
```

Code! -
=
```
full name = "Harsh";
age = 24;
x = null
Price = 99.99;
y = undefined;
Console.log (y)
```
→ undefined

* Boolean! - Two types (1) True (2) false

```
isFollow = true;
isFollow = false;
```

* Javascript is a Dynamically typed language.

a = b
↑ ↳ assignment operator

a = 24
↑

* Variable Rules :-

1. Variable names are case sensitive ("a" ≠ "A" is different.
2. only letters, digit, underscore (_) an $ is allowed. (not even space)
3. only a letter, underscore (_) or $ should be 1st character.

4. Reserved words cannot be variables.

* Reserved words :-

→ These keywords cannot be used as identifies for variables, functions, classes, etc.

1.	break	10.	do
2.	case	11.	else
3.	catch	12.	export
4.	class	13.	extends
5.	const	14.	false
6.	continue	15.	finally
7.	debugger	16.	for
8.	default	17.	function
9.	delete	18.	nan

- fullName → Camel Case.
- fullname X
- full_name → Snake Case.
- full-name → Kabab Case.
- FullName → Pascal Case.

Let, Const & Var :-

1. Var :- Variable can be re-declared & updated. → A global scope variable.
2. Let :- Variable cannot be re-declared but can be updated. → A block scope variable.
3. Const :- Variable cannot be re-declared or updated. → A block scope variable.

Var :-

Ex
```
age = 24;
age = 56;
age = 36;
console.log(age);
```
→ age = 36

Let :-

Ex1-
```
let age = 24;
age = 36;
age = 56;
console.log(age);
```
→

Const :-

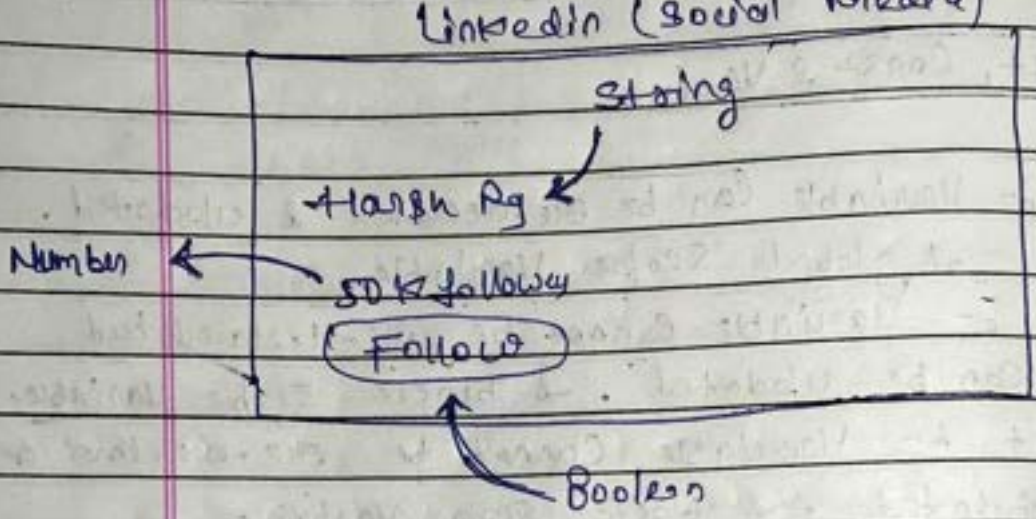
Ex :-
```
Const a = 10;
console.log(a);
```
→ 10

```
{ let a = 5;
  console.log(a);
}
{ let a = 8;
  console.log(a) = 8;
}
```
→ Block scope variable

→ 5 8

Data types in JS

1. Number 2. String 3 Boolean 4. undefined
5. Null 6. BigInt 7. Symbol.



Primitive (7) :-

→ Primitive data types are fixed.

Non-Premitive :-

(a) objects.

(1) Number :-

Ex! let Price = "40";
console.log (Price);

(2) String :-

let x = ("Hari");
console.log (Hari);

∴ We Have Seven Primitive data types.

Objects → Collection of values.

Students :-

↳ Name - String
↳ age - number
↳ marks - number
↳ isPass - boolean

} → objects

∴ In objects we store key pair value.

Key : Value

{
age : 24
name : "Rahul";
}

Ex
=

```
Const Student = {
    fullNam: "Harsh", };
    age : 24,
    cgpa : 8.2,
    isPass : true,
};
```

Ans :-

Student
{fullName : 'Harsh Raj', age : 24,
cgpa : 8.2, is Pass : true }

~~typeof object~~
typeof Student
'object'.

> Student ["fullName"]
> 'Harsh Raj'

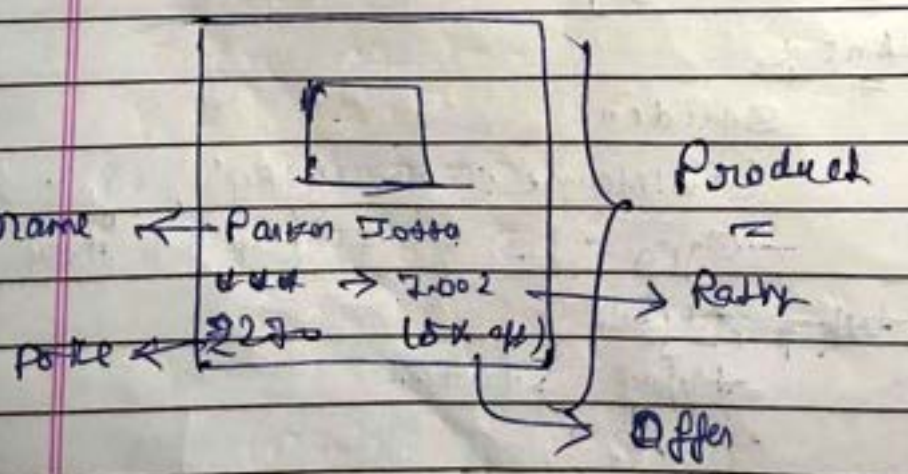
Ans
→ Console.log (Student["age"]);
→ Console.log (Student.age);

* Student["age"] = Student["age"] + 1;

Note
* let (update) ✓
* Const (Not update) X
* Const obj (update) ✓

* Practice question 1

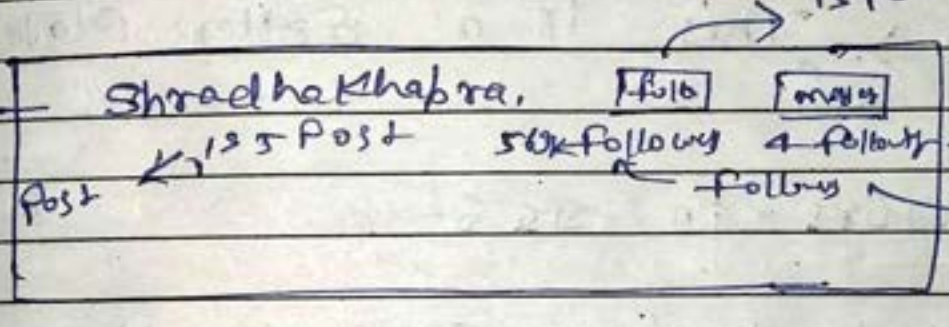
Q1. Create a Const object called "Product" to store information shown in the picture.



Page No.
Date

#1 const Product = {
name: "Parlcour",
rating = 3.5,
price = 270,
offer = 15,
};

Q2). Create a CONST object called "Profile" to store information, isfollow (Boolean



⇒ const Profile = {
name: "Shradha Khapra",
Post: 195,
isfollow: False,
follows: 563,
following: 1,
};

Console.log (typeof profile ["followers"]);

→ Number

Chapter - 2

Page No.
Date

Operators and Conditional Statements

Comments in JS

→ Part of code which is not executed.

```
// This is a single line comment
/* This is a multi - line
   comment. */
```

→ This is a Boiler Plate code.

Operators in JS :-

→ used to perform some operations on data

Arithmetic operator

+, -, *, /

- Modulus
- Exponentiation
- Increment (++)
- Decrement (--)

```
let a = 9;
let b = 8;
Console.log("a+b =", a+b);

Console.log(a+b);
```

⑥ Console.log a = 3;
b = 5;
Console.log ("a|b", a ** b);

* Exponentiation :- (**)

a ** b ⇒ a^b

↓ There are two uniary operators :-

1. Increament
2. Decrement

In Programing	Indecrement
a++ ⇒ (a+1)	a-- ⇒ (a-1)
(a = a+1)	(a = a-1)

let a = 5;
let b = 2;
Console.log ("a =", a, "& b =", b);
a = a+1; // a
Console.log ("a =", a); // a

* a++ (Post Increament) * a-- (Post Increment)
* ++a (Pre Increament * --a (Pre Increment)

Ex
```
let a=9;
let b=8;
Console.log("a=", a, "& b=", b);
Console.log("a--=" a--);
Console.log("a", a);
```

→ a=9 & b=8
a--=9
a=8

1. Assignment Operators :-

=, +=, -=, *=, %=, **=

→ Assignment operator is to assign value.

* += :- a+=1 → a=a+1

• a+=4 → a=a+4

* // Assignment operator

```
let a=6;
let b=7;
a+=4; // a=a+4;
Console.log("a =", a); // 13
b+=6;
Console.log("b=", b); // 13
```

2. Comparison Operators :-

1. Equal to (==) (Number)
2. Equal to & type (===) (Strict) check datatype
3. Not Equal to (!=)
4. Not Equal to & type (!==)

→ Comparison operators to compare two values.

Ex :- let a = 7';
let b = 8';
Console.log("a==b", a==b);
Console.log("a===b", a===b);
Console.log("a!=b", a!=b);
Console.log("a!==b", a!==b);

Ans
a==b false
a===b false
a!=b true
a!==b true

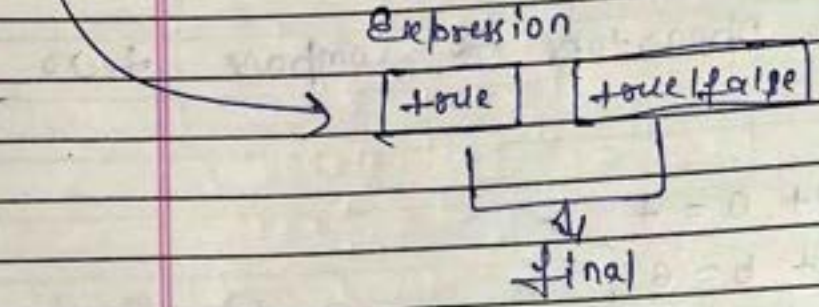
>, >=, <=, <

Nina

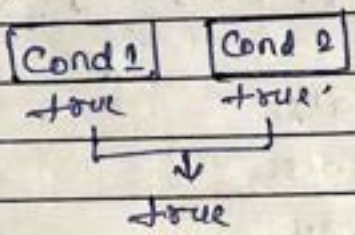
Ajanta
Page No.
Date

(3.) Logical operators → False / True

(a) Logical AND (&&)
(b) Logical OR (||)
(c) Logical NOT (!)



(1) Logical And (&&) :-



(4) Conditional Statements :-

→ To implement some condition in the code.

if Statement

```
let Color;
if (mode == "dark-mode")
{
  Color = "black";
}
```

Ch-3 (Loops and String)

* **Loops in JS :-**

→ Loops are used to Execute a piece of code again and again.

for Ex :-

① **for loop** — initialize → stopping condition → updation.

```
for(let i = 1; i<=5; i++) {
    console.log("apna college");
}
console.log("loop has ended");
```

Syntax

i = 1	1<=5 → T
i = 2	2<=5 → T
i = 3	3<=5 → T
i = 4	4<=5 → T
i = 5	5<=5 → T
i = 6	6<=5 → F

② **Infinite loop :-** A loop that never ends. (∞)

for infinite loop :-

```
for (let i = 1; i>=0; i++)
    console.log("i =", i);
}
```

Reg :- Infinit run and chrome is automatic sleep.

③ * While Loop :- → stop (True)

```
while (condition) {
    // do some work (updation)
}
```

Ex :-

```
let i = 1;
while (i <= 5) {
console.log("i=", i);
i++;
}
```

④ do - while loop :-

```
let i = 20;
do {
    console.log("Apna College");
i++;
} while (i <= 10);
```

Ans

= only one time run.
("Apna College")

⑤ for - of Loop :- → string

```
for(let val of str1) {
    // do some work
}
```

Ex :- let str = "Apna College";

```
for (let i of str)
{
    console.log("i=", i);
}
```

Page No.
Date

Ex 8 -
```
let str = "Javascript";
let size = 0;
for (let i of str)
{
// iterator → Character
console.log("i = ", i);
size++;
```

⑥ for-in-loop :-

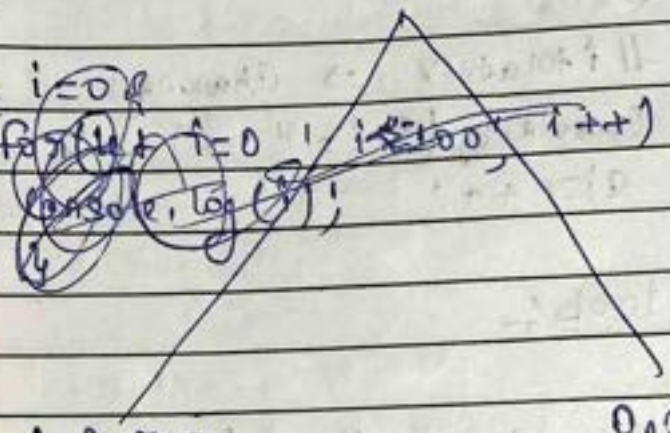
```
for (let key in obj var) {
    // do some work
}
```

Ex
```
let Student = {
    name : "Harsh Raj",
    age : 20
    CgPA : 7.8,
    isPass : true,
};
for (let i in student) {
    console.log(i);
}
for (let key in Student) {
    console.log("key =", key, "value =", student[key]);
}
```

Practice question

1) Print all the even no 0 to 100.

⇒ for (let i=0; i<=100; i++)
{
console.log(i);
}



1, 3, 5, 7
(don't Print)

0, 2, 4, 6, 8
(Print)

loop

if (num % 2 == 0)

for (let num = 0; num <= 100; num++)
{
if (num % 2 == 0)
{
"even number"
console.log ("num =", num);
}
}

Strings

→ Strings is a sequence of characters used to represent text.

- Create String

let str = "Apna College"

- String Length

Str.length

- String Indices

Str[0]. Str[1]. Str[2]

Template Literals in JS

→ A way to have embedded expression in strings.

`This is a template literal`.

String Interpolation

To create strings by doing substitution of placeholders.

variable

`String text ${expression} string text`.

Ex :-

```
let specialString = `This is a template literal`;
console.log(specialString);
```

Ex :-

```
let obj = {
    item : "pen",
    Price : 10,
};

let output = `the cost of ${obj.item} is
    ${obj.price} rupees`;
console.log(output);
```

// Template Literals

```
let specialString = `This is a template literal
    ${1+2+3}`;
console.log(specialString);
```

→ This is a template literal. 6.

* Escape character is the part of string.

```
console.log("Apna \n College");
```

↳ Escape character means the line in next line.

→ Apna
College

∴ \n → next line.
∴ \t → tab space.

```
console.log("Apna \t College");
```

→ Apna ___ College
(space)

* String Method in JS

→ There are built-in functions to manipulate a string.

1. str.toUpperCase()
2. str.toLowerCase()
3. str.trim() // removes whitespace.

* Methods is like a block of code.

1) str.toUpperCase() :-

```
let str = "Apna College";
let newStr = str.toUpperCase();
console.log(str);
console.log(newStr);
```

→ apna college
APNA COLLEGE.

2) Str.toLowerCase()

3. Str.trim() // removes whitespace

Ex:- let str = " Apna college JS "
→ Apna college JS

4. Str.slice(start, end?) // return a part of string.

Ex:- a b c d e f g
↑ ↑
start end

Ex
let Str = "hello";
Console.log(Str.slice(0, 2));

→ hel dy

5) Str1.concat(Str2) // Joins Str2 with Str1

→ Concat (Means Join b/w two words)
let str1 = "Harsh";
let str2 = "Ag";
let res = str1.concat(str2);
Console.log(res);

→ Harsh Ag

9. str. replace(searchVal, newVal)

→ Eg:- let str = "hello";
console.log(str.replaceAll("lo", "p"));

→ help

7. str.charAt(idx)

Ch-4

Arrays

→ It is a collection of items. (Linear way)

Create Array

let heroes = ["ironman", "hulk", "thor", "batman"];

let marks = [96, 75, 48, 83, 66];
let info = ["rahul", 86, "Delhi"];

* Array indices :-

arr[0], arr[1], arr[2], ...

→ Linear way

97	82	64	76	36
1	2	3	4	5

* Strings in Javascript is immutable.
* Array in Javascript is Mutable.
* Looping over an Array.

→ To print all elements in Array.

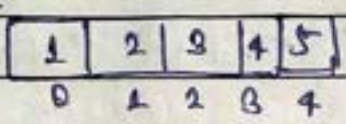
Loops → Iterable (Strings, objects, arrays).

1). For Loop :- (length)

```
for ( let idx = 0 ; idx < arr.length ; idx++)
{

}
```

arr



arr.length = 5

Ex :-

```
let heroes = ["ironman", "thor", "hulk", "shaktiman",
              "spiderman", "antman"]
for ( let idx = 0 ; idx < heroes.length ; idx++)
{
    console.log ( heroes[idx] ) ;
}
```

// for of :-

```
for(let hero of heroes){
    console.log(hero);
}
```

Q). For a given array with marks of student -> [85, 97, 44, 37, 76, 60] : Find the average marks of the entire class.

→

```
let marks = [85, 97, 44, 37, 76, 60];
for(let i=0; i<
let sum=0;
for(let i=0; i<
for(let val of marks){
    console.log(val);
}
```

```
let marks = [85, 97, 44, 37, 76, 60];
let sum = 0;
for(let val of marks)
{
  sum += val;
}
let avg = sum/marks.length;
console.log(`avg marks of the class = ${avg}`);
```

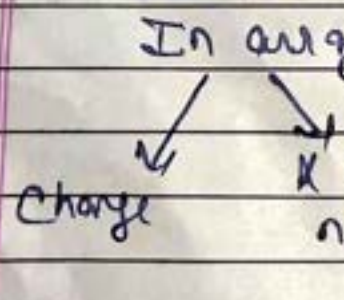
Q). For a given array with prices of 5 items → [250, 645, 300, 900, 50] All items have an offer of 10% off on them. Change the array to store final price after applying offer.

→ let items = [250, 645, 300, 900, 50];

for (let i=0; i < items.length; i++) {
let offer = items[i] / 10;
items[i] -= offer;
}
console.log(items);

Array Methods :-

1. Push() : add to end
2. POP() : delete from end & return.
3. toString() : converts array to string.



Push → add (to add something).
POP → sub (to delete something).
Shift → delete from start.

Ex :-

```
let foodItems = ["potato", "apple", "litchi",
                 "tomato"];

foodItems.push("chips", "burger", "Paneer");
Console.log(foodItems);
```

[7] ["potato", "apple", "Litchi", "tomato", "chips", "burger", "Paneer"].

(2) pop() :-

```
let foodItems = ["potato", "apple", "litchi",
                 "tomato"];
Console.log(foodItems);
let deletedItem = foodItems.pop();
Console.log(foodItems);
Console.log("deleted", deletedItem);
```

(3) toString

→ Ex :- let foodItems = ["potato", "apple", "litchi",
"tomato"];

```
Console.log(foodItems.toString());
Console.log(foodItems);
```

Array Methods

* Slice() :- returns a piece of the array
↓
original array change ✗

. Slice(startIdx, endIdx)

* Splice() :- change original array (add, remove, replace).

. splice(startIdx, delCount, new El...

Eg :-
let arr = [1, 2, 3, 4, 5, 6, 7];
// arr.splice(2, 2, 101, 102);
// Add Element
arr.splice(2, 0, 101);

~~Note~~

Function & Method

* Functions :-
=> Block of code that performs a specific task, can be invoked (call) whenever needed.

fix of code
↓
Call

2) Function Call :-

input → Black Box → output

functionName();

* Redundancy → repeat

1). Function Definition :-

function functionName() {
  // do some works
} → Block of code.

function functionName(Param1, Param2....) {
  // do some works
}

Ex :-
function myfunction() { // Parameter → input
  Console.log("we are learning JS");
}

Call function
myfunction(); // argument

→ Name

Note :- NaN (Not a Number).

Eg :-
```
function sum (a,b) {
    s = a+b;
    return s;
}
let val = sum(4+5);
console.log (val);
```

Note :-

* function Parameters → They like a local Variables → Block scope.

* Arrow function

→ Compact way of writing a function.

```
const functionName =>( Param1, Param2 ){
    //do some work
}
```

```
const sum =>(a,b) {
    return a+b;
}
```

* => (Arrow function)

```
function add (a,b){
    return a+b;
}
const arrowadd = (a,b) => {
    return a+b;
};
```

Ex :-

```
function countvowel (str) {
    let count = 0;
    for (const char of str) {
    if (char === "a" || char === "e" || char === "i"
        || char === "o" || char === "u") {
        count ++;
    }
    }
    return count;
}
```

for Each Loop in Arrays :-

→ arr. forEach (CallBack Function)

CallBack Function :- It is a function to Execute for Each Element in the array.

→ A CallBack is a function passed as an argument to another function.

```
arr. forEach ( val ) => {
    console. log ( val );
});
```

Method Function

Ex :-

```
let arr = [ 1, 2, 3, 4, 5 ];

arr. forEach ( function printVal ( val ) {
    console. log ( val );
});
```

Note :- ForEach is Higher order function = Higher Order Methods.

Q). For a given array of numbers, print the Square of Each value using the ForEach Loop.

[2, 3, 4, 5, 6] → 4, 9, 16, 25, 36
m * n

```
let num = [2,3,4,5,6];
num.forEach((num) => {
        console.log(num*num);
});
```

Next Ex:-

```
let nums = [67, 52, 39];
let calcsquare = (num) => {
  console.log(num * num);
};
nums.forEach(calcsquare);
```

Some More Array Methods :-

1. Map :-

Creates a new array with the results of some operation. The value its callback return are used to form new array.

arr.map(CallbackFnx (value, index, array))

```
let newArr = arr.map((val) => {
    return val*2;
})
```

→ Map returns a New Array.

Ex ?-

```
let nums = [67, 52, 39];

let newArr = nums.map ((val) => {
        return val * 2
});
```

Output :- 67, 52, 39

~~console.log(newArr);~~

2. Filter ?-

Creates a new Array of elements that give true for a condition / filter.

Eg ?- all even elements.

```
let newArr = arr.filter ((val) => {
        return val % 2 === 0;
});
```

Ex :- let arr = [1,2,3,4,5,6,7];
let evenArr = arr.filter((val) => {
return val % 2 == 0;
});
console.log(evenArr);

3. Reduce :-

It performs some operations & reduces the array to a single value. It returns that single value.

arr.reduce((res, curr) => {
return res + curr
})

[1, 2, 3, 4]
↑ ↑ ↑ ↑
res curr curr curr

res → 1
curr → 2

Ex :-
let arr = [1,2,3,4]
const output = arr.reduce((res, curr) => {
return res + curr;
});
console.log(output);

DOM (Document object Model)

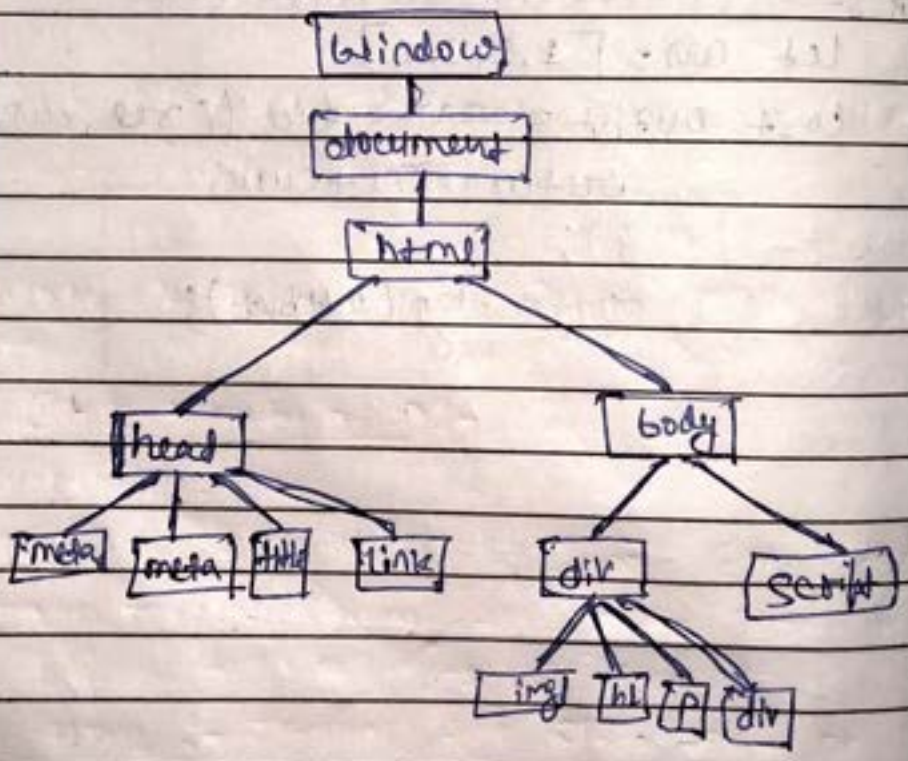
* Window object :-

→ The window object represents an open window in a browser. It is browser's object (not Javascript's) & is automatically created by browser.

→ It is a global object with lots of properties & methods.

* What is DOM ?

→ When a webpage is loaded, the Browser creates a Document object Model (DOM) of the Page.



What is HTML DOM?

→ The HTML DOM is a Standard object model and Programming Interface for HTML.

, The HTML elements as objects
, The Properties of all HTML elements.
, The methods to access all HTML elements.
, The events for all HTML elements

→ The HTML Dom is a Standard for how to get, change, add, or delete HTML elements

Note :- Console.log → Print
Console.dir → document → Property (print)
methods

DOM Manipulation :-

① Selecting with id :-

document.getElementById("myId")

② Selecting with Class :- (HtmlCollection is very similar to array)
document.getElementByClassName("myclass")

③ Selecting with tag :-
document.getElementsByTagName("p")

* #id → unique

④ Query Selector :-

- document.querySelector ("myId / myclass / tag")
  //return first element.

- document.querySelectorAll ("myId / myclass / tag")
  //returns a NodeList.

Ex :- let elements = document.querySelector("p");
Console.dir (elements);

Ex :- let firstEl = document.querySelector("p");
console.dir (firstEl);

Ex :- let AllEl = document.querySelectorAll("p");
Console.dir (allEl);

Note :- queryselector Return Node List.

DOM Manipulation :-

Properties :-

1. tagName :- Returns tag for Element node.
2. Inner Text :- Returns the text Content of the Element and all its children.
3. InnerHTML :- Returns the plain text or HTML Contents in the Element.

DOM tree :-

1). text nodes.
2). Comments,
3). Elements.

(4) textContent :- returns textual content even for hidden elements.

Attributes :-

- getAttribute (attr) // to get the attribute value.
- setAttribute (attr, value) // to set the attribute value.

Style :-

- node.styles

Insert element :- let el = document.createElement ("div")

1). node.append (el) // adds at the end of node (inside)
2). node.Prepend (el) // adds at the start of node (inside)
3). node.before (el) // adds before the node (outside)
4). node.after (el) // adds after the node (outside)

Delete Element :-

1). node.remove () // removes the node.

Homework

- appendChild()
- removeChild()

Events

→ The change in the state of an object is known by an event.

→ Events are fired to notify code of "Interesting changes" that may affect code execution.

1. Mouse events (click, double click etc).
2. keyboard events (keyPress, keyup, keydown)
3. form events (submit, etc).
4. Print event & many more.

* ~~ondblclick~~ (Means Double Click)

Ex:-

(1) <button onclick = "console.log('button was Clicked'); alert('helo!')">
Click me!
</button>

(2) <button ondblclick = "console.log('button was clicked')">
Click me 2 times!
</button>

Event Object :-

→ It is a special object that has details about the event.

→ All event handlers have access to the Event Object's Properties and methods.

```
node.event = (e) => {
    //handle here
}
e.target, e.type, e.clientX, e.clientY
```

Ex :-

```
let btn1 = document.querySelector("#btn1");
btn1.onclick = (evt) => {
Console.log(evt);
Console.log(evt.type);
Console.log(evt.target);
Console.log(evt.clientX, evt.clientY);
}

let div = document.querySelector("div");
div.onmouseover = () => {
    console.log("you are inside div");
};
```

* Note :- Agar Hum Inline way Mai Handle kiye Hai Event ko [redacted] or Js mai kahi Hai [redacted] Event ko Handle to priority Js ki Hogi Na ki Inline Handler ki.

Note :- Event reference MDN

Page No.
Date

Note :- Additional Information about the Event Itself through one Event Object.

Event Listener :- Double click or click → Function → Handler

node.addEventListner (event, Callback);
node.removeEventListner (event, Callback);

* Note :- The Callback reference should be same to remove.

Ex :-

1) node.addEventListner (event, Callback) :-

Ex :- let btn1 = document.querySelector("#btn1");
btn1.addEventListner ("click", (evt) => {
Console.log ("button 1 was clicked");
Console.log (evt);
Console.log (evt.type);
});

btn1.addEventListner ("click", () => {
Console.log ("button 1 was clicked - handler");
});

let div = document.querySelector ("div");

② node.removeEventListner (event, callback) :-

Ex :- let btn1 = document.querySelector("#btn1");
btn1.addEventListener("click", () => {
console.log("button 1 was clicked - handler1");
});
btn1.addEventListener("click", () => {
console.log("button 1 was clicked - handler2");
});

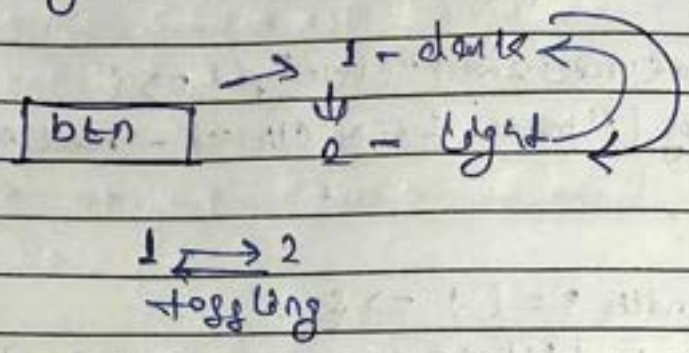
const handler3 = () => {
console.log("button 1 was clicked - handler 3");
};

btn1.addEventListener("click", handler3);
btn1.addEventListener("click", () => {
console.log("button 1 was clicked - handler 4");
});

btn1.removeEventListener("click", handler3);

Let's Practice

Q. Create a toggle button that changes the screen to dark-mode when clicked & light-mode when clicked again.



Toggle btn is used to change the screen is light mode to dark mode.

HTML

⇒ <Button> id = "mode" > Mode </Button>

JS

```
let modeBtn = document.querySelector("#mode");
let currMode = "light"; // dark

modeBtn.addEventListener("click", () => {
  if (currMode === "light") {
    currMode = "dark";
    document.querySelector("body").style.backgroundColor = "black";
  } else {
    currMode = "light";
    document.querySelector("body").style.backgroundColor = "white";
  }
});
```

Console.log(currnode);
});

* append.child() :- The appendchild() method of the Node interface adds a node to the end of the list of children of a specified parent node.

Note:- If the given child is a reference to an existing node in the document, appendchild() moves it from its currentposition to the new position.

* append() :- append() Method inserts a set of Node objects or string objects after the last child of the document.

(add a new child node to an earsthy parent node).

Classes and Objects :-

Prototype in JS

→ A Javascript object is an entity having state and behaviour (Property & method).

JS objects have a special property called Prototype.

This is basically used to get prototype

We can set prototype using __Photo__.

* If Object & Prototype have same Method, Object's method will be used.

Eg :-

```
= Const Student = {
    fullName : "Harsh Raj",
    marks : 94.4,
    PrintMarks : function(){
    Console.log("marks = " + this.marks)
  },
};
```

(this.marks // Means Student Marks.)

Ex :- Const employee = {
    calcTax() {
        console.log("tax rate is 10%.");
    },
};
Const KaranArjun = {
    salary : 50000,
};
KaranArjun.__proto__ = employee;

* Prototype means reference to an object.

Classes in JS :-

→ Class is a Program - Code template for creating objects.
→ Those objects will have some state (variable) & some behaviour (function) inside it.

Class MyClass {
    Constructor() {}
    myMethod() {}
}
let myobj = new MyClass();

} blueprint.

→ Classes is a blueprint of an object.
→ It is like a template.

Eg :- Class Toyotacar {
start () {
console.log ("start")
}
stop () {
console.log ("stop")
}
}
let fortuner = new Toyotacar();
let lexus = new Toyotacar();

* (this) → this means each individual object special method.

* Constructor() Method is:

- automatically invoked by new
- initializes object

Class myclass {
constructor() {
myMethod() {
}
}

Ajinta
Page No.
Date

Spotify Clone (Done)

Portfolio

Javascript Programming questions

1. How to find Duplicate elements in a given Array?

⇒
```
<script>
  Const arrNumber = [1, 2, 8, 2, 3, 8]
  Const duplicates = arrNumber.filter((ele, index, arr) => arr.indexof(ele) !== index)
  Console.log(duplicates)
</script>
```

Filter Method :-

filter (Predicate : (value : number, index : number, array : number [ ]) => value is number, thisArg ? : any number[ ]

⇒ A function that accepts upto three Arguments. The filter method call the predict function one time for each element in the arry

Q. Missing Number in a integer array 1 to 100,

```
const arr = [1, 2, 3, 4, 5, 6, 9, 10]
const missArray = [ ]
const missingValue (arr) => {
const minvalue = Math.min (...arr)
const maxvalue = Math.max (...arr)
for (let i = minValue; i < maxValue; i++)
  {
    if (arr.indexOf (i) < 0){
      missArray.push (i)
  }
}
return (missArray)
}
Console.log (missingValue (arrNum))
```

22. How to find second largest value, and remove First largest value in array.

⇒ Step1:- find largest value from array.
Step2:- find index of largest value.
Step3:- Delete index from Array using splice method.
Step4:- Apply same logic that use for find largest value.

⇒ Largest

```
Const Arr = [ { name: "Ayan", Age: 40 },
              { "Rupa", 41 },
              { "Tash", 29 },
              { "Rakesh", 70 },
              { "Sid", 85 },
            ];

Const filteredItem = arr.filter((item) =>
{
  return item.age > 50
});
console.log(filteredItem);
```