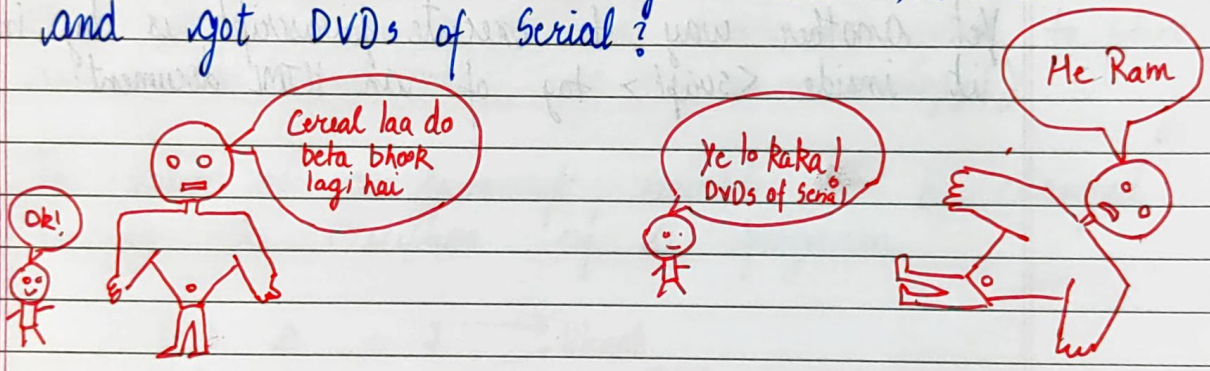
Introduction to programming

Programming is a way to talk to computers.
A language like Hindi, English or Bengali can be used to talk to a human but for computers we need straightforward instructions

Computer is Dumb!
When was the last time you ordered some cereal and got DVDs of Serial?



Programming is the act of constructing a program, a set of precise instructions telling a computer what to do.

What is Ecmascript?
ECMAScript is a standard on which Javascript is based!
It was created to ensure that different documents on javascript are actually talking about the same language.

JavaScript & ECMAScript can almost always be used interchangably
JavaScript is very liberal in what it allows.

How to execute JavaScript?

JavaScript can be executed right inside one's browser. You can open the javascript console and start writing javascript there

Another way to execute javascript is a runtime like Node.js which can be installed and used to run javascript code.

Yet another way to execute javascript is by inserting it inside <script> tag of an HTML document.

Chapter 1 - Variables & Data

Just like we follow some rules while speaking english (the grammar), we have some rules to follow while writing a javascript program. The set of these rules is called Syntax in javascript.

What is a Variable?

A Variable is a container that stores a value. This is very similar to the containers used to store rice, water and oats (Treat this as an analogy!)

The value of a javascript variable can be changed during the execution of a program

```
var a = 7;   → literal
let a = 7;   ⇒ Declaring Variables
    ↑   ↑
Identifier  assignment operator
```

Rules for choosing variable names

→ Letters, digits, underscores & $ sign allowed.
→ Must begin with a $, _ or a letter.
→ JavaScript reserved words cannot be used as a variable name
→ Harry & haRRy are different variable (case sensitive)

Var vs let in JavaScript

1. Var is globally scoped while let & const are block scoped
2. Var can be updated & re-declared within its scope
3. let can be updated but not re-declared
4. const can neither be updated nor be re-declared.

5. var variables are initialized with undefined whereas let and const variables are not initialized.
6. const must be initialized during declaration unlike let and var

Primitive Data Types & Objects
Primitive data types are a set of basic data types in javascript
Object is a non primitive datatype in javascript

These are the 7 primitive datatypes in javascript

- Null
- Undefined
- Number
- Boolean
- String
- BigInt
- Symbol

Object
An object in JavaScript can be created as follows.

```
const item = {
    name : "Led Bulb",
    price : "150"
}
```

key ← name, price ; value → "Led Bulb", "150"

Quick Quiz: Write a JavaScript program to store name, phone number and marks of a student using objects.

Chapter 1 - Practice Set

1. Create a variable of type string and try to add a number to it.

2. Use typeof operator to find the datatype of the string in last question

3. Create a const object in javascript. Can you change it to hold a number later?

4. Try to add a new key to the const object in Problem 3. Were you able to do it?

5. Write a JS program to create a word-meaning dictionary of 5 words.

Chapter 2 - Expressions & Conditionals

A fragment of code that produces a value is called an expression. Every value written literally is an expression. for ex: 77 or "Harry"

Operators in JavaScript

1. Arithmetic operators

+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation
/	Division
%	Modulus
++	Increment
--	Decrement

2. Assignment Operators

=	x = y
+=	x = x + y
-=	x = x - y
*=	x = x * y
/=	x = x / y
%=	x = x % y
**=	x = x ** y

3. Comparison Operators

==	equal to
!=	not equal
===	equal value and type
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

4. Logical Operators

&&	logical and
||	logical or
!	logical not

Apart from these, we also have type and bitwise operators. Bitwise operators perform bit by bit operations on numbers.

operands

7 + 8 = 15 → Result

↳ operator

Comments in JavaScript

Sometimes we want our programs to contain a text which is not executed by the JS Engine

Such a text is called comment in Javascript

A comment in Javascript can be written as follows :

```
let a = 2; // this is a single line comment
```
→ Single line Comment

```
/*
  I am a
multiline comment
*/
```
} Multiline comment

Sometimes comments are used to prevent the execution of some lines of code

```
let switch = true;
// switch = false
```
→ Commented line wont execute

Conditional Statements

Sometimes we might have to execute a block of code based off some condition.
For example a prompt might ask for the age of the user and if its greater than 18, display a special message.

In JavaScript we have three forms of if ... else statement.

1. if statement
2. if ... else statement
3. if ... else if ... else statement

If statement

The if statement in JavaScript looks like this:

```
if (condition) {
    // execute this code
    }
```

The if statement evaluates the condition inside the ( ). If the condition is evaluated to true, the code inside the body of if is executed else the code is not executed.

if- else statement

The if statement can have an optional else clause. The syntax looks something like this

```
if (condition) {
    // block of code if condition true
    }
else {
    // block of code if condition false
    }
```

If the condition is true, code inside if is executed else code inside else block is executed

if - else if statement

Sometimes we might want to keep rechecking a set of conditions one by one until one matches. We use if else if for achieving this.

Syntax of if...else if looks like this

```
if (age > 0) {
    console.log("A valid age");
}
else if (age > 10 && age < 15) {
    console.log("but you are a kid");
}
else if (age > 18) {
    console.log("not a kid");
}
else {
    console.log("Invalid Age")
}
```

JavaScript ternary Operator
Evaluates a condition and executes a block of code based on the condition

Condition ? exp1 : exp2

Example Syntax of ternary operator looks like this:

(marks > 10) ? 'yes' : 'No'

↳ if marks are greater than 10, you are passed else not

Chapter 2 - Practice Set

1. Use logical operators to find whether the age of a person lies between 10 and 20?

2. Demonstrate the use of switch case statements in JavaScript

3. Write a Java Script program to find whether a number is Divisible by 2 and 3.

4. Write a JavaScript program to find whether a number is Divisible by either 2 or 3.

6. Print "You can Drive" or "You cannot Drive" based on age being greater than 18 using ternary operator.

Chapter 3 - Loops & functions

We use loops to perform repeated actions. For example - If you are assigned a task of printing numbers from 1 to 100, it will be very hectic to do it manually, Loops help us automate such tasks

Types of loops in JavaScript

for loop → loop a block of code no of times
for in loop → loops through the keys of an Object
for of loop → loops through the values of an object
while loop → loops a block based on a specific condition
do-while loop → while loop variant which runs atleast once

The for loop

The syntax of a for loop looks something like this

```
for ( statement 1 ; statement 2; statement 3) {
  // code to be executed
}
```

→ statement 1 is executed one time
→ Statement 2 is the condition base on which the loop runs (loop body is executed)
→ statement 3 is executed everytime the loop body is executed

Quick Quiz : Write a sample for loop of your choice.

The for-in loop
The syntax of for-in loop looks like this

```
for (key in object) {
  // code to be executed
}
```

Quick Quiz: Write a sample program demonstrating for-in loop

Note - for-in loops also work with arrays which will be discussed in the later videos.

The for-of loop
The syntax of for-of loop looks like this

```
for (variable of iterable) {
  // code
}
```

↳ For every iteration
↳ Iterable data structure like Arrays, Strings etc.

Quick Quiz: Write a sample program demonstrating for-of loops

The while loop
The syntax of while loop looks like this:

```
while (condition) {
  // code to be executed
}
```

Note: If the condition never becomes false, the loop will never end and this might crash the runtime.

Quick Quiz: Write a sample program demonstrating while loop.

The do-while loop

The do while loop's syntax looks like this:

```
do {
  // code to be executed  => Executed at least once
}
while (condition)
```

Quick Quiz: Write a sample program demonstrating do while loop

Functions in Java Script

A JavaScript function is a block of code designed to perform a particular task.

Syntax of a function looks something like this:

```
function myFunc () {
    // code
}

function bindFunc (parameter 1, parameter 2) {
    // code
}
```

↳ Function with parameters

↓ Here the parameters behave as local variables

binod func (7, 8) ⟹ Function Invocation

Function invocation is a way to use the code inside the function

A function can also return a value. The value is "returned" back to the caller

```
Const sum =(a,b) => {
                      let c = a+b;
                      return c;
                    }
```

↓ Another way to create & use the function ↑

⟹ Returns the sum

```
let y = sum (1, 3)
console.log(y)
```

→ Prints 4

Chapter 3 - Practice Set

1. Write a program to print the marks of a student in an object using for loop

obj = { harry : 98, rohan : 70, aakash : 7 }

2. Write the program in Q 1 using for in loop

3. Write a program to print "try again" until the user enters the correct number.

4. Write a function to find mean of 5 numbers.

Chapter 4 - Strings

Strings are used to store and manipulate text.
String can be created using the following syntax:

```
let name = "Harry" → Creates a string
name.length
    ↳ This property prints length of the string
```

Strings can also be created using single quotes

```
let name = 'Harry'
```

Template Literals
Template literals use backtics instead of quotes to define a string

```
let name = `Harry`
```

With template literals, it is possible to use both single as well as double quotes inside a string

```
let Sentence = `The name "is" Harry's`
               ↓ backtic   ↓ double quote   ↓ Single quote
```

We can insert variable directly in template literal. This is called string interpolation

```
let a = `This is ${name}` → Prints 'This is a Harry'
                 name is a variable
```

Escape Sequence Characters

If you try to print the following string, JavaScript will misunderstand it

```
let name = 'Adam D'Angelo'
```

We can use single quote escape sequence to solve the problem

```
let name = 'Adam D\'Angelo'
```

Similarly we can use \" inside a string with double quotes

Other escape sequence characters are as follows

- \n → Newline
- \t → Tab
- \r → Carriage Return

String properties and Methods

1. `let name = "Harry"`
   `name.length` → prints 5

2. `let name = "Harry"`
   `name.toUpperCase()` → prints HARRY

3. `let name = "Harry"`
   `name.toLowerCase()` → prints harry

4) let name = "Harry"
(0, 1, 2, 3, 4)
name.slice(2, 4) → prints rr
(from 2 to 4, 4 not included)

5) let name = "Harry"
name.slice(2) → prints rry
(from 2 to end)

6) let name = "Harry Bhai"
let newName = name.replace("Bhai", "Bhau")

7) let name1 = "Harry"
let name2 = "Naman"
let name3 = name1.concat(name2, "Yes")
↳ We can even use + operator

8) let name = " Harry "
let newName = name.trim()
↳ Removes whitespaces

Strings are immutable. Inorder to access the character at an index we use the following syntax

let name = "Harry"
name[0] → Prints H
name[1] → Prints a

Chapter 4 - Practice Set

1. What will the following print in JavaScript?

```
Console.log("har\"".length)
```

2. Explore the includes, startsWith & endsWith functions of a string

3. Write a program to convert a given string to lowercase

4. Extract the amount out of this string

"Please give Rs 1000"

5. Try to change 4th character of a given string. Were you able to do it?

Chapter 5 - Practice Set

1. Create an array of numbers and take input from the user to add numbers to this array.

2. Keep adding numbers to the array in (1) until 0 is added to the array.

3. Filter for numbers divisible by 10 from a given array.

4. Create an array of Square of given numbers.

5. Use reduce to calculate factorial of a given number from an array of first n natural numbers. (n being the number whose factorial needs to be calculated)

Chapter 5 - Arrays

Arrays are variables which can hold more than one value.

```
const fruits = ["banana", "apple", "grapes"]

const a1 = [7, "Harry", false]
```
↳ can be different types

Accessing Values

```
let numbers = [1, 2, 7, 9]

numbers[0] → 1
numbers[1] → 2
```

Finding the length

```
let numbers = [1, 7, 9, 21]
              ↙0  ↓1  ↓2  ↘3

numbers[0] → 1
numbers.length → 4
```

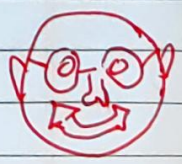
Changing the values

```
let numbers = [7, 2, 40, 9]

numbers[2] = 8
```
↳ "numbers" now becomes [7, 2, 8, 9]

Arrays are mutable

Arrays can be changed



In JavaScript, arrays are objects. The typeof operator on arrays returns object

```
Const n = [1, 7, 9]

typeof n → returns "object"
```

Arrays can hold many values under a single name

Array methods

There are some important array methods in JavaScript. Some of them are as follows:

1. toString() → Converts an array to a string of comma separated values

```
let n = [1, 7, 9]
n.toString() → 1, 7, 9
```

2. join() → joins all the array elements using a separator

```
let n = [7, 9, 13]
n.join("-") → 7-9-13
```

3. pop() → removes last element from the array

```
let n = [1, 2, 4]
n.pop() → updates the original array
          returns the popped value
```

4. push() → Adds a new element at the end of the array

```
let a = [7, 1, 2, 8]
a.push(9) → modifies the original array
          ↳ returns the new array length
```

5. shift() → Removes first element and returns it

6. unshift() → Adds element to the beginning. Returns new array length

7. delete → Array elements can be deleted using the delete operator

```
let d = [7, 8, 9, 10]
delete d[1]        → delete is an operator
```

8. Concat() → Used to join arrays to the given array

```
let a1 = [1, 2, 3]
let a2 = [4, 5, 6]
let a3 = [9, 8, 7]

a1.concat(a2, a3) → Returns [1, 2, 3, 4, 5, 6, 9, 8, 7]
```

↓
Returns a new array
Does not change existing arrays

9> Sort() → sort() method is used to sort an array alphabetically.

```
let a = [7, 9, 8]
a.sort()
```

↳ a changes to [7, 8, 9]
[modifies the original array]

Sort() takes an optional compare function. If this function is provided as the first argument, the sort() function will consider these values (the values returned from the compare function) as the basis of sorting.

10> splice() → splice can be used to add new items to an array

```
Const numbers = [1, 2, 3, 4, 5]
numbers.splice(2, 1, 23, 24)
```

Returns deleted items, modifies the Array
position to add
No of elements to remove
Elements to be added

11> slice() → slices out a piece from an array. It creates a new array

```
Const num = [1, 2, 3, 4]
num.slice(2)    → [3, 4]
num.slice(1, 3) → [2, 3]
```

12. reverse() → Reverses the elements in the source array.

Looping through Arrays
Arrays can be looped through using the classical JavaScript for loop or through some other methods discussed below

1. forEach loop → Calls a function, once for each array element

```
Const a = [1, 2, 3]
a.forEach ((value, index, array) => {
    // function logic
});
```

2. map() → Creates a new array by performing some operation on each array element.

```
Const a = [1, 2, 3]
a.map ((value, index, array) => {
    return value * value;
})
```

3. filter() → Filters ran array with values that passes a test. Creates a new array

```
Const a = [1, 2, 3, 4, 5]
a.filter (greater_than_5)
```

4. reduce method → Reduces an array to a single value

```
Const n = [1, 8, 7, 11]
let sum = numbers.reduce(add)
```

↳ A function

1+8+7+11

5. Array.from → Used to create an array from any other object

```
Array.from("Harry")
```

6. for... of → For-of loop can be used to get the values from an array

7. for ... in → for-in loop can be used to get the keys from an array.

Chapter 6 - JavaScript in the browser

JavaScript was initially created to make web pages alive. JS can be written right in a web page's HTML to make it interactive

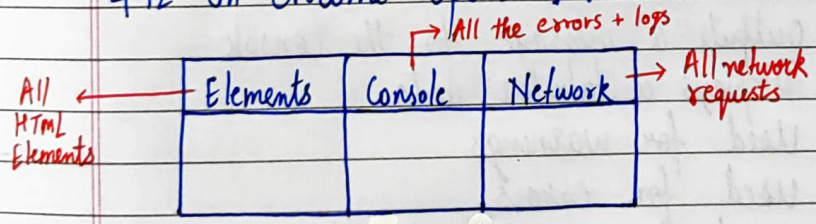
The browser has an embedded engine called the JavaScript engine or the Javascript runtime.

JavaScript's ability in the browser is very limited to protect the user's safety. For example a webpage on http://goofle.com cannot access http://codeswear.com and steal information from there.

Developer tools

Every browser has some developer tools which makes a developer's life a lot easier.

F12 on chrome opens Dev tools



We can also write JavaScript commands in the Console

The script tag

The script tag is used to insert JavaScript into an HTML page

The script tag can be used to insert external or internal scripts

```
<Script>
  alert("Hello")
</Script>
// or...
<Script Src="/js/thisone.js"></Script>
```

The benefit of a separate javascript file is that the browser will download it and store it in its cache

Console object methods
The console object has several methods, log being one of them. Some of them are as follows:

assert() → used to assert a condition
clear() → clears the console
log() → Outputs a message to the console
table() → Displays a tabular data
warn() → Used for warnings
error() → used for errors
info() → Used for special information

You will naturally remember some or all of these with time
Comprehensive list can be looked up on MDN

Interaction : alert, prompt and confirm

alert : Used to invoke a mini window with a msg.

```
alert("hello")
```

prompt : Used to take user input as string

```
inp = prompt("Hi", "No")
```

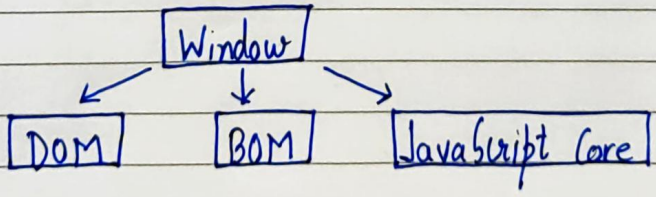
↳ optional default value

Confirm : shows a message and waits for the user to press ok or cancel. Returns true for ok and false for cancel.

The exact location & look is determined by the browser which is a limitation

Window object, BOM & DOM

We have the following when JavaScript runs in a browser



Window object represents browser window and provides methods to control it. It is a global object

classmate
Date
Page

Document Object Model (DOM)

Dom represents the page content as HTML

```
document.body  → Page body as Js object
```

```
document.body.style.background = "green"
```
↳ changes page background to green

Browser Object Model (BOM)

The Browser Object Model (BOM) represents additional objects provided by the browser (host environment) for working with everything except the document.

The functions alert / confirm / prompt are also a part of the BOM

```
location.href = "https://codewithharry.com"
```
↳ Redirect to another URL

Chapter 6 - Practice Set

1. Write a program using prompt function to take input of age as a value from the user and use alert to tell him if he can drive!

2. In Q1 use confirm to ask the user if he wants to see the prompt again

3. In the previous question, use console.error to log the error if the age entered is negative

4. Write a program to change the url to google.com (Redirection) if user enters a number greater than 4

5. Change the background of the page to yellow, red or any other color based on user input through prompt.

Chapter 7 - Walking the DOM

DOM tree refers to the HTML page where all the nodes are objects. There can be 3 main types of nodes in the DOM tree :

1) text nodes
2) element nodes
3) comment nodes

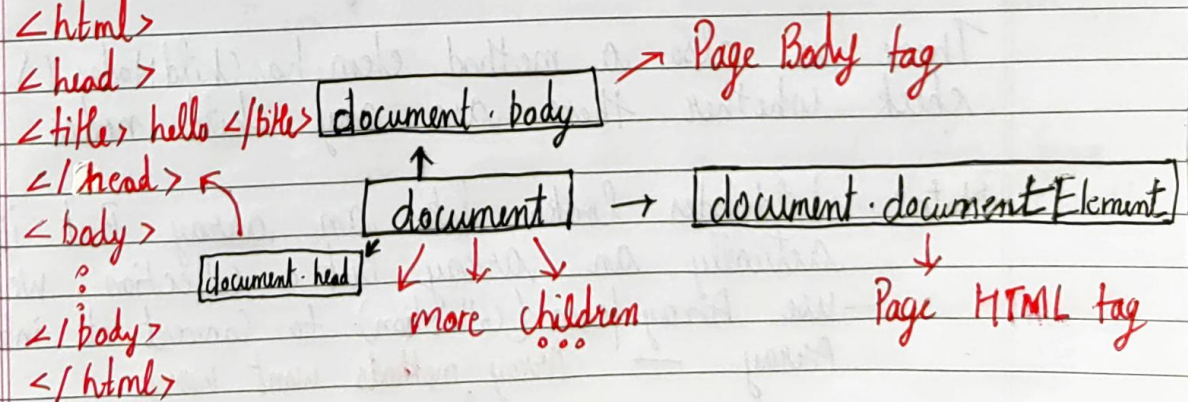
In an HTML page, <html> is at the root and <head> and <body> are its children, etc.

A text node is always a leaf of the tree

Auto Correction

If an erroneous HTML is encountered by the browser, it tends to correct it for example, if we put something after the body, it is automatically moved inside the body. Another example is <table> tag which must contain <tbody>

Walking the DOM



Note : document.body can sometimes be null if the javascript is written before the body tag.

Children of an element
Direct as well as deeply nested elements of an element are called its children

Child nodes → Elements that are direct children
For example head & body are children of <html>

Descendant nodes → All nested elements, children, their children and so on ...

firstChild, lastChild & childNodes

element.firstChild → first child element
element.lastChild → last child element
element.childNodes → All child nodes

Following is always true :

elem.childNodes[0] === elem.firstChild
elem.childNodes[elem.childNodes.length - 1] === elem.lastChild

There is also a method elem.hasChildNodes() to check whether there are any child nodes.

Note : childNodes looks like an array. But its not actually an array but a collection. We can use Array.from(collection) to convert it into an Array. → Array methods went work

Notes on DOM collections

→ They are read-only
→ They are live. elem.childNodes variable (reference) will automatically update if childNodes of elem is changed.
→ They are iterable using for... of loop

Siblings and the parent

Siblings are nodes that are children of the same parent.

→ For example: <head> and <body> are siblings. Siblings have same parent. In the above example its html

→ <body> is said to be the "next" or "right" sibling of <head>, <head> is said to be the "previous" ore "left" sibling of <body>

→ The next sibling is in nextSibling property, and the previous one in previousSibling.
The parent is available as parentNode.

```
alert( document.documentElement.parentNode ); //document
alert( document.documentElement.parentElement ); // null
```

Element only Navigation

Sometimes, we dont want text or comment nodes. Some links only take Element nodes into account. For example

document.previousElementSibling → Previous sibling which is an Element

document.nextElementSibling → next sibling (Element)

document.firstElementChild → first Element child

document.lastElementChild → last Element child.

Table links

Certain DOM elements may provide additional properties specific to their type for convenience.

Table element supports the following properties:

table.rows → collection of tr elements
table.caption → reference to <caption>
table.tHead → reference to <thead>
table.tFoot → reference to <tfoot>
table.tBodies → collection of <tbody> elements
tbody.rows → collection of <tr> inside

tr.cells → collection of td and th
tr.sectionRowIndex → index of tr inside enclosing element
tr.rowIndex → Row number starting from 0

td.cellIndex → no of cells inside enclosing <tr>

Quick Quiz: Print typeof document and typeof window in the console & see what it prints

Searching the DOM

DOM navigation properties are helpful when the elements are close to each other. If they are not close to each other, we have some more methods to search the DOM

→ **document.getElementById**
This method is used to get the element with a given "id" attribute

```
let span = document.getElementById('span')
span.style.color = "red"
```

→ **document.querySelectorAll**
Returns all elements inside an element matching the given CSS selector

→ **document.querySelector**
Returns the first element for the given CSS Selector.
A efficient version of elem.querySelectorAll(css)[0]

→ **document.getElementsByTagName**
Returns elements with the given tag name

→ **document.getElementsByClassName**
Returns elements that have the given CSS class

↳ Dont forget the "s" letter

→ **document.getElementsByName**
Searches elements by the name attribute.

matches, closest & contains methods

There are three important methods to search the DOM

1> elem.matches(css) → To check if element matches the given CSS selector

2> elem.closest(css) → To look for the nearest ancestor that matches the given CSS-selector. The elem itself is also checked

3> elemA.contains(elemB) → Returns true if elemB is inside elemA (a descendant of elemA) or when elemA == elemB

Chapter 7 - Practice Set

1. Create a navbar and change the color of its first element to red.

2. Create a table without tbody. Now use "View page source" button to check whether it has a tbody or not

3. Create an element with 3 children. Now change the color of first and last element to green.

4. Write a javascript code to change background of all <li> tags to cyan.

5. Which of the following is used to look for the farthest ancestor that matches a given CSS selector

(a) matches (b) closest (c) contains (d) none of these

Chapter 8 - Events & other DOM properties

Console.dir function
Console.log shows the element DOM tree
Console.dir shows the element as an object with its properties

tagName / nodeName
Used to read tag name of an element
tagName → only exists for Element nodes
nodeName → defined for any node (text, comment etc.)

innerHTML and outerHTML
The innerHTML property allows to get the HTML inside the element as a string.
↳ Valid for element nodes only

The outerHTML property contains the full HTML. innerHTML + the element itself.

innerHTML is valid only for element nodes. For other node types we can use nodeValue or data.

textContent
Provides access to the text inside the element: only text, minus all tags.

The hidden property
The "hidden" attribute and the DOM property specifies whether the element is visible or not.

```
<div hidden> I am hidden </div>

<div id = "element"> I can be hidden </div>

<script>
element.hidden = true;
</script>
```

Attribute methods

1. elem.hasAttribute(name) → Method to check for existence of an attribute
2. elem.getAttribute(name) → Method used to get the value of an attribute
3. elem.setAttribute(name, value) → Method used to set the value of an attribute.
4. elem.removeAttribute(name) → Method to remove the attribute from elem.
5. elem.attributes → Method to get the collection of all attributes

data-* attributes

We can always create custom attributes but the ones starting with "data-" are reserved for programmers use. They are available in a property named dataset.