📚 DOM (Document Object Model)

📌 What is the DOM?

- The **DOM** represents an HTML document as a **tree structure** of **nodes**.
- It enables **JavaScript** to interact with and modify web pages dynamically.

🔗 DOM Nodes Types

1. **Element Node**
2. **Text Node**
3. **Comment Node**
4. **Whitespace Node**

🌳 DOM Tree Structure

- `HTML` is the **root node**.
- `Head` and `Body` are child nodes.
- All elements inside form a hierarchical **tree-like structure**.

💡 JavaScript Power with DOM

JavaScript can:

- Change **HTML elements** and **attributes**
- Modify **CSS styles**
- Add, remove, or replace elements and attributes
- Handle and trigger **events**

🌐 DOM Standards (by W3C)

- **Core DOM** – For all document types
- **XML DOM** – For XML documents
- **HTML DOM** – For HTML documents

📦 HTML DOM Defines:

- HTML **elements as objects**
- Their **properties**
- Methods to **access/change** elements
- Events associated with them

🔍 Finding HTML Elements

Method	Description
`getElementById(id)`	Finds an element by ID
`getElementsByTagName(tag)`	Finds elements by tag
`getElementsByClassName(class)`	Finds elements by class
`querySelectorAll(css)`	Finds all matching elements using a CSS selector

📝 Changing HTML Elements

Content:

```
element.innerHTML = "New Content";
```

Attribute:

```
element.src = "new.jpg";
```

Style:

```
element.style.color = "blue";
```

Set Attribute:

```
element.setAttribute("class", "new-class");
```

➕ Creating & Removing Elements

Method	Description
`createElement()`	Create a new element
`appendChild()`	Add an element
`removeChild()`	Remove an element
`replaceChild(new, old)`	Replace an element
`document.write()`	Write to document directly (not recommended)

🧠 Event Handling

```
document.getElementById("btn").onclick = function() {
  alert("Clicked!");
};
```

📃 Common `document` Properties

Property	Description
`document.body`	Returns `<body>` element
`document.head`	Returns `<head>` element
`document.forms`	Returns all `<form>` elements
`document.images`	Returns all `<img>` elements
`document.links`	All `<a>` & `<area>` with `href`
`document.cookie`	Returns cookies
`document.URL`	Current page URL
`document.title`	Title of document
`document.readyState`	Loading state

🧪 Examples

Get Element by ID and Change Content

```
<p id="p1">Hello World!</p>
<script>
  document.getElementById("p1").innerHTML = "New text!";
</script>
```

Change `src` of Image

```
<img id="myImage" src="smiley.gif">
<script>
  document.getElementById("myImage").src = "landscape.jpg";
</script>
```

Dynamic Content

```
<p id="demo"></p>
<script>
  document.getElementById("demo").innerHTML = "Date: " + Date();
</script>
```

🧰 Useful Concepts Checklist

1. ☑ Accessing Elements
2. ☑ Modifying Content & Attributes
3. ☑ Creating & Appending Elements
4. ☑ Handling Events
5. ☑ DOM Traversal
6. ☑ Styling with JS
7. ☑ Dynamic Content with JS
8. ☑ Form & Input Validation

🧭 DOM Auto-Correction

Browsers often auto-correct HTML errors:

```
<span>This is me</div>
```

Automatically corrected as:

```
<span>This is me</span>
```

📌 Quick Summary

- Use the DOM to control HTML documents with JavaScript.
- Access and manipulate elements using `getElementById`, `querySelector`, etc.
- Modify inner content, styles, and attributes.
- Create dynamic web pages through event handling and real-time changes.

🧠 JavaScript HTML DOM – Part 2: Animation, Events, and Event Listeners

---

🎬 HTML DOM Animation

📃 Basic Setup

To animate with JavaScript, start with a simple HTML structure:

```html
<!DOCTYPE html>
<html>
<body>

<h1>My First JavaScript Animation</h1>
<div id="animation">My animation will go here</div>

</body>
</html>
```

📦 Create an Animation Container

All animations should be placed relative to a container element.

```html
<div id="container">
  <div id="animate">My animation will go here</div>
</div>
```

🎨 Style the Elements

```css
#container {
  width: 400px;
  height: 400px;
  position: relative;
  background: yellow;
}
#animate {
  width: 50px;
  height: 50px;
  position: absolute;
  background: red;
}
```

🔁 Animation Logic in JavaScript

JavaScript animations are achieved by gradually updating styles using `setInterval`.

```javascript
let id = setInterval(frame, 5);
```

```
function frame() {
  if (/* condition */) {
    clearInterval(id);
  } else {
    // update element style
  }
}
```

☑ Full Working Animation

```
function myMove() {
  let id = null;
  const elem = document.getElementById("animate");
  let pos = 0;
  clearInterval(id);
  id = setInterval(frame, 5);

  function frame() {
    if (pos === 350) {
      clearInterval(id);
    } else {
      pos++;
      elem.style.top = pos + "px";
      elem.style.left = pos + "px";
    }
  }
}
```

🖱 HTML DOM Event Handling

💡 What are Events?

Events are actions like:

- Mouse clicks
- Page load
- Input changes
- Key presses
- Mouse movements

🎯 Reacting to Events

Inline Event Example:

```
<h1 onclick="this.innerHTML='Oops!'">Click me!</h1>
```

Calling Function on Event:

```
<h1 onclick="changeText(this)">Click me!</h1>
<script>
function changeText(el) {
  el.innerHTML = "Ooops!";
}
</script>
```

⭕ HTML DOM Event Attributes

Assign events directly in HTML:

```
<button onclick="displayDate()">Try it</button>
```

🔁 Assigning Events via DOM

```
<script>
document.getElementById("myBtn").onclick = displayDate;
</script>
```

📦 Common HTML Events

Event	Description
`onclick`	User clicks an element
`onload`	Page has loaded
`onchange`	Input field value changed
`onmouseover`	Mouse hovers over an element
`onmouseout`	Mouse leaves an element
`onmousedown`	Mouse button is pressed down
`onmouseup`	Mouse button is released
`onfocus`	Input element gets focus

🧩 JavaScript HTML DOM `addEventListener()`

📌 Syntax

```
element.addEventListener(event, function, useCapture);
```

☑ Benefits

- Doesn't overwrite existing handlers
- Multiple listeners possible
- Cleaner separation of logic and HTML

🧪 Examples

Basic usage:

```
element.addEventListener("click", function () {
  alert("Hello World!");
});
```

Using external function:

```
element.addEventListener("click", myFunction);
function myFunction() {
  alert("Hello World!");
}
```

Multiple handlers:

```
element.addEventListener("click", firstFunction);
element.addEventListener("click", secondFunction);
```

Window object:

```
window.addEventListener("resize", function () {
  document.getElementById("demo").innerHTML = "Window resized!";
});
```

🎯 Passing Parameters

```
element.addEventListener("click", function() {
  myFunction(param1, param2);
});
```

🔄 Bubbling vs Capturing

Type	Description
Bubbling	Inner element fires first (default)
Capturing	Outer element fires first

```
element.addEventListener("click", myFunction, true); // capturing
```

❌ Removing Event Listeners

```
element.removeEventListener("click", myFunction);
```

🌳 HTML DOM Navigation (Intro)

The DOM is a tree of **nodes**:

- Document → Document node
- HTML elements → Element nodes
- Text → Text nodes
- Attributes → Attribute nodes *(deprecated)*
- Comments → Comment nodes

You can navigate using:

- `parentNode`
- `childNodes`
- `firstChild`
- `lastChild`
- `nextSibling`
- `previousSibling`

🖼️ DOM Tree Visual

📓 JavaScript DOM Basics & Traversal Notes

🧱 HTML is Converted into JS Objects

Every node in HTML becomes a JavaScript object:

- 📄 **Text Node**: Represents text inside elements.
- 🏷️ **Element Node**: Represents HTML tags.
- 💬 **Comment Node**: Represents `<!-- comments -->`.

👉 **HTML Auto-Correction**: Browsers automatically fix incorrect HTML structures.

🌳 Walking & Traversing the DOM (DOM Tree)

- `document.body` → `<body>` tag
- `document.documentElement` → `<html>` tag
- `document.head` → `<head>` tag
- `document.title` → returns page title as string

⚠️ `document.body` can be `null` if JS is placed before `<body>` is loaded.

👶 Child, Descendant & Sibling Nodes

👶 Child Nodes

- `element.childNodes` → all types of direct children (text, element, etc.)
- `element.children` → **only element** children

🧬 Descendant Nodes

All nested nodes inside an element (deep traversal).

👯 Siblings

Nodes with the same parent.

- `nextSibling` / `previousSibling` → includes **text nodes** too
- `nextElementSibling` / `previousElementSibling` → **only elements**

🔍 Child Node Access

```
element.firstChild
element.lastChild
element.hasChildNodes()
Array.from(element.childNodes) // Convert collection to real array
```

☑️ Always true:

```
element.firstChild === element.childNodes[0]
element.lastChild === element.childNodes[element.childNodes.length - 1]
```

👪 Parent Node Access

- `element.parentNode` → works for any node
- `element.parentElement` → only returns if parent is an **element**, otherwise `null`

🧭 Element-Only Navigation

```
element.firstElementChild
element.lastElementChild
element.nextElementSibling
element.previousElementSibling
```

📋 Special DOM Collections

- Read-only
- Live-updated (dynamic)
- Iterable using `for...of`

🍽 Table-Specific Properties

```
table.rows              // <tr> collection
table.caption           // <caption> element
table.tHead / tFoot     // <thead> / <tfoot>
table.tBodies           // Collection of <tbody>
tbody.rows              // All <tr> in that section
tr.cells                // <td> and <th> cells
tr.rowIndex             // Row index from top
tr.sectionRowIndex      // Row index within section
td.cellIndex            // Index of <td> in row
```

🔍 Searching the DOM

```
document.getElementById("id")
document.getElementsByClassName("class") // HTMLCollection
document.getElementsByTagName("tag")
document.getElementsByName("name")
```

```
document.querySelector(".class")         // first match
document.querySelectorAll(".class")      // NodeList of all
```

🧠 Remember:

- No "s" → Returns one element
- With "s" → Returns all

☑️ `matches`, `closest`, and `contains`

```
elem.matches(".class")     // Does it match?
elem.closest(".wrapper")   // Nearest matching ancestor
elem.contains(otherElem)   // Is otherElem a child or same?
```

🧪 DOM Practice Set (Chapter 7)

🟤 Q1: Make navbar's first element red

```
document.getElementsByTagName("nav")[0].firstElementChild.style.color = "red";
```

🟣 Q2: Make a table green

```
document.getElementsByTagName("table")[0].style.background = "green";
```

⬛ Q3: Make first & last children of an element green

```
let nav = document.getElementsByTagName("nav")[0];
nav.firstElementChild.style.color = "green";
nav.lastElementChild.style.color = "green";
```

🟦 Q4: Make all `<li>` backgrounds cyan

```
Array.from(document.getElementsByTagName("li")).forEach((el) => {
  el.style.background = "cyan";
});
```

❓ Q5: Find farthest ancestor matching selector

Answer: `None of These` (Closest finds nearest, not farthest)

📚 Quick Q&A

Q: Access element by ID?

```
document.getElementById("myElement");
```

Q: Access by class?

```
document.getElementsByClassName("myClass");
```

Q: Access by tag?

```
document.getElementsByTagName("h1");
```

Q: Access using CSS selector?

```
document.querySelector(".mySelector");
```

Q: Change content?

```
myDiv.innerHTML = "Hello, JavaScript!";
```

Q: Change attribute?

```
myImage.setAttribute("src", "new_image.jpg");
```

Q: Change style?

```
myDiv.style.backgroundColor = "red";
```

Q: Add class?

```
myDiv.classList.add("highlight");
```

Q: Create and append element?

```
const newEl = document.createElement("p");
newEl.innerHTML = "I'm new!";
document.body.appendChild(newEl);
```

Q: Check if element contains another?

```
parent.contains(child); // true or false
```
