

JavaScript DOM Lab Manual

Department of Computer Science and IT

Course: Web Programming Lab

Prepared by: Muhammad Haroon

Date: November 6, 2025

Objective

To understand and implement the **Document Object Model (DOM)** in JavaScript — manipulating HTML elements, attributes, and content dynamically using DOM properties and methods.

Introduction to DOM

The **DOM (Document Object Model)** represents the structure of a web page as a tree of objects. Each HTML element (like `<p>`, `<div>`, `<button>`) is represented as a node in this tree.

JavaScript can access, modify, add, or delete elements in the DOM.

Basic DOM Structure

Example HTML:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>DOM Example</title>
5   </head>
6   <body>
7     <h1 id="title">Welcome!</h1>
8     <p class="message">This is a paragraph.</p>
9     <div id="container"></div>
10    <button id="btn">Click Me</button>
11    <script src="script.js"></script>
12  </body>
13 </html>
```

Commonly Used DOM Properties & Methods

1. `document.getElementById()`

Description: Returns the element that has the specified `id` attribute.

```
1 let heading = document.getElementById("title");
2 heading.style.color = "blue";
```

2. `document.getElementsByClassName()`

Description: Returns a collection of all elements with the given class name.

```
1 let messages = document.getElementsByClassName("message");
2 messages[0].innerText = "Class Accessed Successfully!";
```

3. document.getElementsByTagName()

Description: Returns all elements with the specified tag name.

```
1 let paragraphs = document.getElementsByTagName("p");
2 paragraphs[0].style.fontWeight = "bold";
```

4. document.querySelector() / document.querySelectorAll()

Description: Selects the first element (or all matching elements) using a CSS selector.

```
1 let firstPara = document.querySelector(".message");
2 firstPara.style.color = "green";
3
4 let allParas = document.querySelectorAll("p");
5 allParas.forEach(p => p.style.fontSize = "18px");
```

5. innerHTML

Description: Sets or gets the HTML content inside an element.

```
1 document.getElementById("container").innerHTML = "<b>New HTML Added!</b>";
```

6. innerText / textContent

Description:

- **innerText:** Returns the visible text of an element.
- **textContent:** Returns all text, even hidden elements.

```
1 let text = document.getElementById("title").innerText;
2 console.log(text);
```

7. setAttribute() / getAttribute()

Description: Used to set or get the value of an attribute in an element.

```
1 let btn = document.getElementById("btn");
2 btn.setAttribute("title", "Click this button");
3 console.log(btn.getAttribute("id"));
```

8. style Property

Description: Used to modify CSS styles directly.

```
1 let heading = document.getElementById("title");
2 heading.style.backgroundColor = "yellow";
3 heading.style.padding = "10px";
```

9. createElement() and createTextNode()

Description: Creates new HTML elements and text nodes dynamically.

```
1 let newDiv = document.createElement("div");
2 let textNode = document.createTextNode("This is a new Div!");
3 newDiv.appendChild(textNode);
4 document.body.appendChild(newDiv);
```

10. appendChild()

Description: Adds a node as the last child of a parent element.

```
1 let newPara = document.createElement("p");
2 newPara.textContent = "This paragraph was added!";
3 document.getElementById("container").appendChild(newPara);
```

11. append() / prepend()

Description:

- `append()` adds nodes or text at the end.
- `prepend()` adds them at the beginning.

```
1 let div = document.getElementById("container");
2 div.append("Appended text!");
3 div.prepend("Prepended text!");
```

12. removeChild() / remove()

Description: Removes a child node or an element itself.

```
1 let para = document.querySelector("p");
2 para.remove(); // removes the element itself
```

13. parentNode / parentElement

Description: Returns the parent of a specified node.

```
1 let button = document.getElementById("btn");
2 console.log(button.parentNode);
```

14. childNodes / children

Description:

- **childNodes**: Returns all child nodes (including text).
- **children**: Returns only element nodes.

```
1 let container = document.getElementById("container");
2 console.log(container.children);
```

15. firstChild / lastChild

Description: Gives access to the first or last child of an element.

```
1 let first = container.firstChild;
2 let last = container.lastChild;
3 console.log(first, last);
```

16. nextSibling / previousSibling

Description: Accesses the next or previous node in the DOM tree.

```
1 let para = document.querySelector("p");
2 console.log(para.nextSibling);
```

17. classList

Description: Provides methods to add, remove, toggle, or check CSS classes.

```
1 let heading = document.getElementById("title");
2 heading.classList.add("highlight");
3 heading.classList.toggle("active");
```

18. insertBefore()

Description: Inserts a new node before a specified child node.

```
1 let newItem = document.createElement("p");
2 newItem.textContent = "Inserted paragraph!";
3 let container = document.getElementById("container");
4 container.insertBefore(newItem, container.firstChild);
```

19. replaceChild()

Description: Replaces one child node with another.

```
1 let newPara = document.createElement("p");
2 newPara.textContent = "I replaced the old one!";
3 container.replaceChild(newPara, container.children[0]);
```

20. DOM Events (onclick, addEventListener())

Description: Used to add interactivity (click, hover, input, etc.).

```
1 document.getElementById("btn").addEventListener("click", function() {
2     alert("Button clicked!");
3});
```

Lab Exercises

Exercise 1: Create and Append Elements

- Create a new <div> dynamically.
- Add some text to it.
- Append it to the body.

Exercise 2: Modify CSS Using JavaScript

Change the color and font size of an element when a button is clicked.

Exercise 3: DOM Traversal

Print all child elements of a specific container in the console.

Exercise 4: Replace and Remove Elements

Replace one paragraph with another dynamically and remove a node after 3 seconds.

Exercise 5: Interactive To-Do List

Add input and “Add” button. Add new list items dynamically using `appendChild()`.

Conclusion

Through this lab, you learned:

- DOM access and traversal methods
- Creating, appending, and removing nodes
- Manipulating attributes and styles
- Handling user interactions with events

DOM manipulation is the backbone of dynamic, interactive web applications.