Advanced JavaScript DOM Lab Manual

Department of Computer Science and IT

Course: Web Programming Lab

Prepared by: Muhammad Haroon

Date: November 6, 2025

Objective

To gain hands-on experience in manipulating the Document Object Model (DOM) in JavaScript through advanced concepts and real-world projects. The aim is to dynamically create, modify, and manage web content using JavaScript.

Introduction to DOM

The Document Object Model (DOM) represents an HTML or XML document as a structured tree of nodes. Each HTML tag becomes an object that JavaScript can manipulate. Using DOM methods, developers can change the content, style, structure, and behavior of a webpage dynamically.

Advanced concepts introduced:

- Event bubbling and delegation
- Dynamic element creation and removal
- Local Storage manipulation
- Working with form validation
- Creating dynamic interfaces using DOM

Advanced DOM Topics

1. Event Delegation

Description: A technique where a single event listener is added to a parent element to handle events on its children using `event.target`.

```
document.getElementById("list").addEventListener("click", function(e) {
    if (e.target.tagName === "LI") {
        e.target.classList.toggle("completed");
    }
});
```

2. Traversing the DOM Tree

Properties: `parentNode, children, nextElementSibling, previousElementSibling, firstElementChild, lastElementChild`

```
let ul = document.querySelector("ul");
console.log(ul.parentNode);
console.log(ul.firstElementChild.textContent);
```

3. Cloning Elements

Method: `cloneNode(deep)`

```
let item = document.querySelector("li");
let copy = item.cloneNode(true);
document.querySelector("ul").appendChild(copy);
```

4. Working with Local Storage

```
localStorage.setItem("username", "Haroon");
let name = localStorage.getItem("username");
console.log("Welcome " + name);
```

5. Dynamic Class Management

```
let box = document.getElementById("box");
box.classList.add("active");
box.classList.remove("hidden");
box.classList.toggle("highlight");
```

6. Creating Elements from Templates

```
let template = ‘<li class="item">New Item</li>‘;
document.querySelector("ul").insertAdjacentHTML("beforeend", template);
```

Mini Projects Using DOM

Project 1: Interactive To-Do List

Objective: Build a dynamic to-do list using `createElement()`, `appendChild()`, and `remove()`.

```
<input type="text" id="taskInput" placeholder="Enter task">
<button id="addBtn">Add Task</button>
<ul id="taskList"></ul>
```

```javascript
const input = document.getElementById("taskInput");
const list = document.getElementById("taskList");
document.getElementById("addBtn").addEventListener("click", () => {
    if (input.value.trim() !== "") {
        let li = document.createElement("li");
        li.textContent = input.value;
        li.addEventListener("click", () => li.remove());
        list.appendChild(li);
        input.value = "";
    }
});
```

Key Concepts:

- Node creation and appending
- Event handling
- DOM cleanup and state management

Project 2: Live Form Validation

Objective: Validate user input in real time using DOM events and class toggling.

```html
<form id="userForm">
  <input type="email" id="email" placeholder="Enter your email">
  <small id="msg"></small>
</form>
```

```javascript
const email = document.getElementById("email");
const msg = document.getElementById("msg");

email.addEventListener("input", () => {
    const pattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (pattern.test(email.value)) {
        msg.textContent = "Valid Email       ";
        msg.style.color = "green";
    } else {
        msg.textContent = "Invalid Email    ";
        msg.style.color = "red";
    }
});
```

Key Concepts:

- `input` event handling
- Regular expressions

- Real-time DOM updates

Project 3: Image Gallery with Lightbox

Objective: Create an interactive gallery using DOM events and dynamic image loading.

```html
<div id="gallery">
  <img src="img1.jpg" width="100">
  <img src="img2.jpg" width="100">
</div>
<div id="lightbox" style="display:none;">
  <img id="lightImage">
</div>
```

```javascript
let gallery = document.getElementById("gallery");
let lightbox = document.getElementById("lightbox");
let lightImage = document.getElementById("lightImage");

gallery.addEventListener("click", e => {
  if (e.target.tagName === "IMG") {
    lightImage.src = e.target.src;
    lightbox.style.display = "block";
  }
});
lightbox.addEventListener("click", () => {
  lightbox.style.display = "none";
});
```

Key Concepts:

- Event delegation
- Dynamic content switching
- Style manipulation

Project 4: Dark/Light Theme Switcher

Objective: Toggle between light and dark mode using DOM and Local Storage.

```html
<button id="toggleTheme">Switch Theme</button>
\div id="content">Welcome to My Page!</div>
```

```javascript
const toggle = document.getElementById("toggleTheme");
const body = document.body;

toggle.addEventListener("click", () => {
    body.classList.toggle("dark");
    localStorage.setItem("theme", body.classList.contains("dark") ? "
        dark" : "light");
});

window.onload = () => {
    if (localStorage.getItem("theme") === "dark") {
        body.classList.add("dark");
    }
};
```

Key Concepts:

- `classList.toggle()`
- Persistent data with Local Storage
- Theming and user preferences

Project 5: Dynamic Table Generator

Objective: Dynamically create a table based on user input.

```
<input type="number" id="rows" placeholder="Rows">
<input type="number" id="cols" placeholder="Columns">
<button id="generate">Generate Table</button>
<div id="tableContainer"></div>
```

```
document.getElementById("generate").addEventListener("click", () => {
  let rows = document.getElementById("rows").value;
  let cols = document.getElementById("cols").value;
  let table = document.createElement("table");
  table.border = "1";

  for (let i = 0; i < rows; i++) {
    let tr = document.createElement("tr");
    for (let j = 0; j < cols; j++) {
      let td = document.createElement("td");
      td.textContent = `R${i+1}C${j+1}`;
      tr.appendChild(td);
    }
    table.appendChild(tr);
  }

  let container = document.getElementById("tableContainer");
  container.innerHTML = "";
  container.appendChild(table);
});
```

Key Concepts:

- Nested loops with DOM creation
- Input handling
- Dynamic content generation

Conclusion

In this advanced DOM lab manual, students explored:

- DOM event handling and dynamic manipulation
- Advanced node operations (insert, clone, replace)
- Local Storage and persistent UI states
- Real-world applications with interactive interfaces

These techniques form the foundation for building dynamic, user-responsive web applications and are essential for modern front-end development.