Document Object Model (DOM) - A Comprehensive Guide

Introduction to the DOM

The **Document Object Model (DOM)** is a programming interface for web documents. It represents a webpage as a structured tree of objects, allowing JavaScript to manipulate the content, structure, and styling dynamically.

Understanding the DOM Tree Structure

An HTML document is structured like a tree with **nodes** representing elements, attributes, and text content.

Example of a DOM Tree:

HTML Code

```html
<!DOCTYPE html>
<html>
<head>
    <title>DOM Example</title>
</head>
<body>
    <h1 id="title">Hello World</h1>
    <p class="description">This is a sample paragraph.</p>
</body>
</html>
```

Corresponding DOM Representation

```
Document
  html
     head
        title ("DOM Example")
     body
         h1#title ("Hello World")
         p.description ("This is a sample paragraph.")
```

Selecting Elements in the DOM

JavaScript provides various methods to select elements:

Selecting by ID

```javascript
document.getElementById("title");
```

Selecting by Class Name

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

Selecting by Tag Name

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

Using Query Selector

```
document.querySelector("#title"); // Selects the first matching element
document.querySelectorAll(".description"); // Selects all matching elements
```

Manipulating the DOM

Once an element is selected, we can modify its properties.

Changing Text Content

```
let title = document.getElementById("title");
title.textContent = "DOM Manipulation";
```

Modifying HTML Content

```
let para = document.querySelector(".description");
para.innerHTML = "<strong>Updated Content</strong>";
```

Changing Styles

```
title.style.color = "red";
title.style.fontSize = "24px";
```

Adding and Removing Classes

```
title.classList.add("highlight"); // Adds a class
title.classList.remove("highlight"); // Removes a class
```

Creating and Appending Elements

We can create new elements dynamically and append them to the DOM.

```
let newElement = document.createElement("div");
newElement.textContent = "I am a new div!";
document.body.appendChild(newElement);
```

Event Handling

Events allow interaction with users, like clicking buttons or submitting forms.

```
let button = document.getElementById("myButton");
button.addEventListener("click", function() {
    alert("Button Clicked!");
});
```

Traversing the DOM

Navigating through elements:

```
let parent = document.getElementById("title").parentNode;
let children = document.body.children;
let nextElement = document.getElementById("title").nextElementSibling;
```

Final Project: Interactive To-Do List

HTML

```
<!DOCTYPE html>
<html>
<head>
    <title>To-Do List</title>
    <style>
        .completed { text-decoration: line-through; color: gray; }
    </style>
</head>
<body>
    <h1>To-Do List</h1>
    <input type="text" id="taskInput" placeholder="Enter a task">
    <button id="addTask">Add Task</button>
    <ul id="taskList"></ul>
    <script src="script.js"></script>
</body>
</html>
```

JavaScript (script.js)

```
document.getElementById("addTask").addEventListener("click", function() {
    let taskInput = document.getElementById("taskInput");
    let taskText = taskInput.value.trim();

    if (taskText === "") return;

    let li = document.createElement("li");
    li.textContent = taskText;
```

```
    li.addEventListener("click", function() {
        li.classList.toggle("completed");
    });

    document.getElementById("taskList").appendChild(li);
    taskInput.value = "";
});
```

Search Functionality for a Table of Products

HTML

```
<input type="text" id="searchInput" placeholder="Search products...">
<table id="productTable">
    <tr>
        <th>Product Name</th>
    </tr>
    <tr>
        <td>Apple</td>
    </tr>
    <tr>
        <td>Banana</td>
    </tr>
    <tr>
        <td>Carrot</td>
    </tr>
</table>
```

JavaScript (script.js)

```
document.getElementById("searchInput").addEventListener("keyup", function() {
    let filter = this.value.toLowerCase();
    let rows = document.querySelectorAll("#productTable tr");

    rows.forEach((row, index) => {
        if (index === 0) return; // Skip header row
        let cell = row.getElementsByTagName("td")[0];
        if (cell) {
            let text = cell.textContent.toLowerCase();
            row.style.display = text.includes(filter) ? "" : "none";
        }
    });
});
```

Conclusion

The DOM provides a powerful way to dynamically manipulate HTML content, styles, and interactions. By mastering the DOM, developers can create engaging web applications with JavaScript, including interactive search functionalities.