**Experiment No:** 12

**Experiment Name:** DOM Interaction and Manipulation in JavaScript.

**Theory:** When a web page loads, the browser reads the HTML code and creates a tree-like structure called the DOM. Each HTML element (such as headings, paragraphs, and buttons) becomes an object in this tree. JavaScript can access and manipulate this DOM tree. It can locate elements, modify their content and style, or even add new elements. This process enables web pages to become interactive. For example, when a button is clicked or a form is filled out, JavaScript can respond by updating the page's display or sending data to the server. In summary, the DOM acts as a bridge between the static HTML content of a web page and the dynamic functionality of JavaScript. It allows JavaScript to read and modify the web page, making websites interactive and responsive.

**Selecting Elements:**

❖ getElementById() → select an element by id.

❖ getElementsByTagName() → select elements by a tag name.

❖ getElementsByClassName() → select elements by class name.

❖ querySelector() → select single elements by a CSS selector.

❖ querySelectorAll() → select all elements by the same CSS selectors.

**Manipulating Elements:**

❖ createElement() – create a new element.

❖ appendChild() – append child node to a specific parent node.

❖ textContent – get and set the text content of a node.

❖ innerHTML – get and set the HTML content of an element.

❖ appendChild() – insert a node after the last child node of a parent node.

❖ removeChild() – remove child elements of a node.

**Element Attributes:**

❖ setAttribute() → set the value of a specified attribute on an element.

❖ getAttribute() → get the value of an attribute on an element.

❖ removeAttribute() → remove an attribute from a specified element.

❖ hasAttribute() → check if an element has a specified attribute or not.

**Working with Events using addEventListener():**

❖ change

→ When an HTML element has been changed

❖ click

→ When the user clicks an HTML element

❖ mouseover

→ When the user moves the mouse over an HTML element

❖ keydown

→ When the user pushes a keyboard key

❖ submit

→ When the user submit a form element

❖ load

→ When the browser has finished loading the page

**Lab Task:** Develop the following interface with HTML and CSS, and then write a JavaScript program to create a comment section where previously added comments are displayed and new comments can be added by clicking a button.

**Source Code:**

Inside int.html

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>Comment Section</title>

    <link rel="stylesheet" href="int.css" />

  </head>

  <body>

    <div class="comment-section">

      <h2>Comments</h2>

      <div id="comments-container"></div>

      <textarea

        id="comment-input"

        placeholder="Write your comment..."

      ></textarea>

      <button id="add-comment-btn">Add Comment</button>

    </div>

    <script src="int.js"></script>

  </body>

</html>

Inside int.css

body {

  font-family: Arial, sans-serif;

  background-color: #f4f4f4;

  display: flex;

  justify-content: center;

  align-items: center;

  height: 100vh;

  margin: 0;

}

.comment-section {

  background-color: #fff;

  padding: 20px;

  border-radius: 5px;

  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

  width: 400px;

}

.comment-section h2 {

  margin-top: 0;

}

#comments-container {

  margin-bottom: 20px;

  max-height: 200px;

  overflow-y: auto;

  padding-right: 10px;

}

.comment {

  background-color: #f9f9f9;

  border: 1px solid #ddd;

  padding: 10px;

  border-radius: 3px;

  margin-bottom: 10px;

}

#comment-input {

  width: 100%;

  padding: 10px;

  margin-bottom: 10px;

  border: 1px solid #ddd;

  border-radius: 3px;

}

#add-comment-btn {

  padding: 10px 20px;

  border: none;

  background-color: #5cb85c;

  color: white;

  cursor: pointer;

  border-radius: 3px;

}

#add-comment-btn:hover {

  background-color: #4cae4c;

}

Inside int.js

var addCommentBtn = document.getElementById("add-comment-btn");

var commentInput = document.getElementById("comment-input");

var commentsContainer = document.getElementById("comments-container");

addCommentBtn.addEventListener("click", function () {

  var commentText = commentInput.value;

  if (commentText !== "") {

    var newComment = document.createElement("div");

    newComment.className = "comment";

    newComment.innerText = commentText;

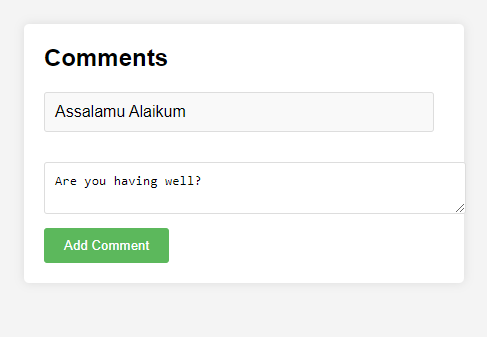
    commentsContainer.appendChild(newComment);

    commentInput.value = "";

  }

});

**Output:**

****

**Discussion:** In today’s lab report we got knew about the DOM Interaction and Manipulation in JavaScript.When a web page loads, the browser reads the HTML code and creates a tree-like structure called the DOM. Each HTML element (such as headings, paragraphs, and buttons) becomes an object in this tree. JavaScript can access and manipulate this DOM tree. It can locate elements, modify their content and style, or even add new elements. At last, the DOM acts as a bridge between the static HTML content of a web page and the dynamic functionality of JavaScript. It allows JavaScript to read and modify the web page, making websites interactive and responsive.