**Tasks:**

**a.What is HTML DOM? (0.5 points)**

HTML DOM stands for Document Object Model. It is a programming interface for web documents. The DOM represents the structure of a document as a tree of objects, where each object corresponds to a part of the document, such as elements, attributes, and text.

**b. What JavaScript can do to web pages by utilizing HTML DOM? (0.5 points)**

JavaScript can dynamically manipulate the content and structure of web pages by utilizing HTML DOM. It can update, delete, or add new elements, change styles, handle events, and modify attributes, providing a way to create interactive and dynamic web pages.

**c. List and briefly explain at least three different kinds of HTML Node types? (0.5 points)**

Element Nodes: Represent HTML elements like <div>, <p>, or <a>.

Text Nodes: Contain the actual text content within an element.

Attribute Nodes: Represent attributes of elements, such as id or class.

**d. Criticize HTML DOM. (0.5 points)**

One criticism of HTML DOM is that it can become slow and inefficient when dealing with large and complex web pages. Manipulating the DOM Frequently can lead to performance issues. Additionally, the DOM can be verbose, making code harder to read and maintain.

**2.**  **List and explain different ways to attach event handlers in JavaScript. What are the pros and cons of each of them? (1 point)**

**Attaching Event Handlers in JavaScript:**

There are different ways to attach event handlers in JavaScript, each with its pros and cons.

**Inline HTML Event Handlers:**

Pros: Simple and straightforward.

Cons: Mixing HTML with JavaScript is not a good practice; it can make code harder to maintain and less organized.

**Traditional DOM Event Handlers:**

Pros: Better separation of HTML and JavaScript.

Cons: Still involves mixing JavaScript with HTML, and managing multiple event handlers can be challenging.

**DOM Level 2 Event Handlers:**

Pros: Improved separation of concerns allows attaching multiple handlers.

Cons: Older browsers may not support it fully.

**addEventListener Method:**

Pros: Clean separation of HTML and JavaScript, supports multiple handlers.

Cons: Involves more code than inline handlers, and older browsers may not support it.

**jQuery Event Handling:**

Pros: Simplifies event handling, ensures cross-browser compatibility.

Cons: Adds an external dependency, which might affect page load times.

We should choose the method based on the project's needs, considering factors like code maintainability, performance, and browser compatibility.

**3.      Create an application that makes it possible to add messages on the web page. The web page displays a text area into which a new message can be written. The web page also displays a button “Add a new message!” that can be used to add a new message on the page. The message is added as a new paragraph on the page below the text area and the button when the button is clicked. The new message doesn’t replace the old ones already on the page. (2 points)**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Message Board</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            text-align: center;

            margin: 20px;

            background-color: cornflowerblue;

        }

        textarea {

            width: 300px;

            height: 100px;

        }

        button {

            margin-top: 20px;

            border-radius: 4px;

        }

        #messages {

            margin-top: 20px;

        }

    </style>

</head>

<body>

    <h1>Message Board</h1>

    <textarea id="messageInput" placeholder="Type your message"></textarea><br>

    <button onclick="addMessage()">Add a new message!</button>

    <div id="messages"></div>

    <script src="script.js"></script>

</body>

</html>

**script.js**

function addMessage() {

    // Get the message from the text area

    var message = document.getElementById("messageInput").value;

    // Check if the message is not empty

    if (message.trim() !== "") {

        // Create a new paragraph element

        var paragraph = document.createElement("p");

        // Create a text node with the message

        var textNode = document.createTextNode(message);

        // Append the text node to the paragraph

        paragraph.appendChild(textNode);

        // Get the messages container and append the new paragraph

        var messagesContainer = document.getElementById("messages");

        messagesContainer.appendChild(paragraph);

        // Clear the text area for the next message

        document.getElementById("messageInput").value = "";

    }

}

A screenshot of a message board

Description automatically generated

**4.      Create a page with 3 buttons. Layout is not important. When button 1 is pressed, background of a page should change to blue, button 2 changes it to gray and button 3 resets background to white (2 pts)**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Background Changer</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <button onclick="changeBackground('blue')">Change to Blue</button>

    <button onclick="changeBackground('gray')">Change to Gray</button>

    <button onclick="changeBackground('white')">Reset to White</button>

    <script>

        function changeBackground(color) {

    document.body.style.backgroundColor = color;

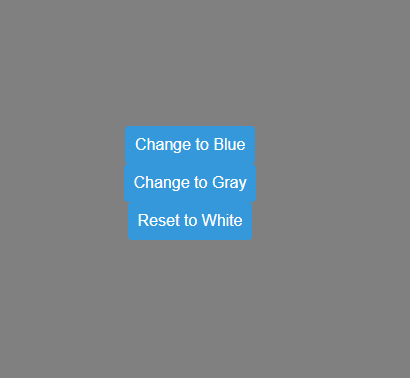
}

    </script>

</body>

</html>

**A blue background with white text

Description automatically generated A blue sign with white text

Description automatically generated**

**5.      Related to event handlers, answer 2 out of the following 3 questions. (1 point)**

**a. Why would you sometimes like to control event propagation? (0.5 points)**

Reason: Event propagation refers to the order in which events are handled as they move through the DOM hierarchy. In certain cases, we might want to control the propagation to ensure that an event doesn't trigger unnecessary or conflicting actions on parent or child elements. This is particularly useful when dealing with nested elements and handling events at different levels of the DOM.

**b. Why would you sometimes like to prevent a default action? (0.5 points)**

Reason: When an event occurs on an element, the browser often performs a default action associated with that event (e.g., clicking a link Navigates to a new page). There are situations where you might want to override this default behavior. For example, in a single-page application, you might handle form submissions via JavaScript without navigating to a new page. By preventing the default action, you can implement custom behavior without the browser's default actions interfering.

**c. What is debouncing an event? (0.5 points)**

Explanation: Debouncing an event is a technique used to ensure that time-consuming tasks (like expensive calculations or API requests) triggered by an event don't fire so often, especially during rapid and repeated events like scrolling or resizing. Instead of executing the associated function for every event, debouncing introduces a delay. If the events are closely spaced in time, only the last event within the specified delay period triggers the function. This helps optimize performance and avoids unnecessary or redundant processing for events that occur in quick succession.

**User6.      Make exercise “Balloon” from book (Exercises, chapter 15).  (2 pts)**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Balloon</title>

    <style>

        body {

            text-align: center;

            font-size: 20px;

            margin: 100px;

        }

    </style>

</head>

<body>

    <div id="balloon">🎈</div>

    <script src="script.js"></script>

</body>

</html>

**script.js**

const balloon = document.getElementById("balloon");

let fontSize = 20; // initial font size in pixels

function inflateBalloon() {

    fontSize \*= 1.1; // inflate by 10%

    balloon.style.fontSize = fontSize + "px";

    // check if balloon size exceeds a certain limit

    if (fontSize > 100) {

        balloon.innerHTML = "💥"; // explode

        document.removeEventListener("keydown", handleKey); // remove event listener

    }

}

function deflateBalloon() {

    fontSize \*= 0.9; // deflate by 90%

    balloon.style.fontSize = fontSize + "px";

}

function handleKey(event) {

    if (event.key === "ArrowUp") {

        inflateBalloon();

    } else if (event.key === "ArrowDown") {

        deflateBalloon();

    }

}

// Add event listener for key presses

document.addEventListener("keydown", handleKey);

A red balloon with a blue string

Description automatically generated A red balloon with a blue string

Description automatically generated

**A red and yellow star

Description automatically generated**