CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY
An Autonomous Institute | Affiliated to Osmania University
Kokapet Village, Gandipet Mandal, Hyderabad, Telangana-500075, www.cbit.ac.in
COMMITTED TO RESEARCH, INNOVATION AND EDUCATION 46 years

1.HTML code to demonstrate Methods of window object in Javascript.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Window Object Methods Demo</title>
</head>
<body>
    <h2>Window Object Methods Demonstration</h2>
    <button onclick="showAlert()">Alert</button>
    <button onclick="showConfirm()">Confirm</button>
    <button onclick="showPrompt()">Prompt</button>
    <button onclick="openNewWindow()">Open Window</button>
    <button onclick="closeNewWindow()">Close Window</button>
    <button onclick="startTimeout()">Set Timeout</button>

    <script>
        let newWindow;

        function showAlert() {
            alert("This is an alert box!");
        }

        function showConfirm() {
            let result = confirm("Do you want to proceed?");
            alert("User selected: " + (result ? "OK" : "Cancel"));
        }

        function showPrompt() {
            let userInput = prompt("Please enter your name:", "Guest");
            alert("Hello, " + (userInput ? userInput : "Guest") + "!");
        }

        function openNewWindow() {
            newWindow = window.open("https://www.example.com", "_blank",
"width=600,height=400");
        }
```

```
        function closeNewWindow() {
            if (newWindow) {
                newWindow.close();
            } else {
                alert("No window to close.");
            }
        }

        function startTimeout() {
            setTimeout(function() {
                alert("This alert appeared after a delay!");
            }, 3000);
        }
    </script>
</body>
</html>
```

2.HTML code to demonstrate Mouse CLICK EVENTS in JavaScript

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Event Handling Demo</title>
</head>
<body>
    <h2>Event Handling Demonstration</h2>

    <button onclick="showAlert()">Click Me</button>
    <button onmouseover="changeText(this)">Hover Over Me</button>

    <input type="text" id="focusInput" onfocus="focusMessage()"
placeholder="Click here to focus">
    <input type="text" id="keyInput" onkeydown="keyPressed(event)"
placeholder="Type something">

    <p id="message">Interact with the elements to see the effects.</p>

    <script>
        function showAlert() {
            alert("Button Clicked!");
        }

        function changeText(element) {
```

```
            element.textContent = "Hovered!";
        }

        function focusMessage() {
            document.getElementById("message").textContent = "Input field
focused!";
        }

        function keyPressed(event) {
            document.getElementById("message").textContent = "Key Pressed: " +
event.key;
        }
    </script>
</body>
</html>
```

3.HTML code to demonstrate `OnSubmit()` in Javascript.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>OnSubmit Event Demo</title>
    <script>
        function validateForm(event) {
            event.preventDefault(); // Prevents the form from submitting
            let name = document.getElementById("name").value;
            if (name === "") {
                alert("Name cannot be empty!");
            } else {
                alert("Form submitted successfully with name: " + name);
            }
        }
    </script>
</head>
<body>
    <h2>OnSubmit Event Demonstration</h2>
    <form onsubmit="validateForm(event)">
        <label for="name">Enter your name:</label>
        <input type="text" id="name" name="name" required>
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```

4.HTML code to demonstrate `onLoad() and unload()` in Javascript.

```
<!DOCTYPE html>
<html lang="en" onload="pageLoaded()" onunload="pageUnloaded()">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>OnLoad and Unload Events Demo</title>
    <script>
        function pageLoaded() {
            alert("Page has loaded successfully!");
        }

        function pageUnloaded() {
            alert("Page is unloading. Goodbye!");
        }
    </script>
</head>
<body>
    <h2>OnLoad and Unload Events Demonstration</h2>
    <p>Open this page to see the onLoad event in action. Close or reload to
trigger the unload event.</p>
</body>
</html>
```

5.HTML code to demonstrate `Date and Time in JavaScript.`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Date and Time Demo</title>
    <script>
        function showDateTime() {
            let now = new Date();
            let date = now.toDateString();
            let time = now.toLocaleTimeString();
            document.getElementById("dateTime").textContent = "Date: " + date
+ " | Time: " + time;
        }

        function updateTime() {
            let now = new Date();
```

```
            document.getElementById("clock").textContent =
now.toLocaleTimeString();
        }

        setInterval(updateTime, 1000);
    </script>
</head>
<body onload="showDateTime()">
    <h2>Date and Time Demonstration</h2>
    <p id="dateTime"></p>
    <h3>Live Clock:</h3>
    <p id="clock"></p>
    <button onclick="showDateTime()">Refresh Date & Time</button>
</body>
</html>
```

Record programs: Navigate to a particular element using DOM and modify it.

CaseStudy 1:**Updating a Product Price on an E-Commerce Website**

Scenario:

You are developing an e-commerce website where product details are displayed dynamically. One of the products listed has a price that needs to be updated due to a discount. Your task is to navigate to the price element using the DOM and modify its value when the "Apply Discount" button is clicked.

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DOM Navigation - Modify Element</title>
    <style>
        .product {
            border: 2px solid black;
            padding: 20px;
            width: 300px;
            text-align: center;
            margin: 20px auto;
        }
```

```
        #price {
            font-size: 20px;
            font-weight: bold;
            color: red;
        }
    </style>
</head>
<body>

    <div class="product">
        <h2>Wireless Headphones</h2>
        <p id="price">$100</p>
        <button onclick="applyDiscount()">Apply Discount</button>
    </div>

    <script>
        function applyDiscount() {
            // Navigate to the price element
            let priceElement = document.getElementById("price");

            // Modify the price content
            priceElement.innerHTML = "$80"; // New discounted price

            // Change style to indicate discount applied
            priceElement.style.color = "green";
            priceElement.style.fontSize = "24px";
        }
    </script>

</body>
</html>
```

Case Study 2 : Real-Time Stock Price Update in a Finance Website

Scenario:

You are developing a **finance website** that displays real-time **stock prices** for multiple companies. The stock prices change dynamically, and when the "Update Prices" button is clicked, the values for all stocks are updated using **`document.getElementsByClassName()`**

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Live Stock Price Update</title>
    <style>
        body {
            text-align: center;
            font-family: Arial, sans-serif;
        }
        .stock-container {
            width: 60%;
            margin: 20px auto;
            padding: 20px;
            border: 2px solid black;
            background-color: #f4f4f4;
        }
        .stock {
            font-size: 24px;
            margin: 10px 0;
        }
        .price {
            color: red;
            font-weight: bold;
        }
        button {
            margin-top: 20px;
            padding: 10px 20px;
            font-size: 18px;
            cursor: pointer;
        }
    </style>
</head>
<body>

    <h1>Live Stock Price Update</h1>

    <div class="stock-container">
        <p class="stock">Apple (AAPL): <span class="price">$150</span></p>
        <p class="stock">Google (GOOGL): <span class="price">$2800</span></p>
        <p class="stock">Amazon (AMZN): <span class="price">$3300</span></p>
        <p class="stock">Tesla (TSLA): <span class="price">$900</span></p>
    </div>

    <button onclick="updatePrices()">Update Prices</button>

    <script>
        function updatePrices() {
            // Select all elements with class "price"
            let priceElements = document.getElementsByClassName("price");
```

```
            // Loop through each price element and update its content
            for (let i = 0; i < priceElements.length; i++) {
                let randomPrice = (Math.random() * (5000 - 500) +
500).toFixed(2);
                priceElements[i].innerHTML = "$" + randomPrice;

                // Change style to indicate a change
                priceElements[i].style.color = "green";
                setTimeout(() => {
                    priceElements[i].style.color = "red";
                }, 1000);
            }
        }
    </script>

</body>
</html>
```

Challenge 1: Auto-Update Prices Every 5 Seconds

- **Task:** Modify the code to **automatically update** stock prices **every 5 seconds**, simulating a real stock ticker.
- **Hint:** Use setInterval() to call updateStockPrices() every 5 seconds.

Challenge 2: Add a "Stop Auto-Update" Button

- **Task:** Allow users to **stop the auto-update** when they click a button.
- **Hint:** Use clearInterval() to stop setInterval().

Case Study 3 : Live Score Update in a Sports Website

Scenario:

You are developing a **sports website** that displays the **live score** of a football match. Every time a team scores a goal, the score updates dynamically using the **DOM**.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Live Football Score</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
        }
        .scoreboard {
            width: 50%;
            margin: auto;
            padding: 20px;
            border: 2px solid black;
            background-color: #f4f4f4;
        }
        .team {
            font-size: 24px;
            font-weight: bold;
            margin: 10px;
        }
        .score {
            font-size: 40px;
            color: blue;
        }
        .highlight {
            color: red;
            font-weight: bold;
        }
        button {
            margin: 10px;
            padding: 10px 20px;
            font-size: 18px;
            cursor: pointer;
        }
    </style>
</head>
<body>

    <h1>Live Football Score</h1>

    <div class="scoreboard">
        <p class="team">Team A: <span id="scoreA" class="score">0</span></p>
        <p class="team">Team B: <span id="scoreB" class="score">0</span></p>
    </div>

    <button onclick="goalScored('A')">Goal Scored by Team A</button>
    <button onclick="goalScored('B')">Goal Scored by Team B</button>

    <script>
```

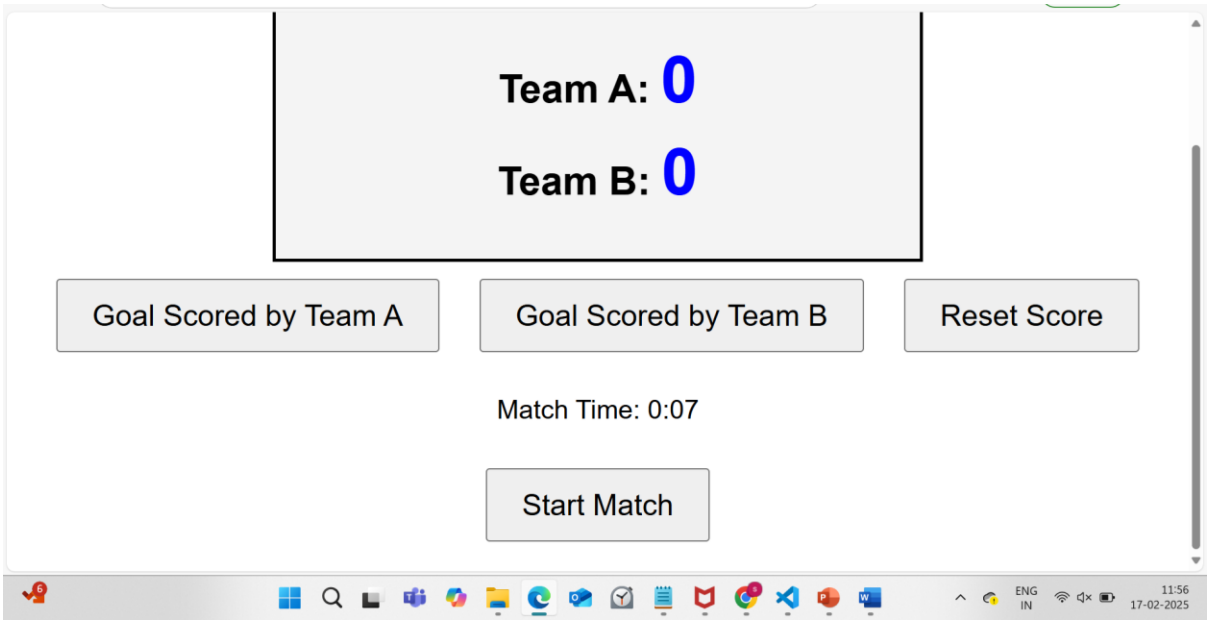
```
        function goalScored(team) {
            // Select the correct score element
            let scoreElement = document.querySelector("#score" + team);
            let allScores = document.querySelectorAll(".score");

            // Increase the score by 1
            let currentScore = parseInt(scoreElement.innerHTML);
            scoreElement.innerHTML = currentScore + 1;

            // Remove highlight from all scores
            allScores.forEach(score => score.classList.remove("highlight"));

            // Highlight the updated score
            scoreElement.classList.add("highlight");

            // Remove highlight after 1 second
            setTimeout(() => {
                scoreElement.classList.remove("highlight");
            }, 1000);
        }
    </script>

</body>
</html>
```

Challenge 1: Add a Reset Button

- **Task:** Add a button to reset both scores to **zero**.
- **Hint:** Use document.querySelectorAll() to select both scores and reset them.

Challenge 2: Add a Match Timer

- **Task:** Display a timer that **counts up** from 0:00 when the match starts.
- **Hint:** Use setInterval() to update the time every second.



Challenge 3: Add a Foul Counter

- **Task:** Keep track of fouls for each team and update dynamically.
- **Hint:** Similar to score tracking but with a separate counter.