# ENSF 381
# Full Stack Web Development

## Lecture 15: Events and DOM

**Slides: Ahmad Abdellatif, PhD**

**Instructor: Novarun Deb, PhD**

UNIVERSITY OF CALGARY

# Outline

- Introduction to events.
- Common HTML events.
- Document Object Model (DOM).

# What are the events in JavaScript?

- Actions or occurrences that happen in the browser, often as a result of user interactions or other activities.

- Allows developers to respond to events by attaching event handlers, which are functions that get executed when a specific event occurs.

- Here are some common types of events in JavaScript:
  - Mouse Events
  - Keyboard Events
  - Form Events
  - Window Events

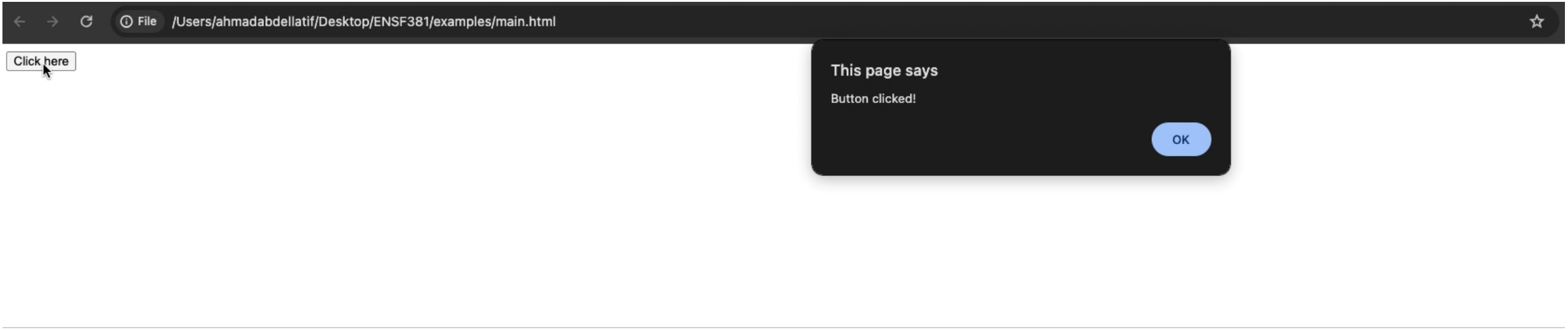
# Common HTML events

- `onclick`: triggered when an element, such as a button or a link, is clicked by the user.

# Events - example

```
<!DOCTYPE html>
<html>
<head>
  <title>onclick Attribute Example</title>
</head>
<body>

<button onclick="showMessage()">Click here</button>

<script>
// JavaScript function to be called when the button is clicked
function showMessage() {
  alert("Button clicked!");
}
</script>

</body>
</html>
```

# Events - example



# Common HTML events

- `onclick`: triggered when an element, such as a button or a link, is clicked by the user.

- `onkeydown`: fired when a key on the keyboard is pressed down. It is often used to capture and respond to user keyboard input.

# Events - example

```
<!DOCTYPE html>
<html>
<head>
  <title>onkeydown Event Example</title>
</head>
<body>

<label>Press a key in the input field:</label> <br>
<input type="text" id="myInput" onkeydown="keyPressed(event)">

<script>
// JavaScript function to handle the keydown event
function keyPressed(event) {
  // Display the key code and the pressed key
  alert("Key Code: " + event.keyCode + "\nPressed Key: " + event.key);
}
</script>

</body>
</html>
```

# Events - example

# Common HTML events

- `onclick`: triggered when an element, such as a button or a link, is clicked by the user.
- `onkeydown`: fired when a key on the keyboard is pressed down. It is often used to capture and respond to user keyboard input.
- `onchange`: occurs when the value of an input element, such as a text field or dropdown, is changed by the user.
- `onmouseover`: occurs when the mouse pointer is moved over an element, such as an image or a hyperlink.
- `onmouseout`: triggered when the mouse pointer moves out of an element after previously being over it.
- `onload`: triggered when a webpage or document has finished loading in the browser. It provides an opportunity to perform actions or execute scripts once the page is fully rendered.

# Document Object Model (DOM)

- DOM is a programming interface that represents the structure of a document as a tree of objects.

- It provides a way for programs to **manipulate** the structure, style, and content of these documents.

- Using DOM, we can:
  - Modify the HTML elements and attributes throughout the page.
  - Change all the CSS styles in the page.
  - Introduce additional HTML elements and attributes.

# Locating HTML elements within the document

- `document.getElementById(id)`: returns a reference to the element with the specified id attribute within the document (used to access and manipulate a specific HTML element).

- `document.getElementsByTagName(name)`: returns a collection of HTML elements with the given tag name (used to select multiple elements with the same tag name).

- `document.getElementsByClassName(name)`: returns a collection of HTML elements with the specified class name.

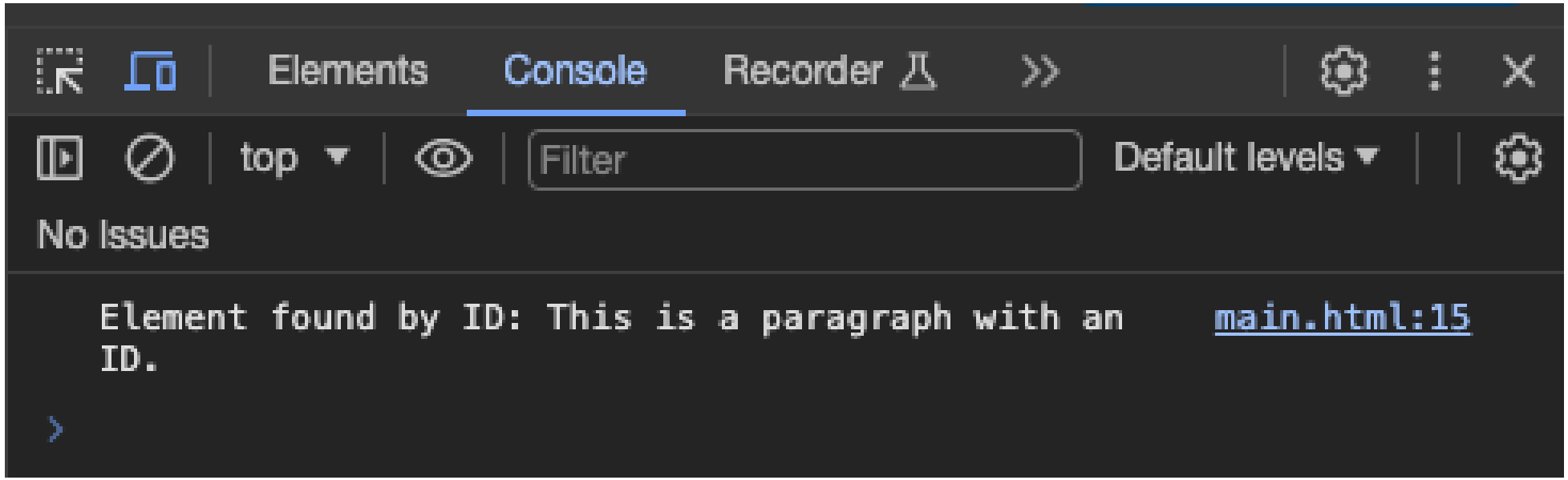
# Accessing the content of HTML elements

- `element.innerHTML`: sets or gets the HTML content.
- `element.textContent`: sets or gets the text content inside the HTML element.
- `element.attribute`: assigns a new value to the specified attribute of an HTML element.
- `element.style.property`: allows you to change the style of an HTML element by assigning a new value to a specific CSS property.

# Example on getting the element by ID and log its text content

```
<!DOCTYPE html>
<html>
<head>
  <title>Get Element by ID Example</title>
</head>
<body>

<!-- Get Element by ID Example -->
<p id="demoParagraph">This is a paragraph with an ID.</p>

<script>
var paragraphById = document.getElementById("demoParagraph");
if (paragraphById) {
  console.log("Element found by ID: " + paragraphById.textContent);
}
</script>
</body>
</html>
```

# Example on getting the element by ID and log its text content



# Example on modifying element content

```
<!DOCTYPE html>
<html>
<head>
  <title>DOM Example</title>
</head>
<body>

<h1 id="myHeading">Hello, World!</h1>
<button onclick="changeText()">Change Text</button>
<script>
function changeText() {
  // Access the h1 element with the id "myHeading"
  var headingElement = document.getElementById("myHeading");

  // Change the text content of the heading
  headingElement.textContent = "New Text!";
}
</script>
</body>
</html>
```

# Example on modifying element content

# innerHTML vs textContent - example

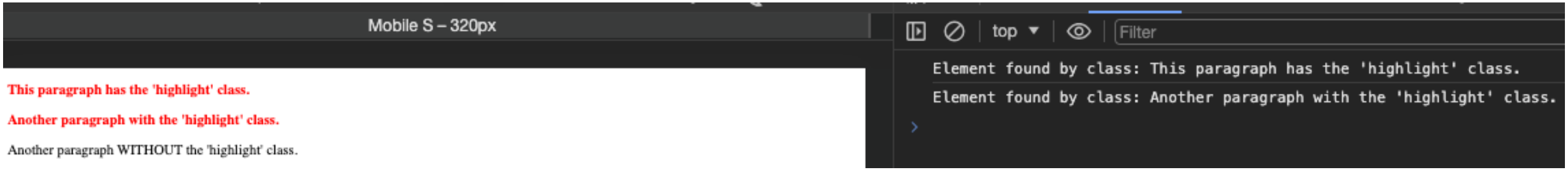
```
<html>
<head> <title>elements</title> </head>
<body>
  <h1>  ENSF381 Course </h1>

  <div id="test-btn">
    This div element contains <b>bold</b> and <i>italic text</i>.
  </div>

  <button onClick="innerHTMLOutput()"> innerHTML </button>

  <button onClick="textContentOutput()"> textContent </button>

  <script>
    function textContentOutput() {
      var div_elem = document.getElementById('test-btn');
      console.log(div_elem.textContent);
    }

    function innerHTMLOutput() {
      var div_elem = document.getElementById('test-btn');
      console.log(div_elem.innerHTML);
    }
  </script>
</body> </html>
```

# innerHTML vs textContent - example

# Example on getting elements by class and log their text content

```
<!DOCTYPE html>
<html>
<head>
  <title>Get Elements by Class Example</title>
  <style>
    .highlight {
      color: red;
      font-weight: bold;
    }
  </style>
</head>
<body>
<p class="highlight">This paragraph has the 'highlight' class.</p>
<p class="highlight">Another paragraph with the 'highlight' class.</p>
<p> Another paragraph WITHOUT the 'highlight' class.</p>

<script>
var elementsByClass = document.getElementsByClassName("highlight");
for (var i = 0; i < elementsByClass.length; i++) {
  console.log("Element found by class: " + elementsByClass[i].textContent);
}
</script>
</body></html>
```

# Example on getting elements by class and log their text content



# Example on change the image on mouse over

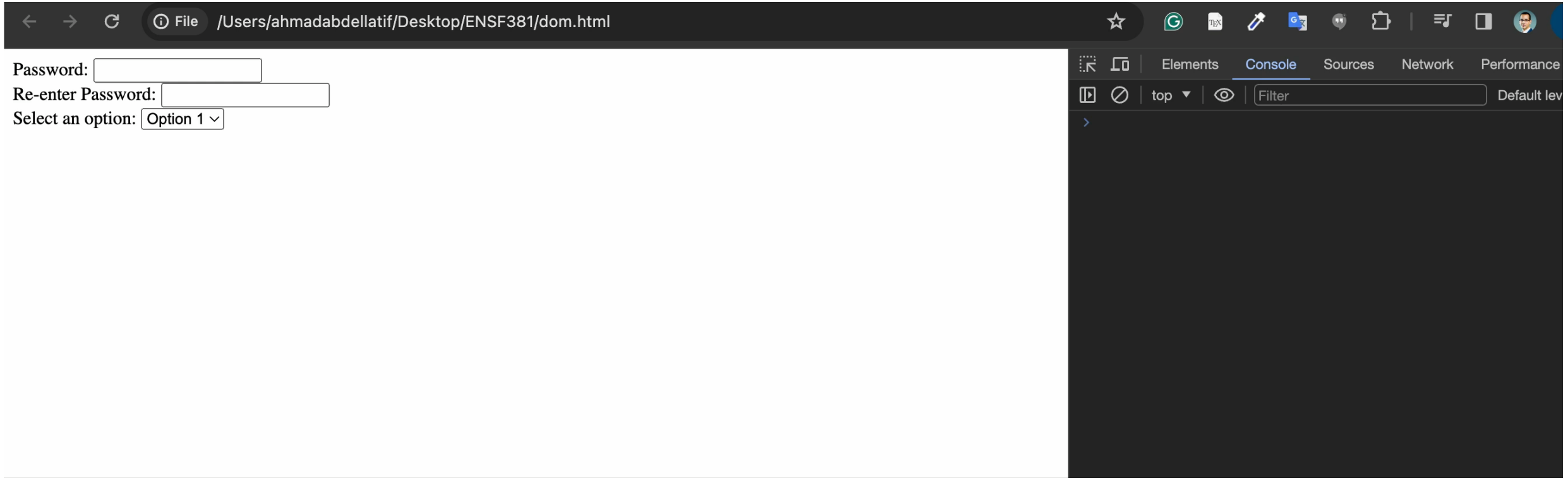
```
<!DOCTYPE html>
<html><head>
  <title>Change Image on Mouse Over Example</title>
  <style>
    /* Add some styling to the image for better visibility */
    #myImage {
      width: 300px;
      height: 300px;
    }
  </style>
</head> <body>
<!-- Image to be changed on mouse over -->
<img id="myImage" src="https://ucalgary.ca/themes/ucalgary/ucws_theme/images/UCalgary.svg" alt="Original Image"
onmouseover="changeImage()" onmouseout="restoreImage()">

<script>
// JavaScript functions to change and restore the image
function changeImage() {
  document.getElementById("myImage").src = "https://wallpapers.com/images/featured-full/flower-pictures-
unpxbv1q9kxyqr1d.jpg";
}

function restoreImage() {
  document.getElementById("myImage").src = "https://ucalgary.ca/themes/ucalgary/ucws_theme/images/UCalgary.svg"
}
</script>
</body></html>
```

# Example on change the image on mouse over


# Example on validating the user's input

```
<!DOCTYPE html>
<html>
<head> <title>Input Validation</title> </head>
   <body> <script>
        function validatePassword() {
            var password = document.getElementById("password").value;
            var confirmPassword = document.getElementById("confirmPassword").value;

            if (password !== confirmPassword) {
                alert("Passwords do not match! Please re-enter.");
                console.log(`password: ${password}`)
                console.log(`confirmPassword: ${confirmPassword}`)
                document.getElementById("confirmPassword").value = "";
            }
        }

        function handleDropdown() {
            var selectedValue = document.getElementById("dropdown").value;
            console.log("Selected value: " + selectedValue);
        }
    </script>

    <label for="password">Password:</label>
    <input type="password" id="password">
    <br>

    <label for="confirmPassword">Re-enter Password:</label>
    <input type="password" id="confirmPassword" onchange="validatePassword()">
    <br>

    <label for="dropdown">Select an option:</label>
    <select id="dropdown" onchange="handleDropdown()">
        <option value="option1">Option 1</option>
        <option value="option2">Option 2</option>
        <option value="option3">Option 3</option>
    </select>
</body> </html>
```

# Example on validating the user's input



# Adding and deleting HTML elements

- `document.createElement(element)`: creates a new HTML element with the specified tag name.
- `ParentElement.appendChild(element)`: appends a new child element to an existing parent element.
- `ParentElement.removeChild(element)`: removes a specified child element from the document.
- `ParentElement.replaceChild(new, old)`: replaces an existing child element with a new one.

# Example on adding a new HTML element

```
<!DOCTYPE html>
<html>
<head>
  <title>createElement Example</title>
  <style>
    .newDiv {
      background-color: lightblue;
      padding: 10px;
      margin: 10px;
    }
  </style>
</head><body id="body">

<!-- Button to create a new div -->
<button onclick="createNewDiv()">Create New Div</button>
<script>
function createNewDiv() {
  // Create a new div element
  var newDiv = document.createElement("div");

  // Set the class of the new div
  newDiv.className = "newDiv";

  // Set the text content of the new div
  newDiv.textContent = "This is a dynamically created div!";

  // Append the new div to the body of the document
  var body_elem = document.getElementById('body')
  body_elem.appendChild(newDiv);
}
</script></body></html>
```

# Example on adding a new HTML element

# Questions