Q.4.1 Demonstrate DOM manipulation using getElementById(, querySelector(, and innerHTML

| Introduction

DOM manipulation allows JavaScript to access and modify HTML elements dynamically.
Three commonly used methods are:

D: > BCA_A1 > Ayush_Kumar_Dubey > Untitled-1.html > html

```html
<!DOCTYPE html>
<html>
<head>
    <title>DOM Manipulation Demo</title>

    <style>
        body {
            font-family: Arial, sans-serif;
            padding: 20px;
            background: #f1f5f9;

            /* Centering content */
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }

        .box {
            background: white;
            padding: 20px;
            width: 350px;
            border-radius: 8px;
            box-shadow: 0px 0px 8px rgba(0,0,0,0.2);
            text-align: center; /* Center text and buttons */
        }

        button {
            padding: 10px 16px;
```

* getElementById() → Selects an element by its ID

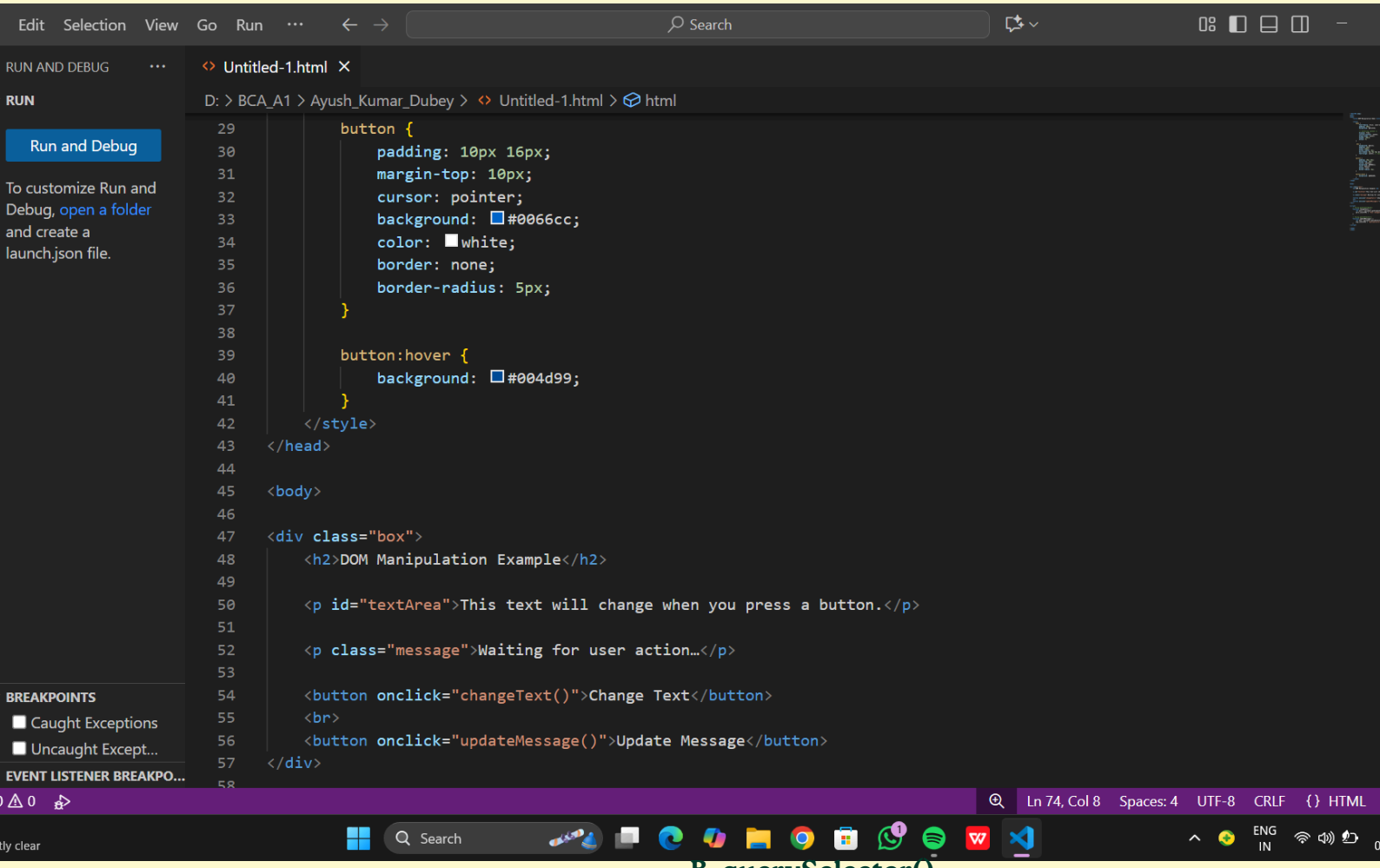
* querySelector() → Selects the first matching CSS selector

* innerHTML → Changes or inserts HTML content inside an element

These tools make web pages interactive and responsive.

Q.4.1 Demonstrate DOM manipulation using getElementById(, querySelector(, and innerHTML

Explanation of the Methods



A. getElementById()

What it does:

Selects an element by its id attribute
Example:

```
let heading = document.getElementById("title");
```

Actions performed:

Changes text color

Replaces the title text using innerHTML

B. querySelector()

What it does:

Selects the first element matching a CSS selector
Example:

```
let message = document.querySelector(".msg");
```

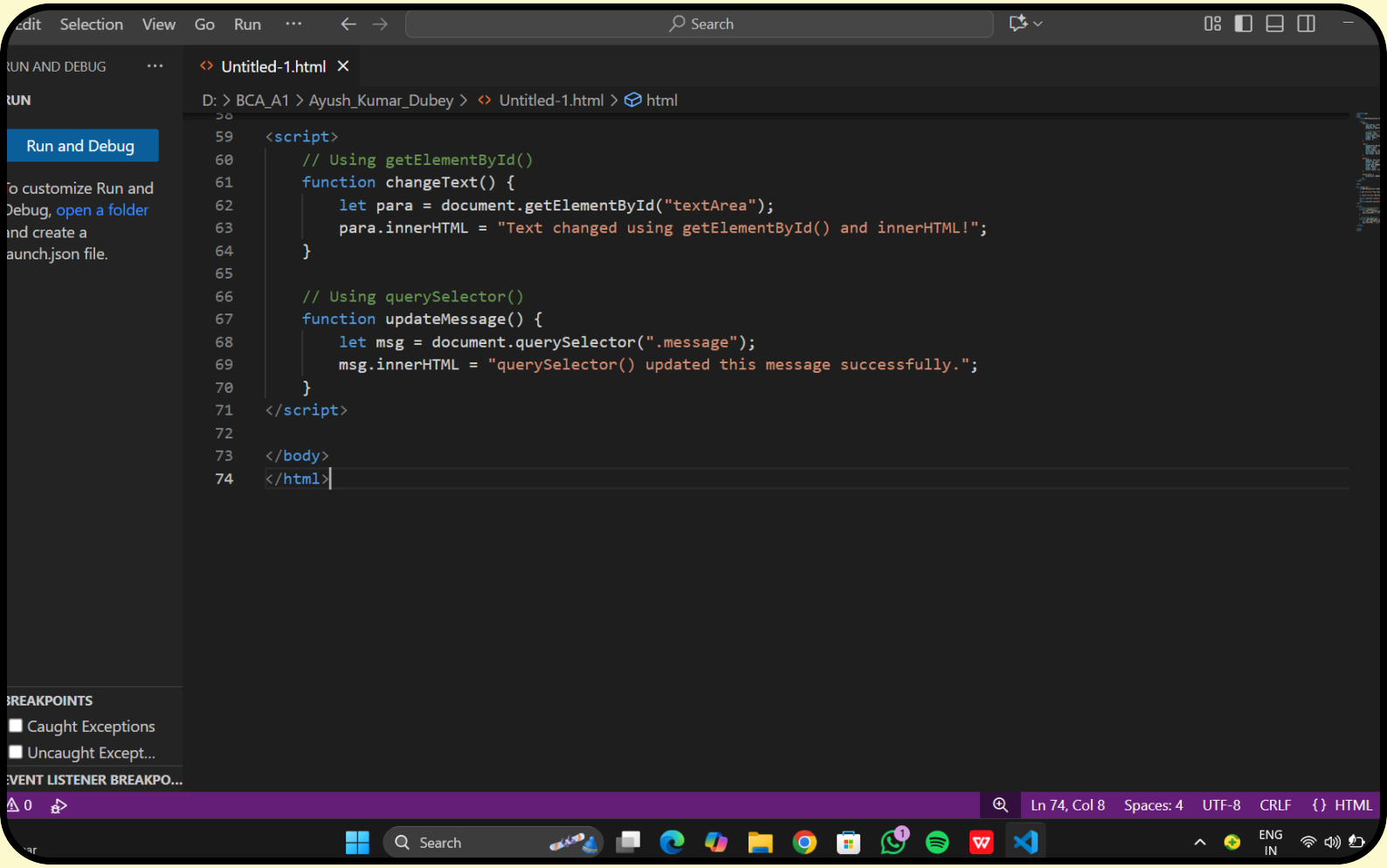
Actions performed:

Makes text bold

Changes its displayed message

Q.4.1 Demonstrate DOM manipulation using getElementById(, querySelector(, and innerHTML

Explanation of the Methods



C. innerHTML

What it does:

Changes or insert

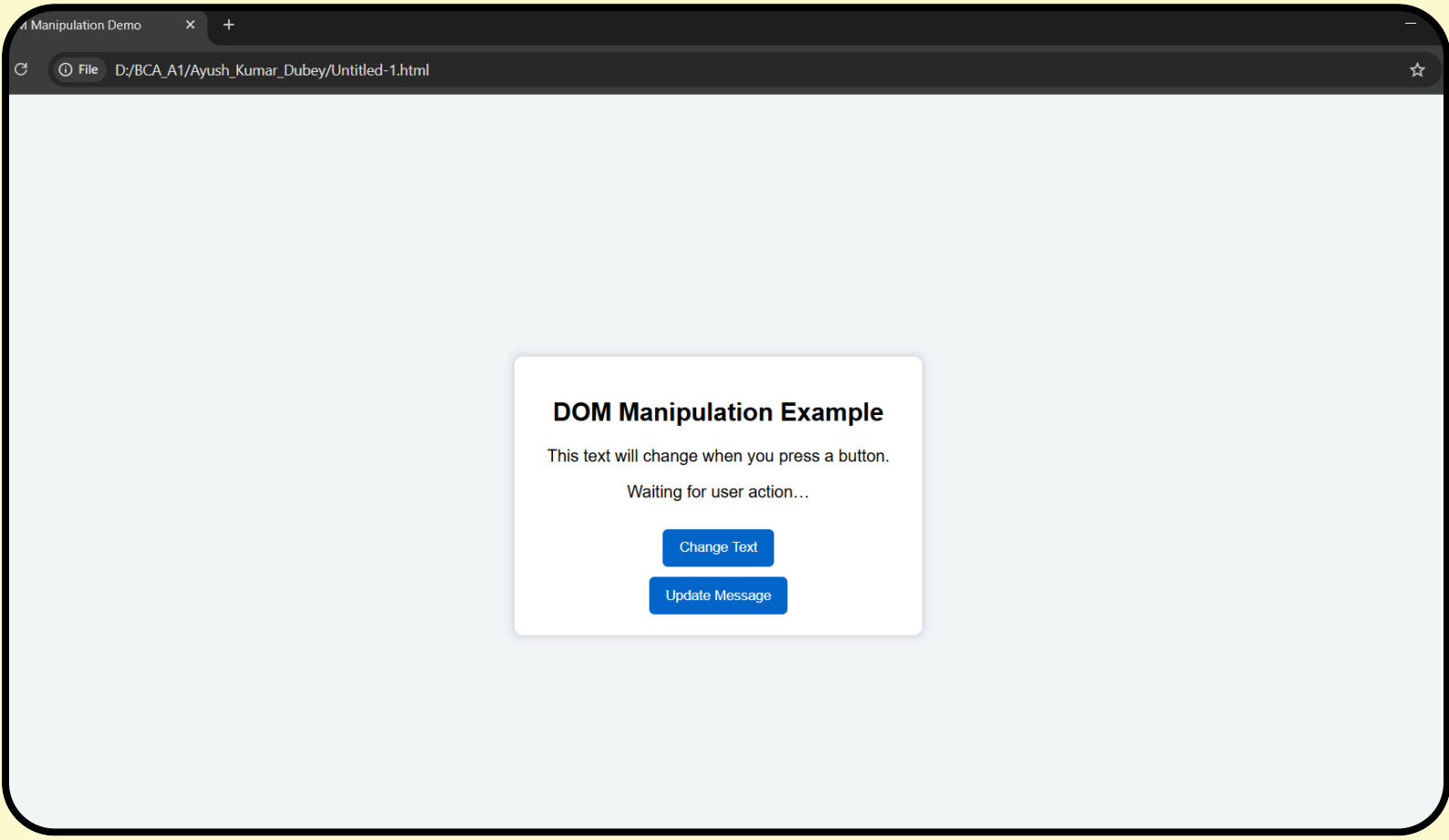
```
document.getElementById("output").innerHTML =
"<p>This is dynamic content added using innerHTML.</p>";
```

Actions performed:

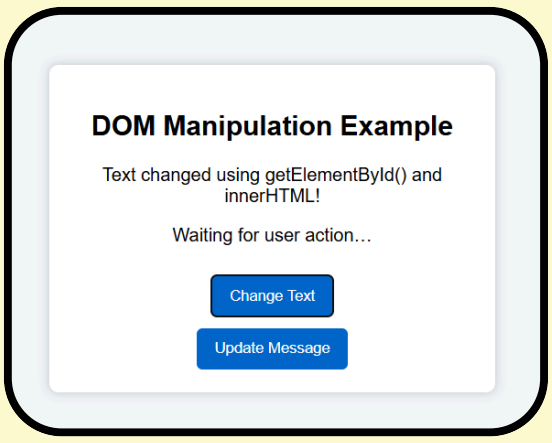
Inserts a new <p> element inside the output container

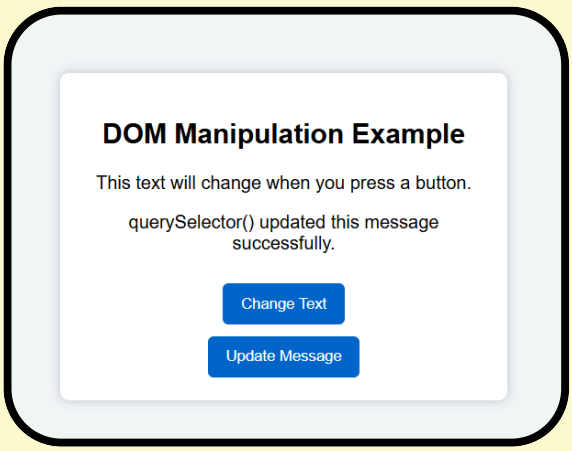
Q.4.1 Demonstrate DOM manipulation using getElementById(, querySelector(, and innerHTML

OUTPUT



EFFECT AFTER USING BUTTON





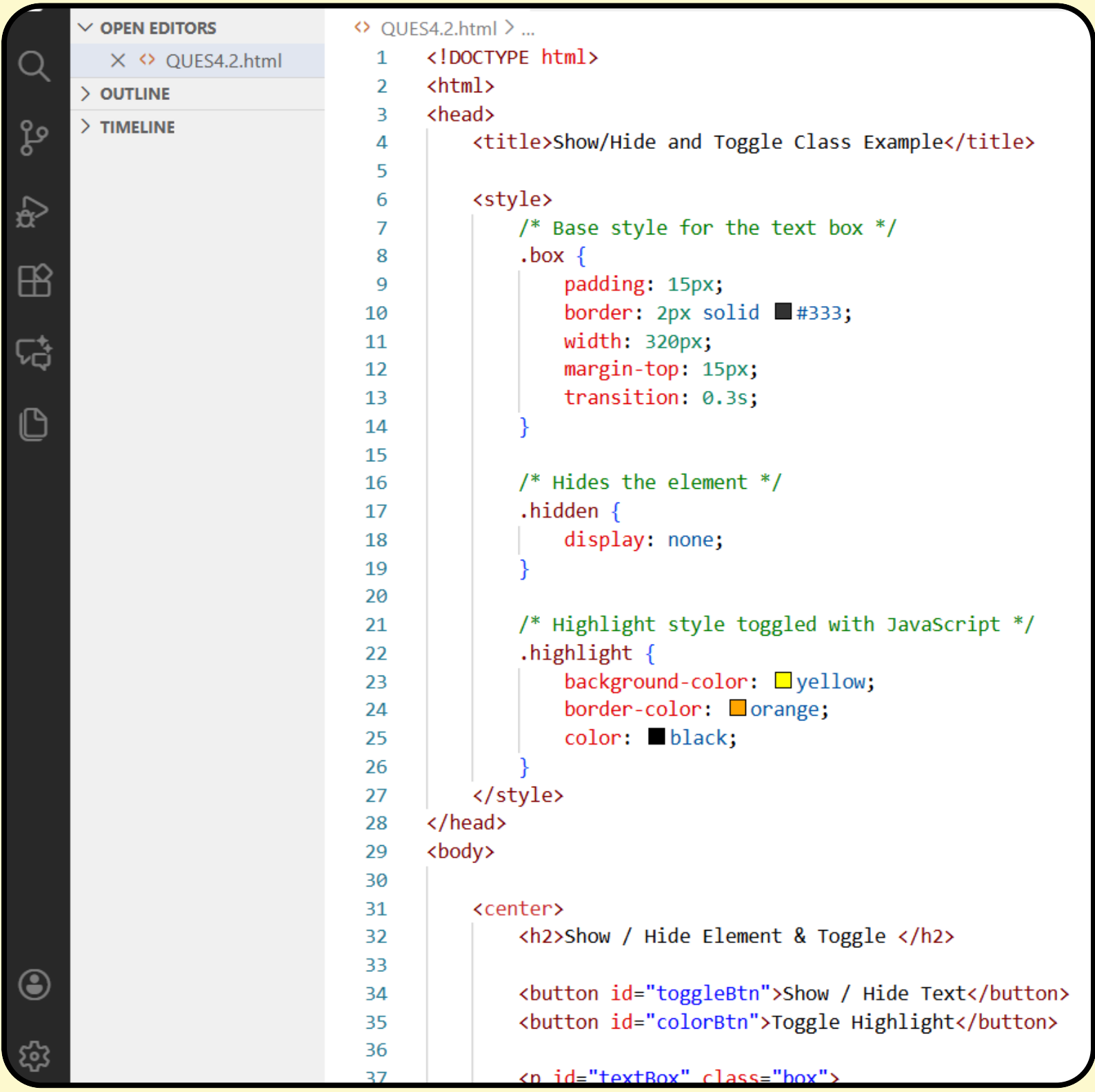
Conclusion

Using getElementById(), querySelector(), and innerHTML, JavaScript can easily select, modify, and inject HTML elements. These methods are essential for building interactive web pages, updating content dynamically, and improving user experience.

Q.4.2 Show or hide HTML elements using JavaScript and toggle CSS classes dynamically

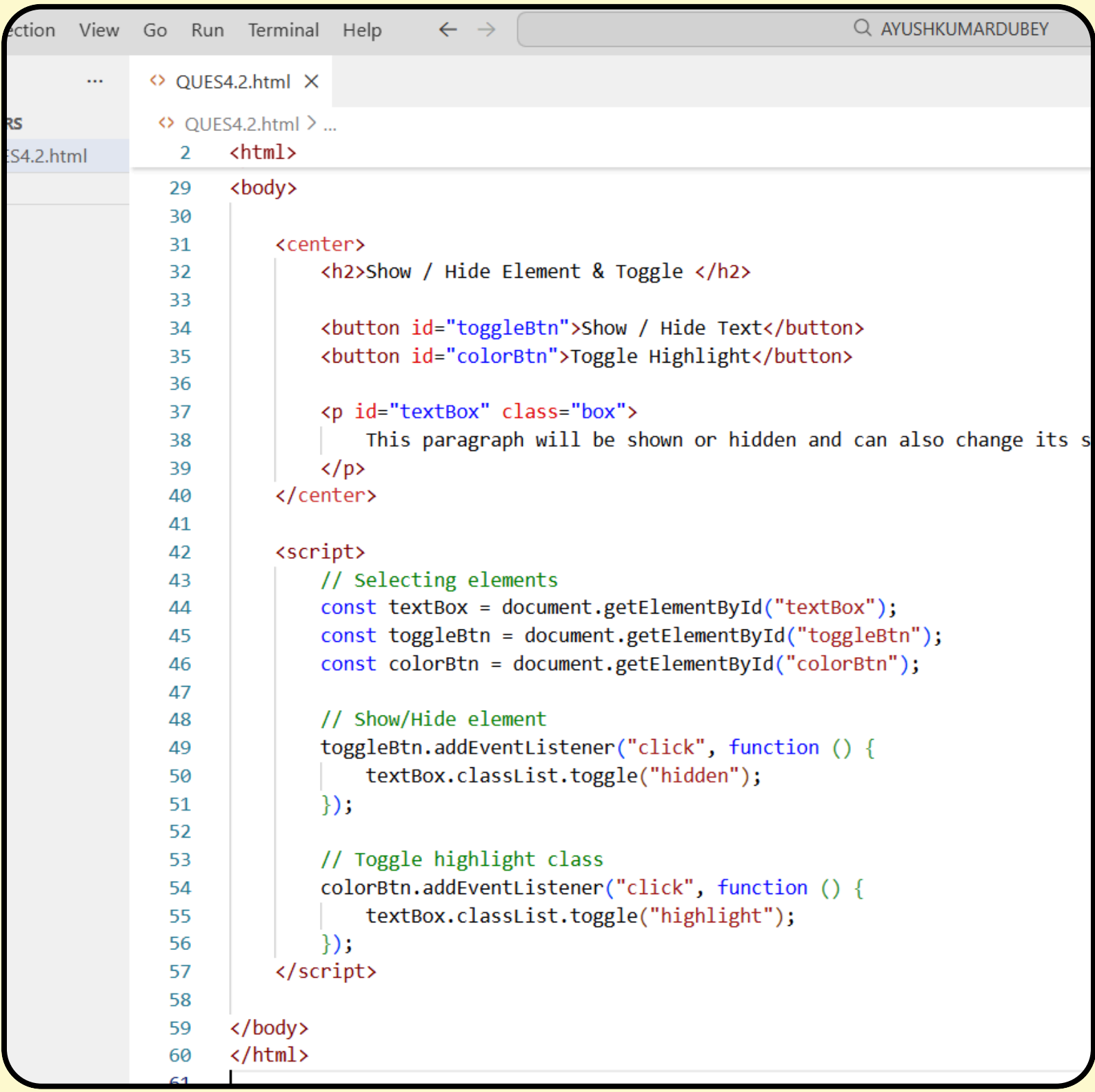
Introduction

JavaScript is commonly used to control the visibility of HTML elements and modify their appearance. Two powerful techniques include:



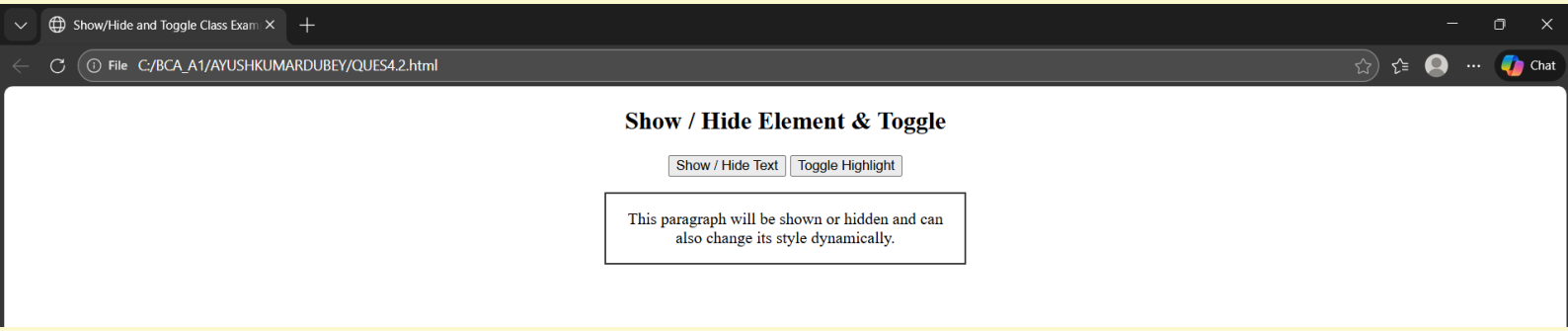
Q.4.2 Show or hide HTML elements using JavaScript and toggle CSS classes dynamically

These features are essential for building interactive UI components such as dropdown menus, pop-ups, alerts, and navigation bars.

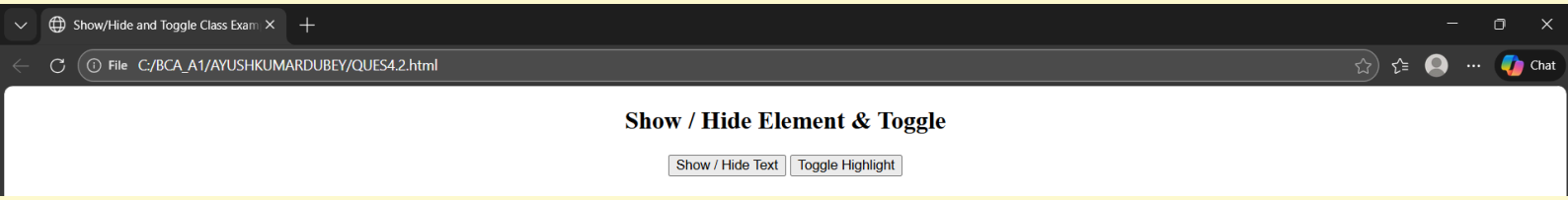


Q.4.2 Show or hide HTML elements using JavaScript and toggle CSS classes dynamically

OUTPUT

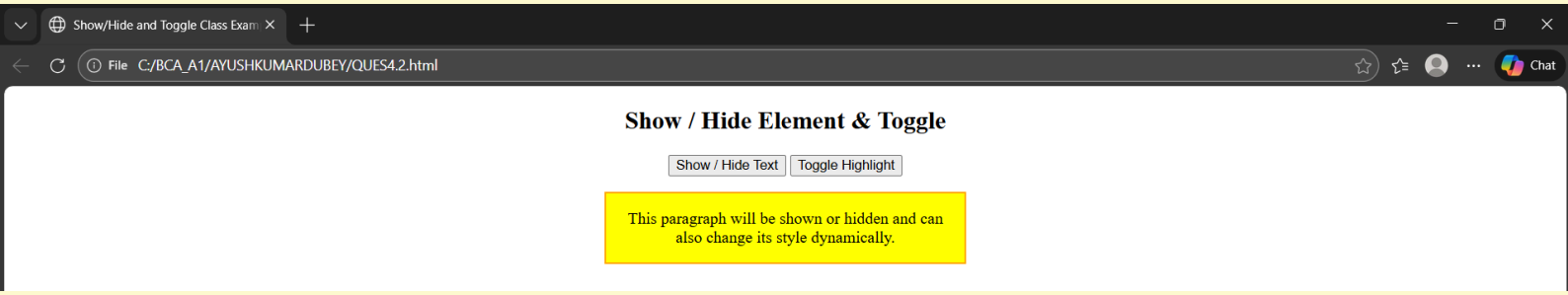


Add A SubheadingNO TEXT BOX HIDDEN WITH SHOW/HIDE TEXT



USING TOGGLE BUTTON

EXAMPLE OF TOGGLE BUTTON



Conclusion

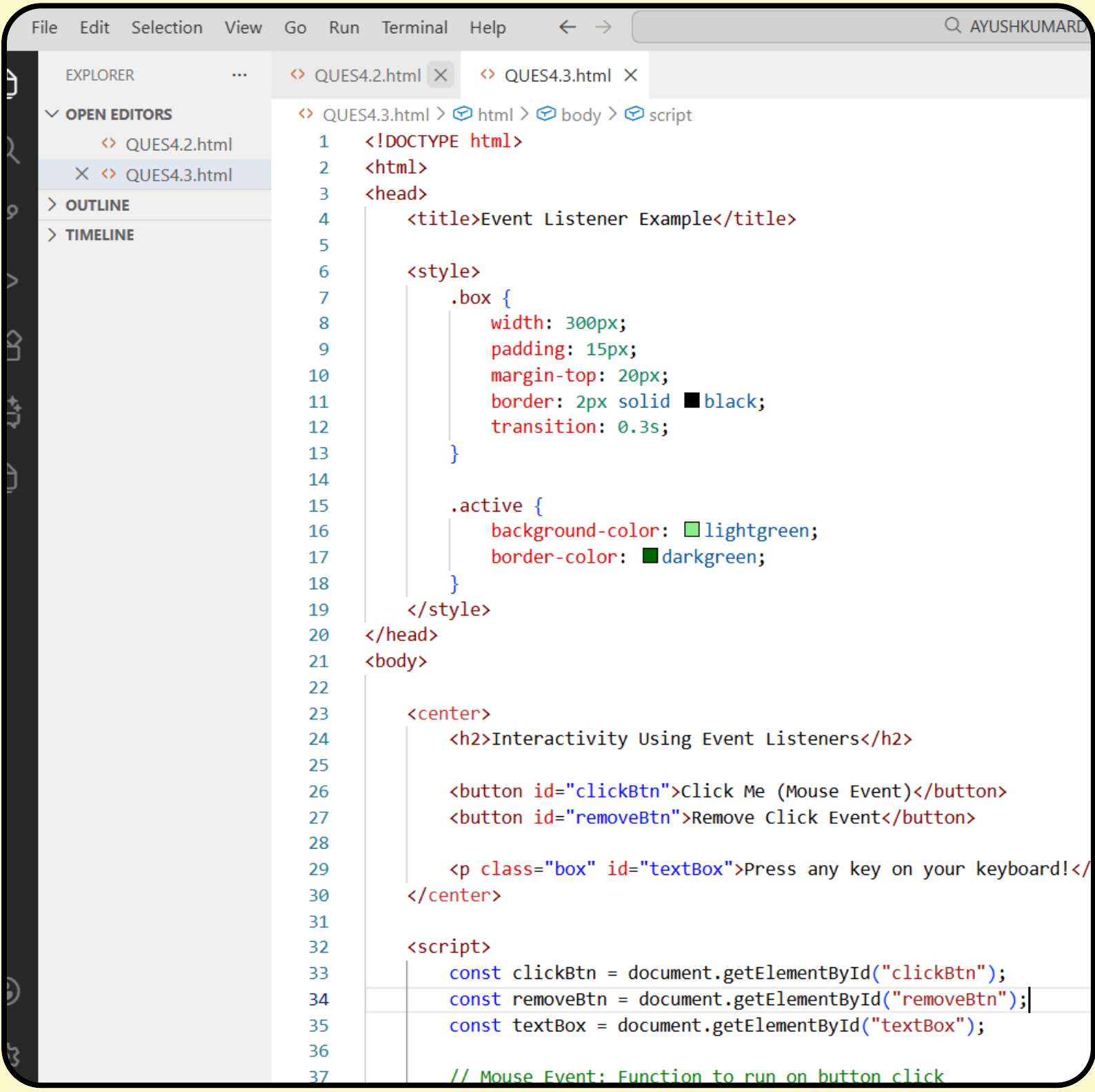
JavaScript provides flexible methods to control visibility and styling of HTML elements. By using;

- style.display
- classList.toggle()
- Custom CSS classes

Q.4.3 Add interactivity using event listeners (addEventListener, removeEventListener) for mouse/keyboard events

INTRODUCTION

Event listeners allow JavaScript to respond to user actions such as clicking, typing, moving the mouse, or pressing keys.

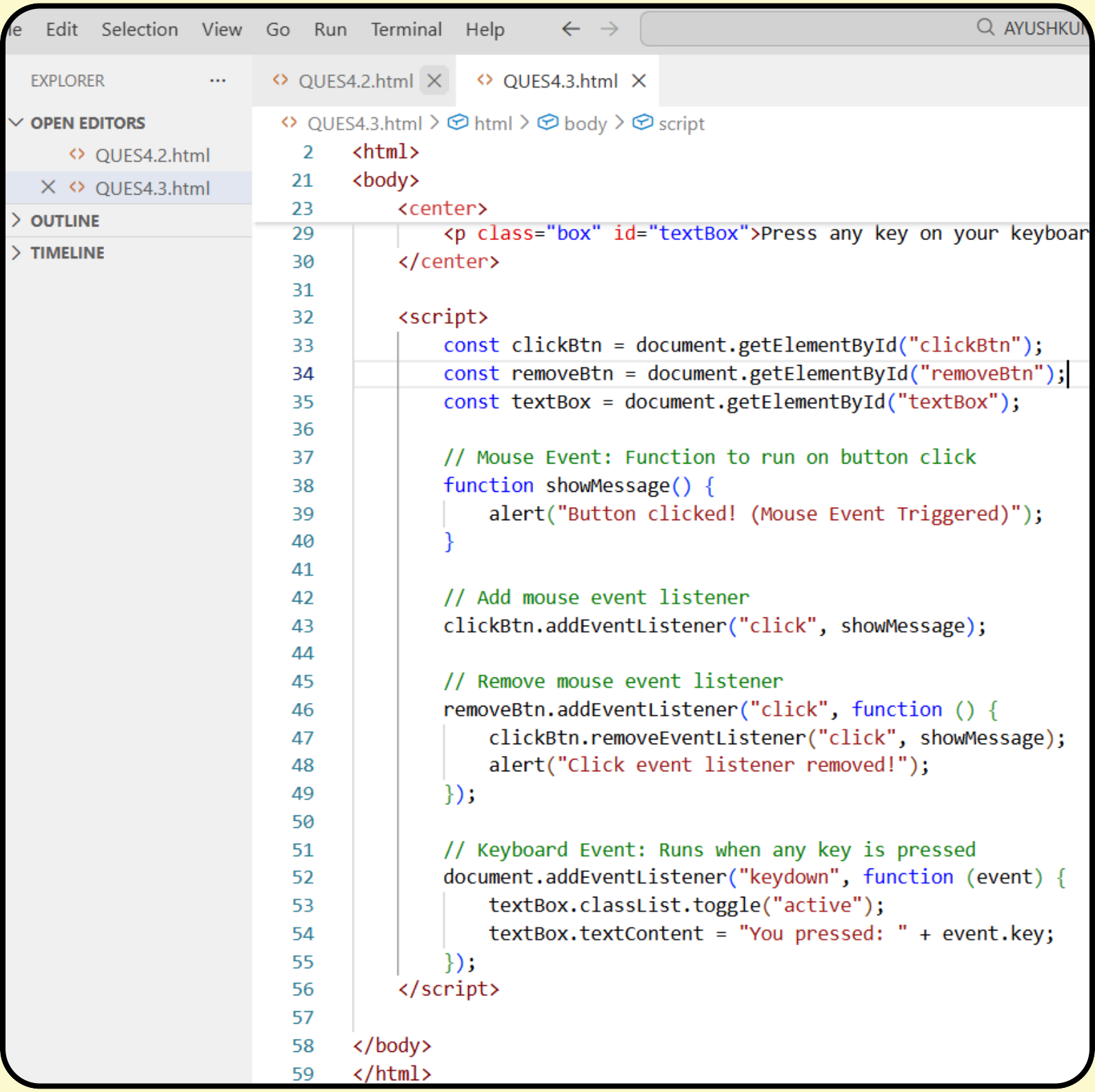


addEventListener()→ **Attach an event**

removeEventListener() → Detach an event

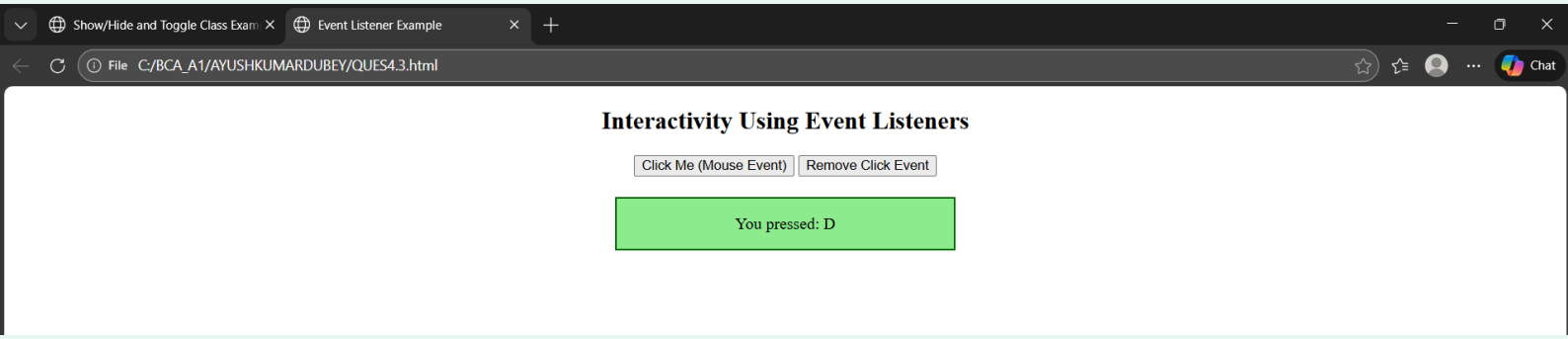
WORKS FOR MOUSE, KEYBOARD, AND OTHER BROWSER EVENTS

Q.4.3 Add interactivity using event listeners (addEventListener, removeEventListener) for mouse/keyboard events

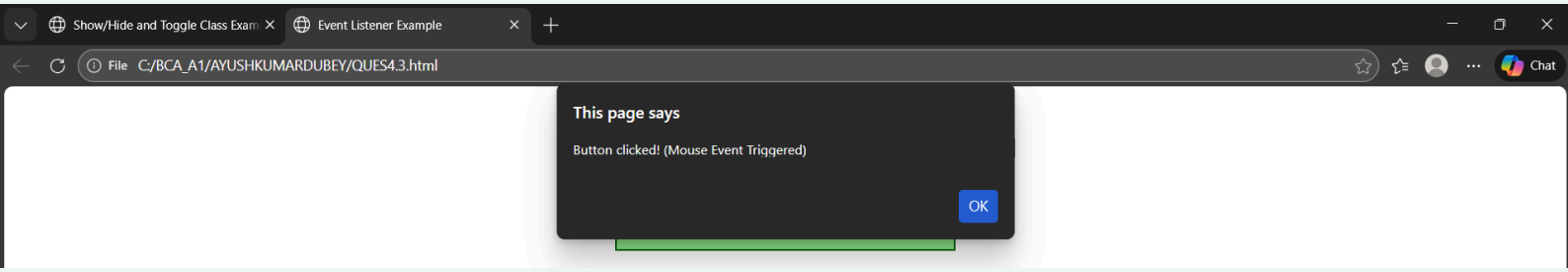


Q.4.3Add interactivity using event listeners (addEventListener, removeEventListener) for mouse/keyboard events

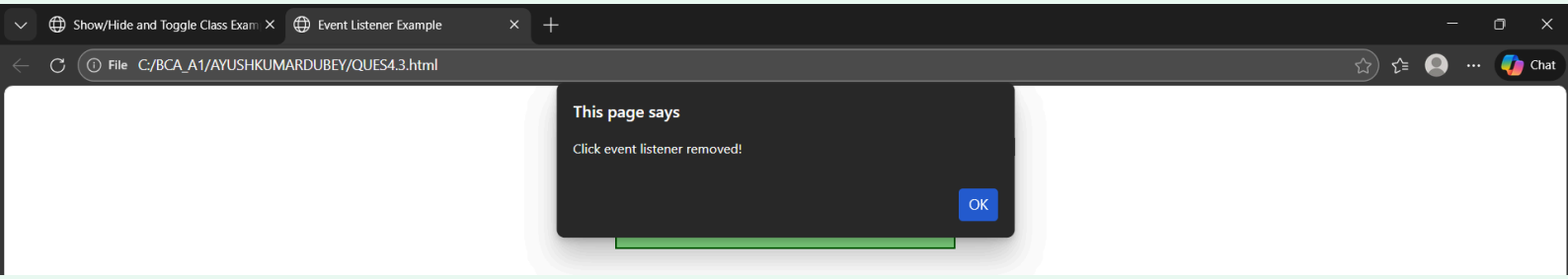
OUTPUT



BUTTON CLICKED



Listener removed



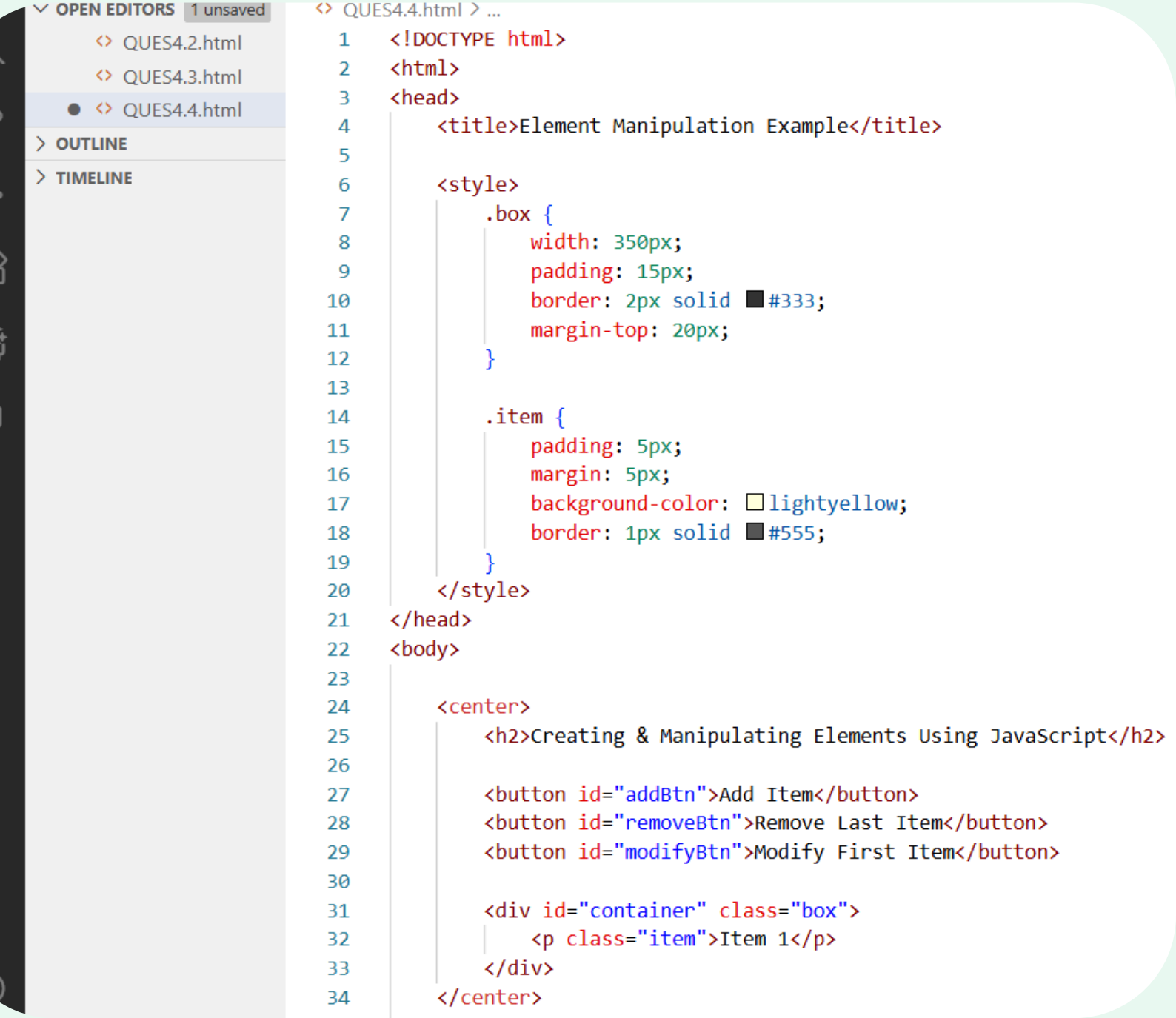
CONCLUSION

- **Event listeners are essential for building interactive front-end experiences.**
- **Using addEventListener() and removeEventListener(), developers can:**

Q.4.4 Create and manipulate elements using JavaScript: append, remove, or modify child nodes

Introduction

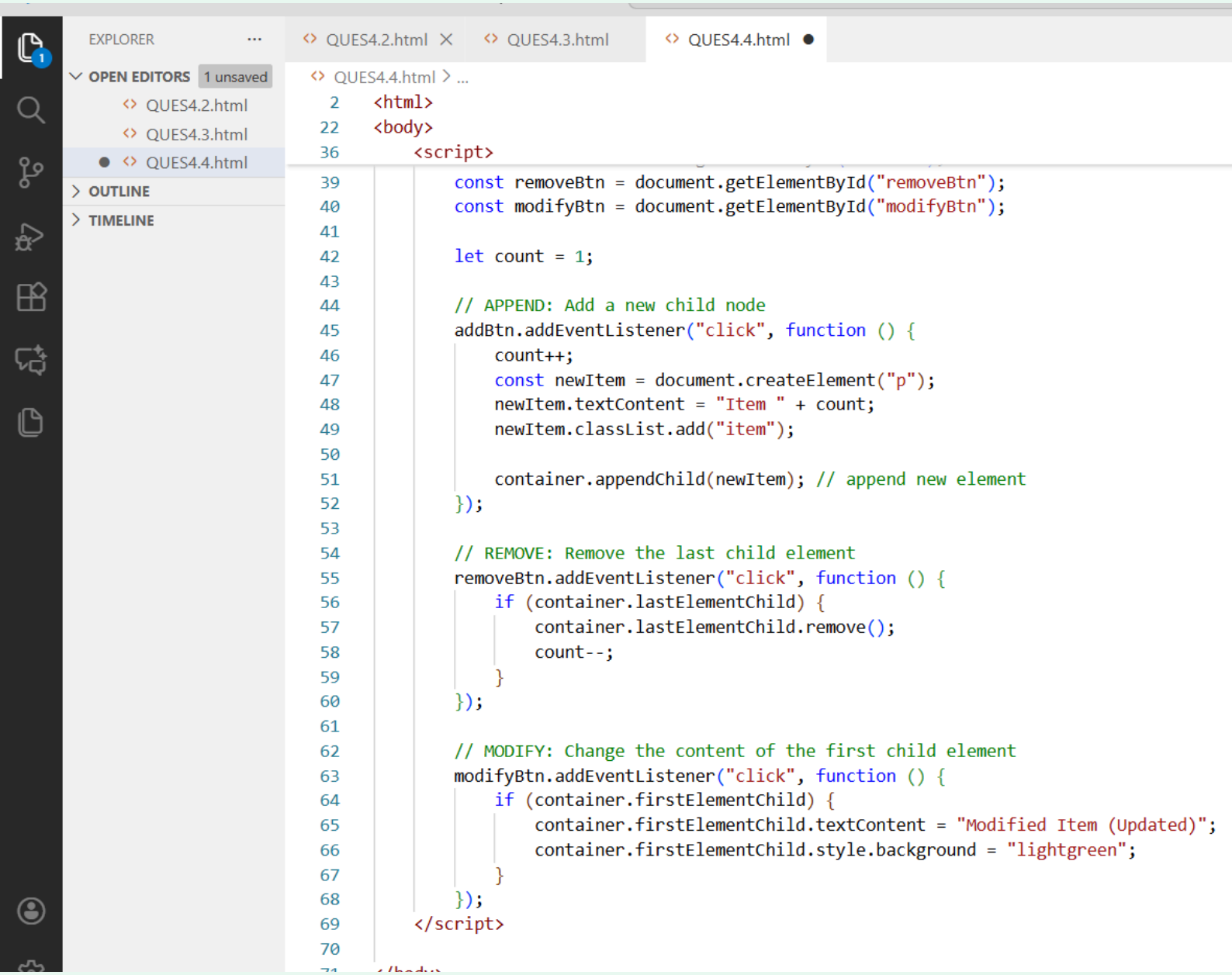
JavaScript allows you to dynamically create and manage HTML elements. The most used DOM functions include:



These techniques are used in:

- Dynamic lists (To-do apps)
- Chat message rendering
- Updating UI sections
- Dashboard widget creation
- Interactive menus

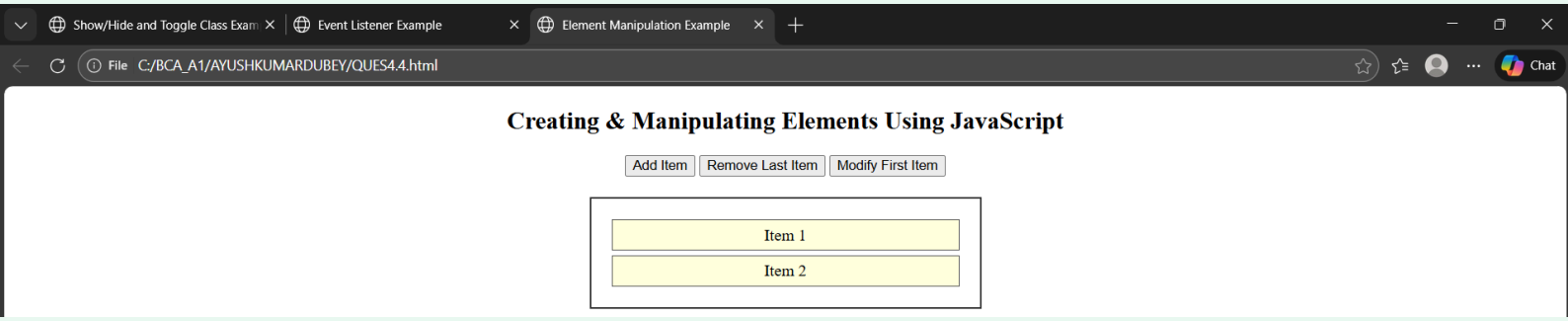
Q.4.4Create and manipulate elements using JavaScript: append, remove, or modify child nodes



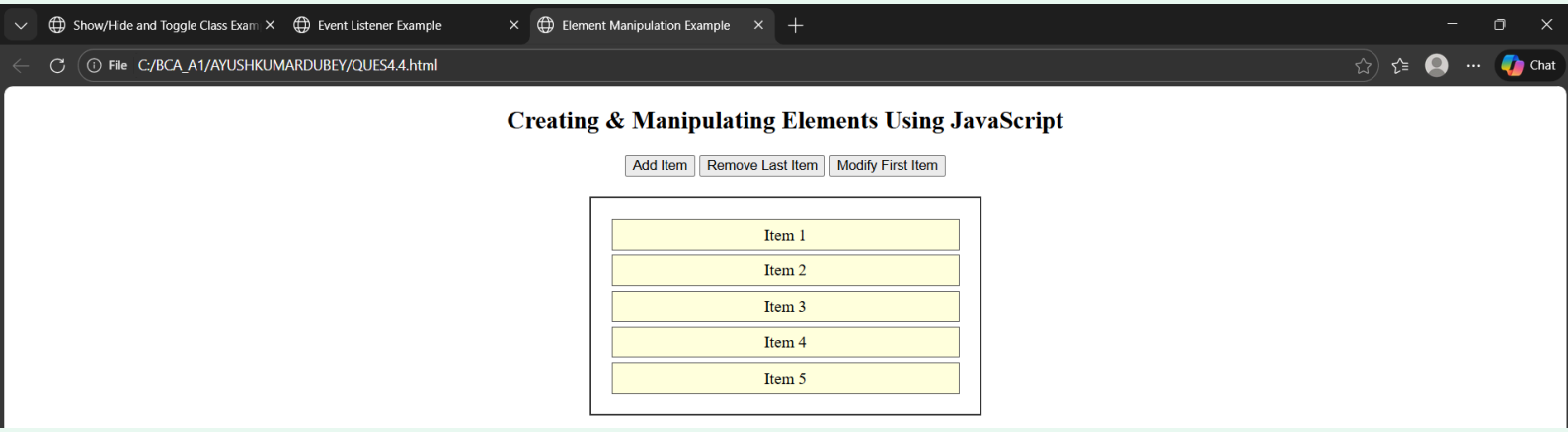
JavaScript's DOM manipulation methods allow developers to create, insert, update, and remove HTML elements dynamically. Understanding createElement, appendChild, removeChild, and innerHTML is essential for developing interactive and responsive web pages.

Q.4.4 Create and manipulate elements using JavaScript: append, remove, or modify child nodes

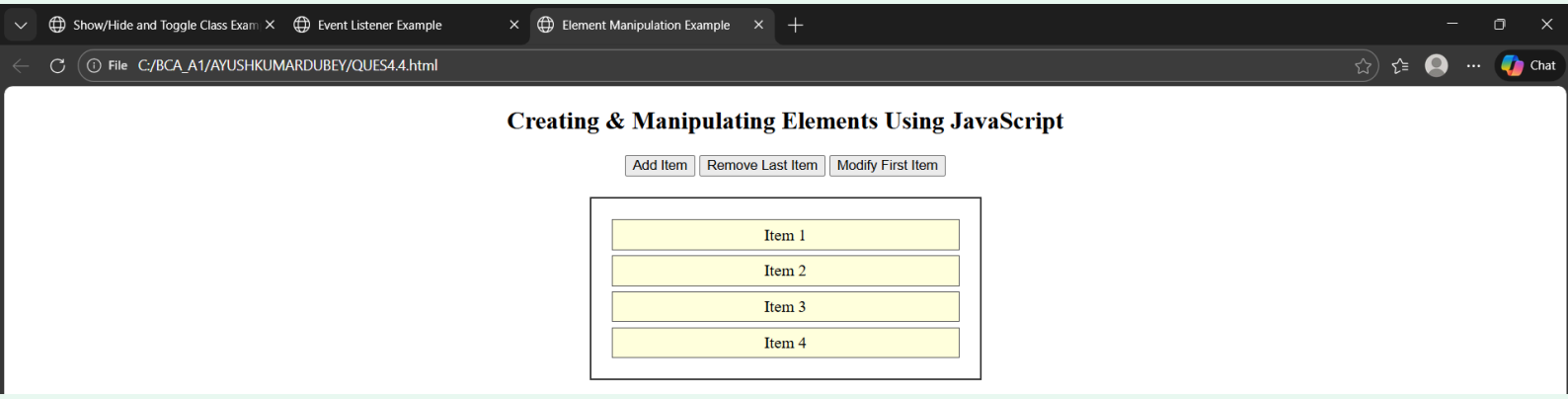
OUTPUT



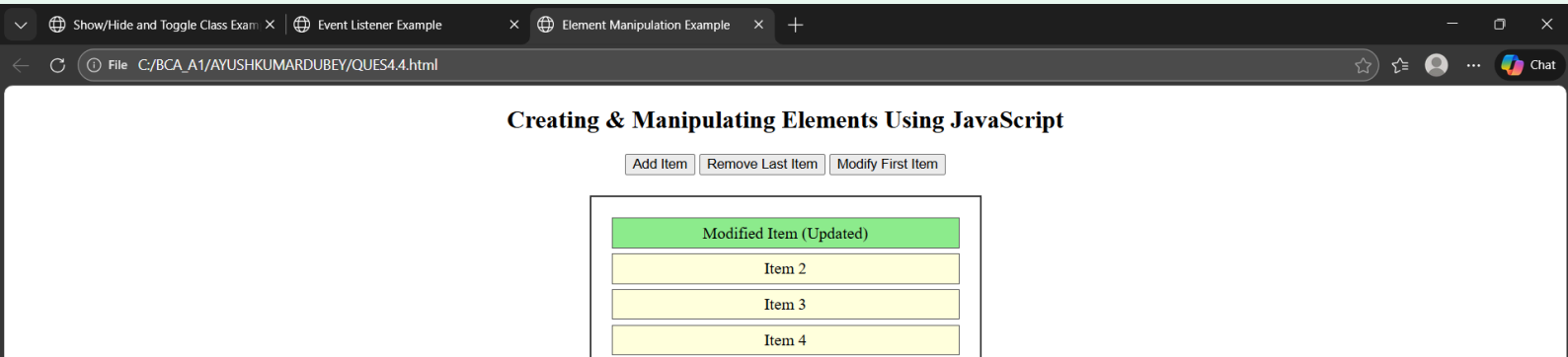
ADDING ITEM



ITEM 5 REMOVED



ITEMS MODIFIED

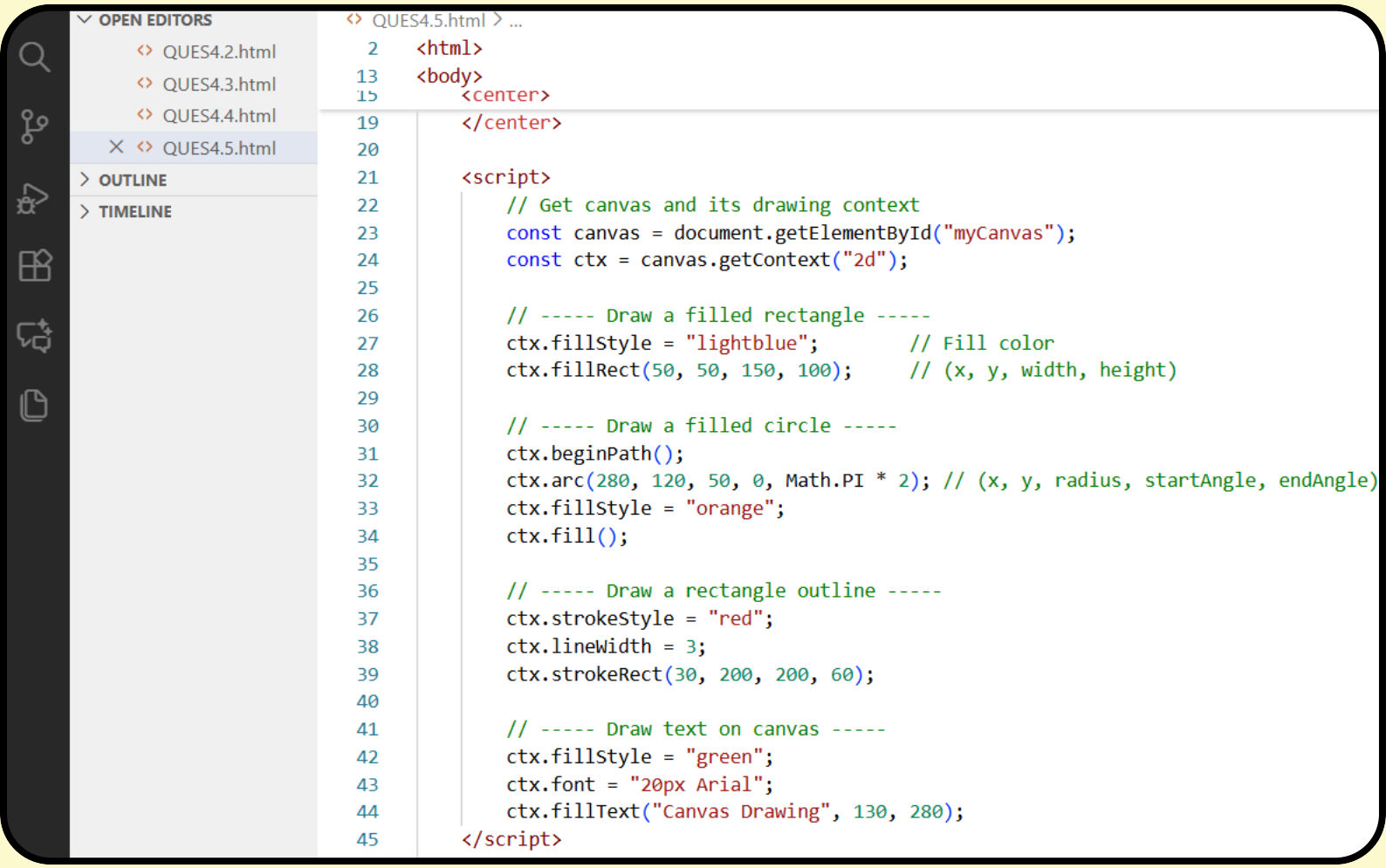


Q.4.5 Create a simple drawing using the <canvas> element: draw shapes and fill colors

Explanation of the Methods

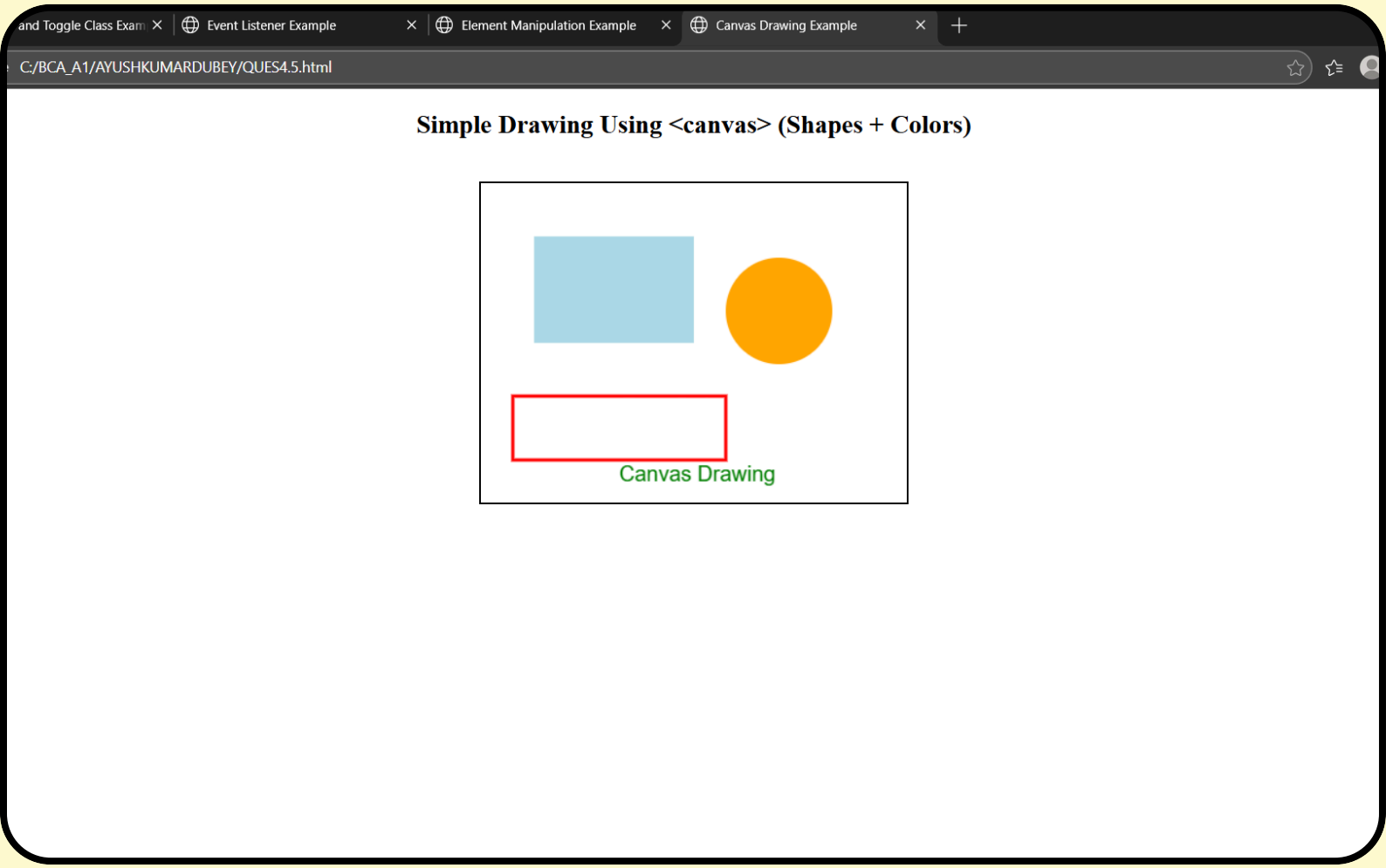
The HTML <canvas> element allows you to draw graphics using JavaScript. With the canvas 2D drawing context, you can create shapes, color fills, lines, text, and even animations.

Q.4.5 Create a simple drawing using the <canvas> element: draw shapes and fill colors



Q.4.5 Create a simple drawing using the <canvas> element: draw shapes and fill colors

OUTPUT



Conclusion

Using the <canvas> API allows you to draw any shape by combining paths, fills, and strokes.
The essential methods you learned—fillRect, strokeRect, beginPath, arc, lineTo, fill, and stroke—form the foundation for more advanced drawings like animations and interactive graphics.

Thank you

for reviewing my assignment.
Your time and guidance are
truly appreciated.

By:-

Ayush Kumar Dubey

RIU:-11741