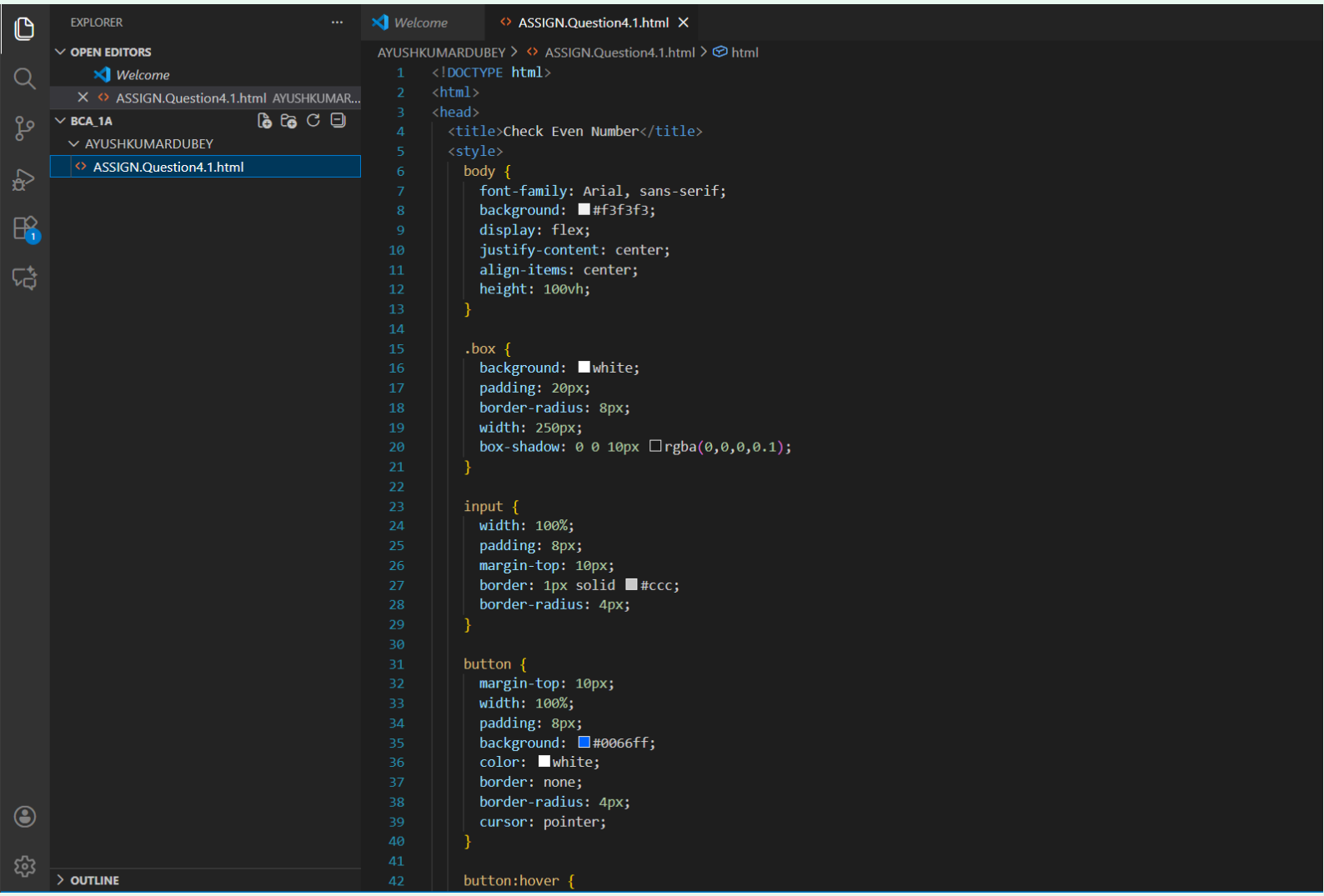
Q.4.1 Write an arrow function isEven (num) that returns true if a number is even.

This task involves creating an arrow function named "isEven(num)" that checks whether a given number is even. The function evaluates the number using the modulus operator and returns "true" if the number is divisible by 2, otherwise "false". This provides a simple and efficient way to test even numbers in JavaScript.



Key Points:

- The function uses the arrow function syntax in JavaScript.
- It takes one parameter: "num".
- A number is even if "num % 2 === 0".
- The function returns a boolean value: "true" for even, "false" for odd.
- Useful for input validation, loops, and filtering arrays.

Q.4.1 Write an arrow function isEven (num) that returns true if a number is even.

```
      background: #004bcc;
    }

    #result {
      margin-top: 15px;
      font-weight: bold;
      text-align: center;
    }
  </style>
</head>

<body>

  <div class="box">
    <h3>Even Number Checker</h3>

    <input type="number" id="numInput" placeholder="Enter a number">

    <button onclick="checkEven()">Check</button>

    <div id="result"></div>
  </div>

  <script>
    // arrow function to check even number
    const isEven = (num) => {
      return num % 2 === 0;
    };

    function checkEven() {
      const value = document.getElementById("numInput").value;
      const result = document.getElementById("result");

      if (value === "") {
        result.textContent = "Please enter a number.";
        return;
      }

      if (isEven(Number(value))) {
        result.textContent = value + " is even ✓";
      } else {
        result.textContent = value + " is not even ✗";
      }
    }
  </script>

</body>
</html>
```

Example:

If the input is: "isEven(10)"

Calculation: "10 % 2 === 0"

Output: true

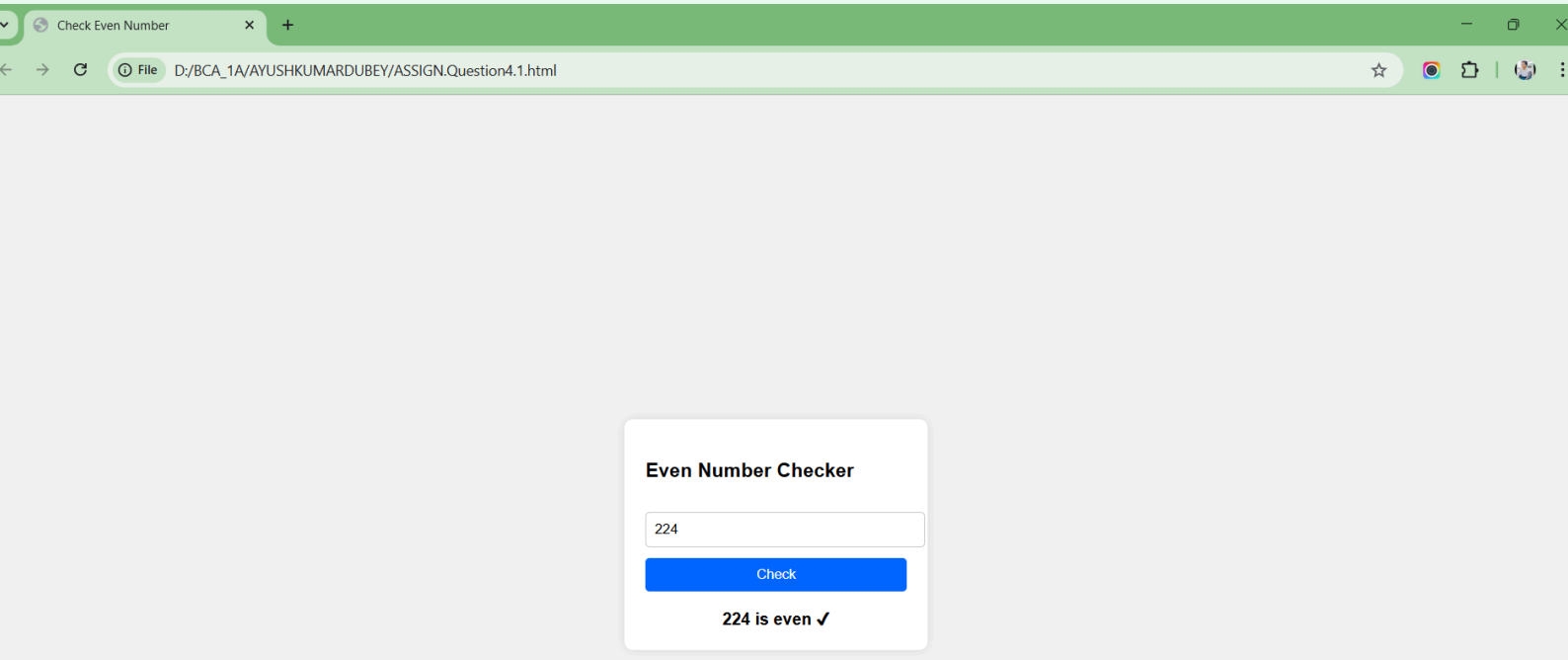
If the input is: "isEven(7)"

Calculation: "7 % 2 === 1"

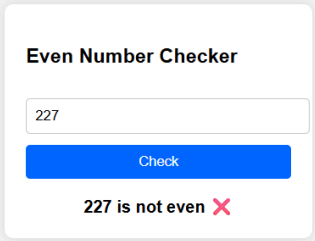
Output: false

Q.4.1 Write an arrow function isEven (num) that returns true if a number is even.

OUTPUT



Here's the result



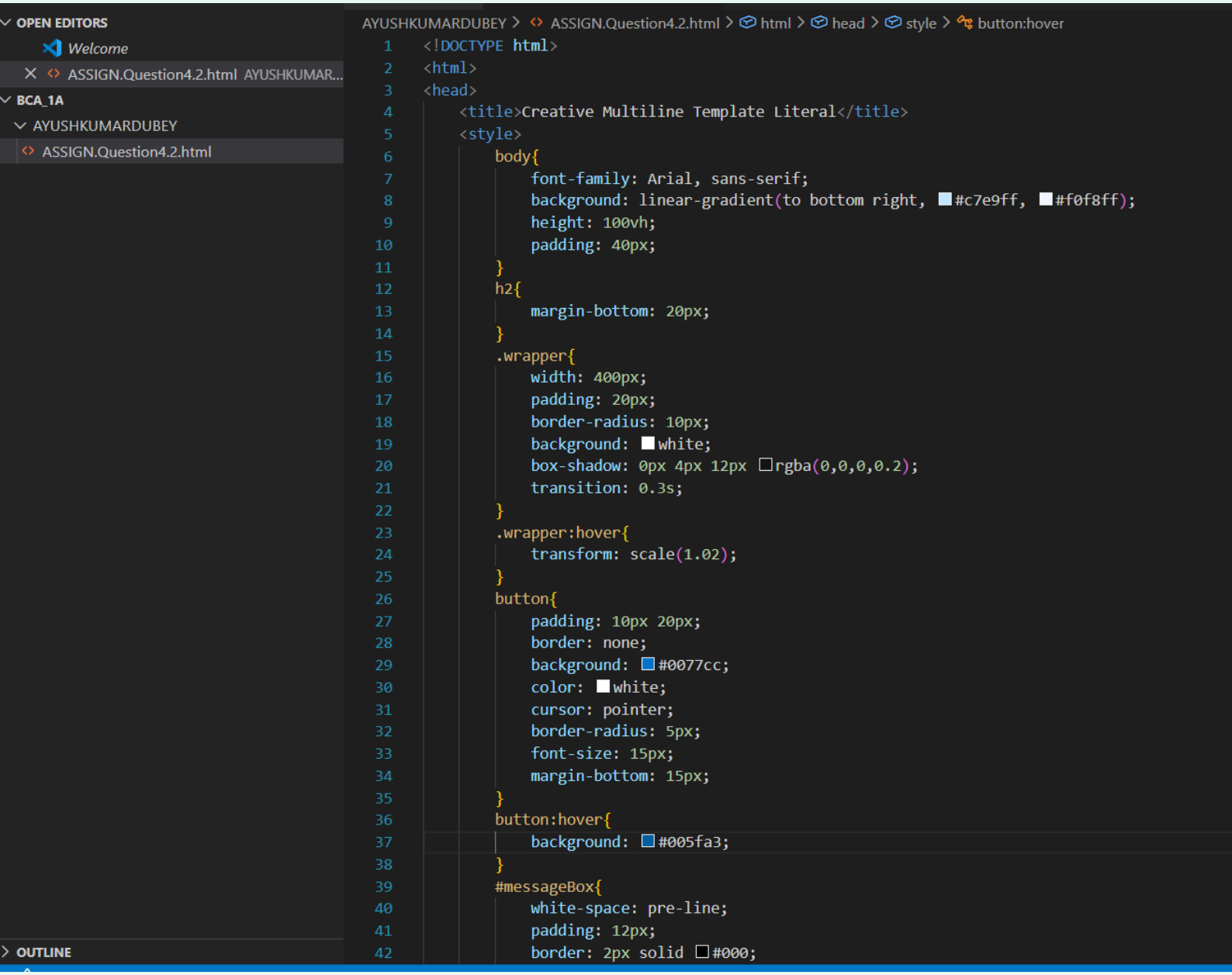
ON ODD NUMBER IT WON'T WORK

Conclusion:

The arrow function successfully determines whether a given number is even by applying the modulus check and returning a clear true/false result, making it simple and reliable for use in various programming tasks.

Q.4.2 Create a multiline string using template literals showing a 3-line message.

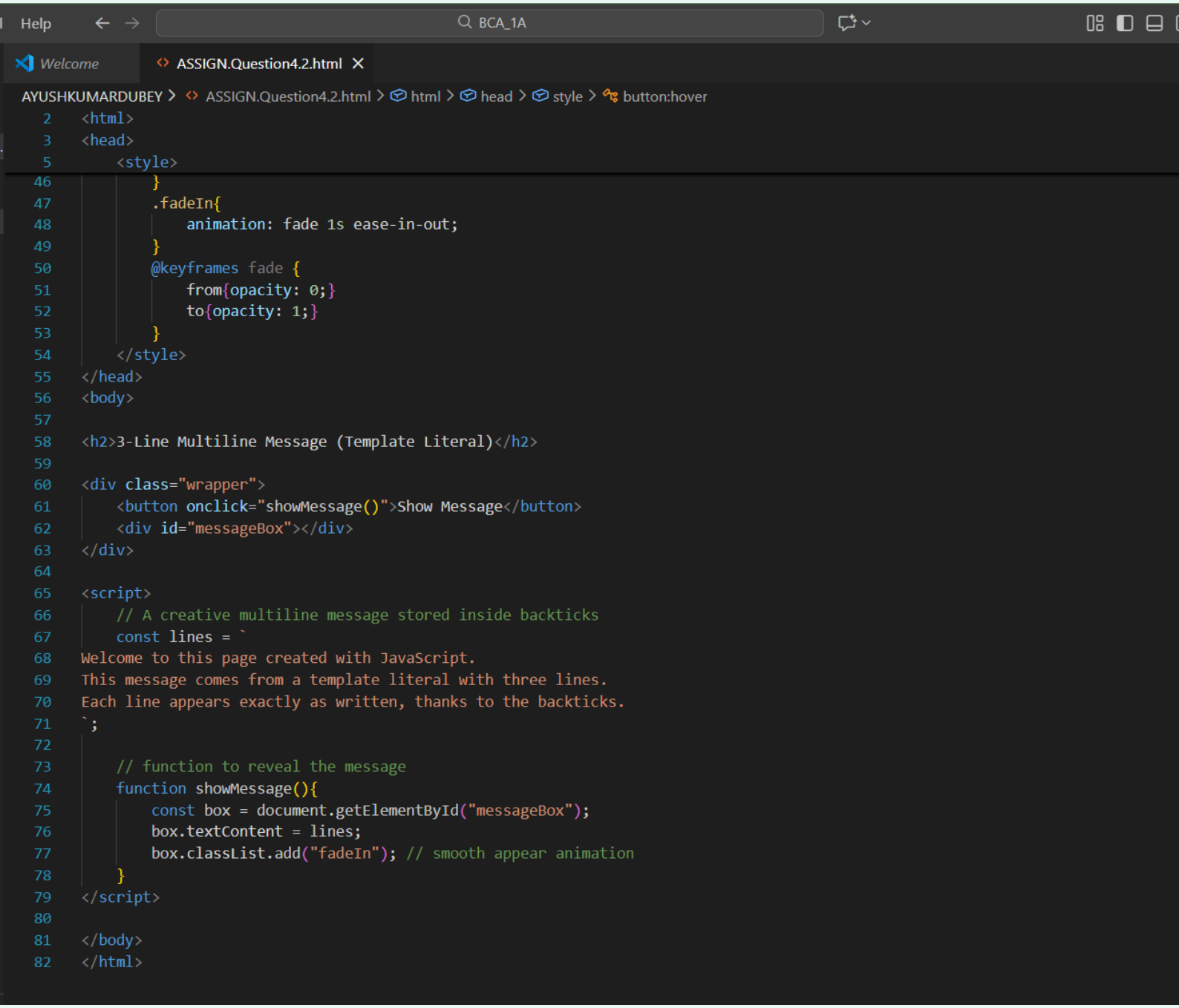
This task demonstrates how to create a multiline string in JavaScript using template literals. Template literals allow text to span several lines by using backticks, making it possible to write clean and readable multiline messages without special characters or concatenation.



Key Points:

- Template literals use backticks (`) instead of quotes.
- They support multiline text naturally.
- No need for \n or string concatenation.
- Ideal for paragraphs, formatted messages, and readable output.
- Improves clarity when displaying multiple lines of text.

Q.4.2 Create a multiline string using template literals showing a 3-line message.

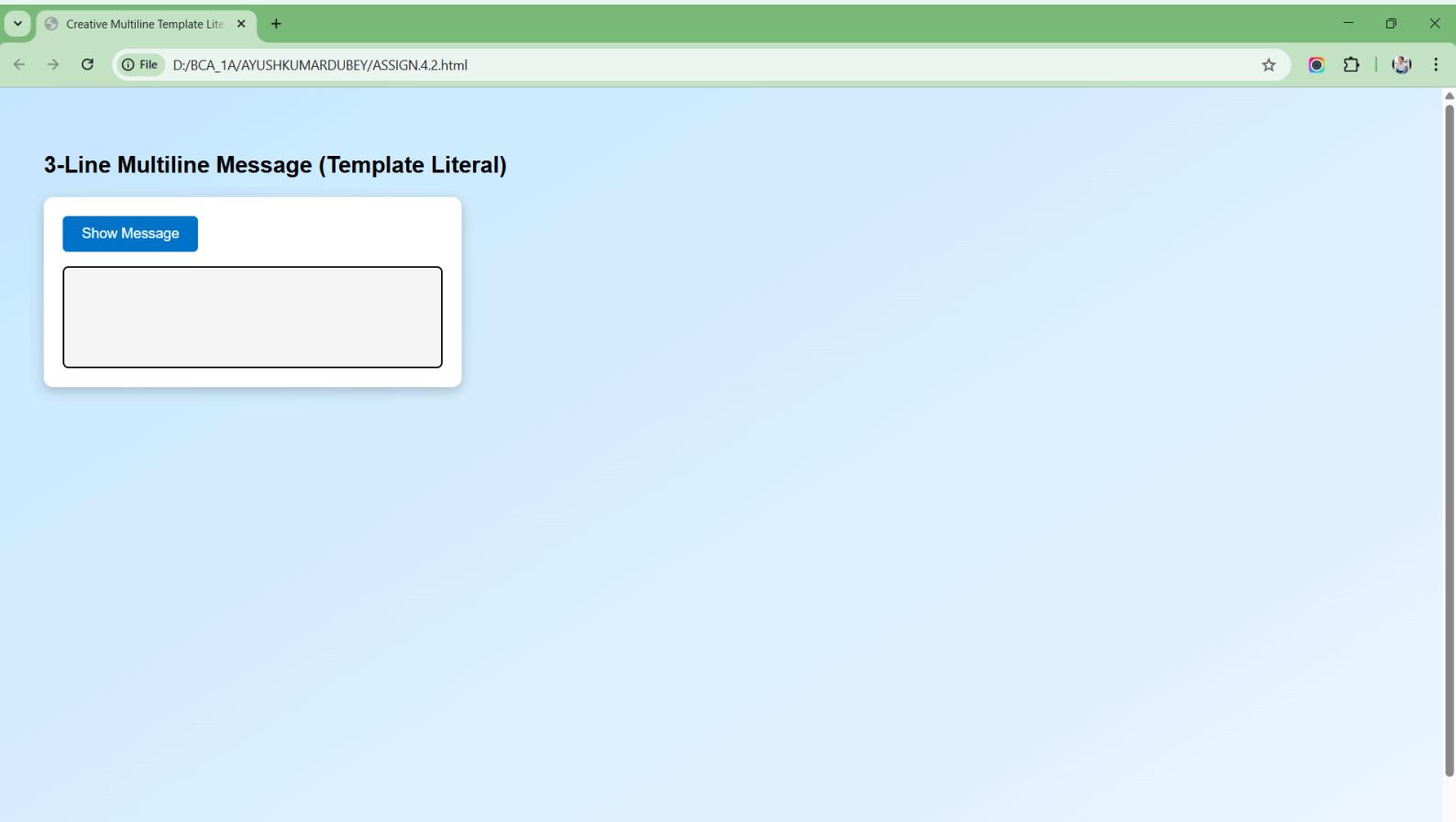


EXAMPLE:

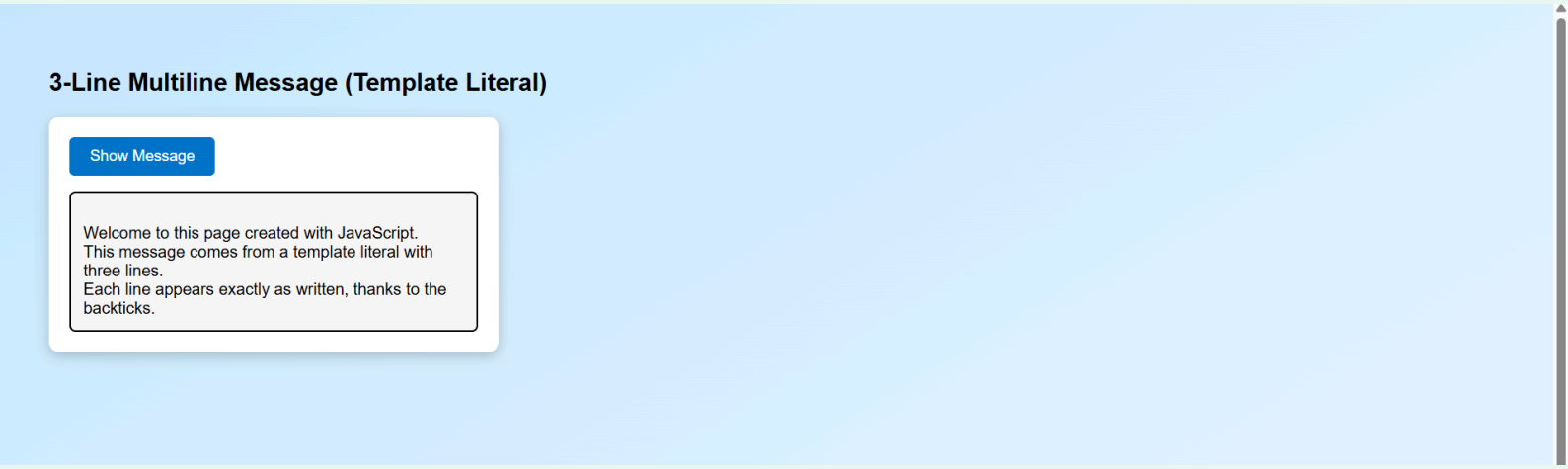
```
const message =
"This is line one.
This is line two.
This is line three.";

console.log(message);
```

Q.4.2 Create a multiline string using template literals showing a 3-line message.

OUTPUT



After clicking on show message it will appear like:-



Conclusion;

Template literals make it easy to create multiline strings in JavaScript by allowing text to span multiple lines naturally. This approach results in cleaner code and clearer output, especially when working with messages or blocks of text.

Q. 4.3 Merge two arrays using the spread operator

This task demonstrates how to merge two arrays in JavaScript using the spread operator. The spread operator allows elements from multiple arrays to be expanded into a new array easily. This method creates a clean, readable, and efficient way to combine arrays without using loops or complex functions.

AYUSHKUMARDUBEY > ASSIGN.4.3.html > html

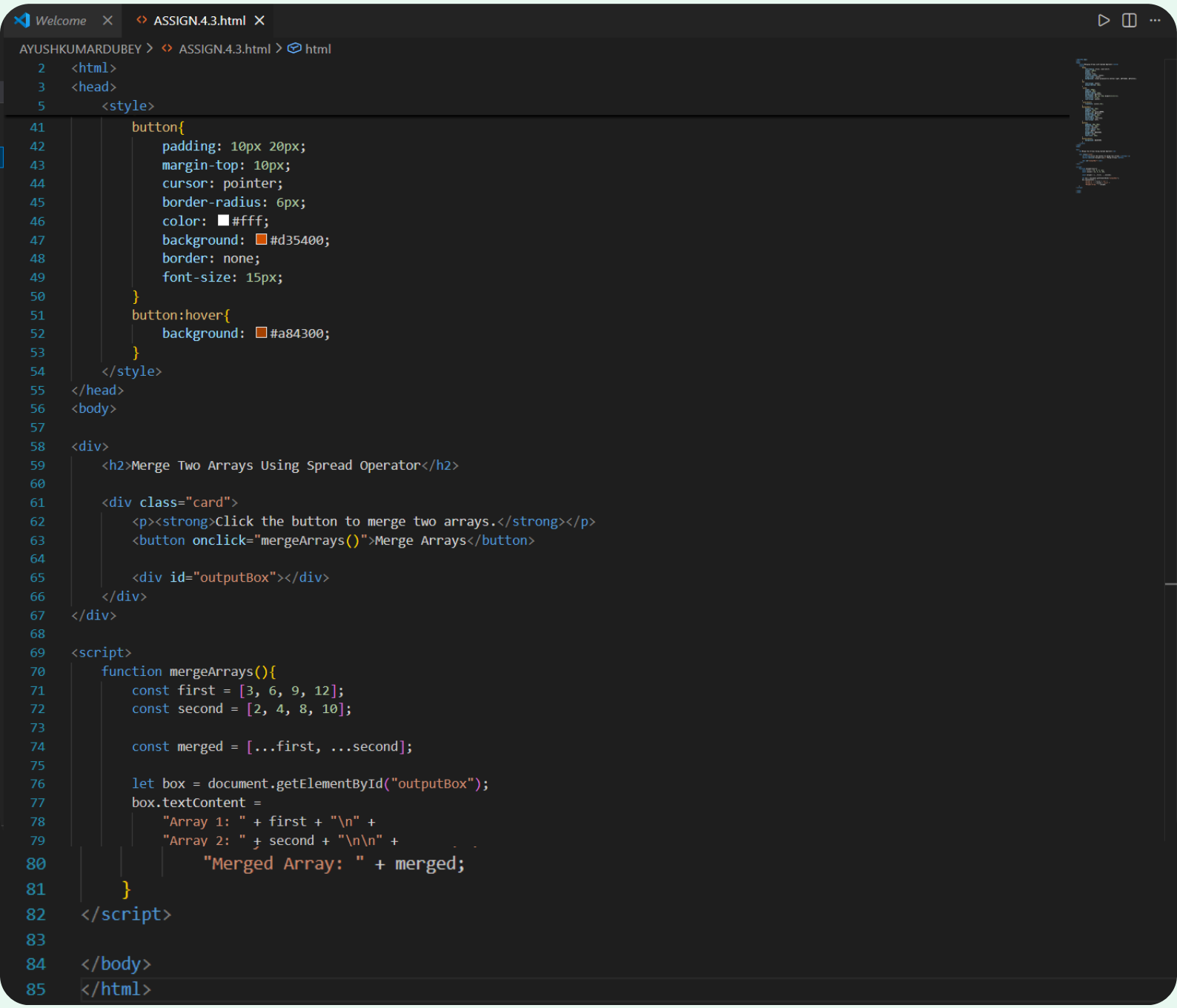
```html
<!DOCTYPE html>
<html>
<head>
    <title>Merging Arrays with Spread Operator</title>
    <style>
        body{
            font-family: Arial, sans-serif;
            height: 100vh;
            margin: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            background: linear-gradient(to bottom right, #ffe8d6, #fafafa);
        }
        h2{
            text-align: center;
            margin-bottom: 20px;
        }
        .card{
            width: 430px;
            padding: 20px;
            border-radius: 12px;
            background: #ffffff;
            box-shadow: 0px 4px 14px rgba(0,0,0,0.2);
            transition: 0.25s;
            text-align: center;
        }
        .card:hover{
            transform: scale(1.02);
        }
        #outputBox{
            margin-top: 15px;
            padding: 12px;
            border: 2px solid #000;
            background: #f1f1f1;
            border-radius: 6px;
            min-height: 70px;
            white-space: pre-line;
            text-align: left;
        }
        button{
            padding: 10px 20px;
```

Key Points:

- The spread operator is written as three dots (...).
- It expands array elements into another array.
- Useful for merging, copying, or adding elements.
- Prevents modification of the original arrays.
- Produces a new combined array in a simple and clean way.

Q. 4.3 Merge two arrays using the spread operator

This task demonstrates how to merge two arrays in JavaScript using the spread operator. The spread operator allows elements from multiple arrays to be expanded into a new array easily. This method creates a clean, readable, and efficient way to combine arrays without using loops or complex functions.

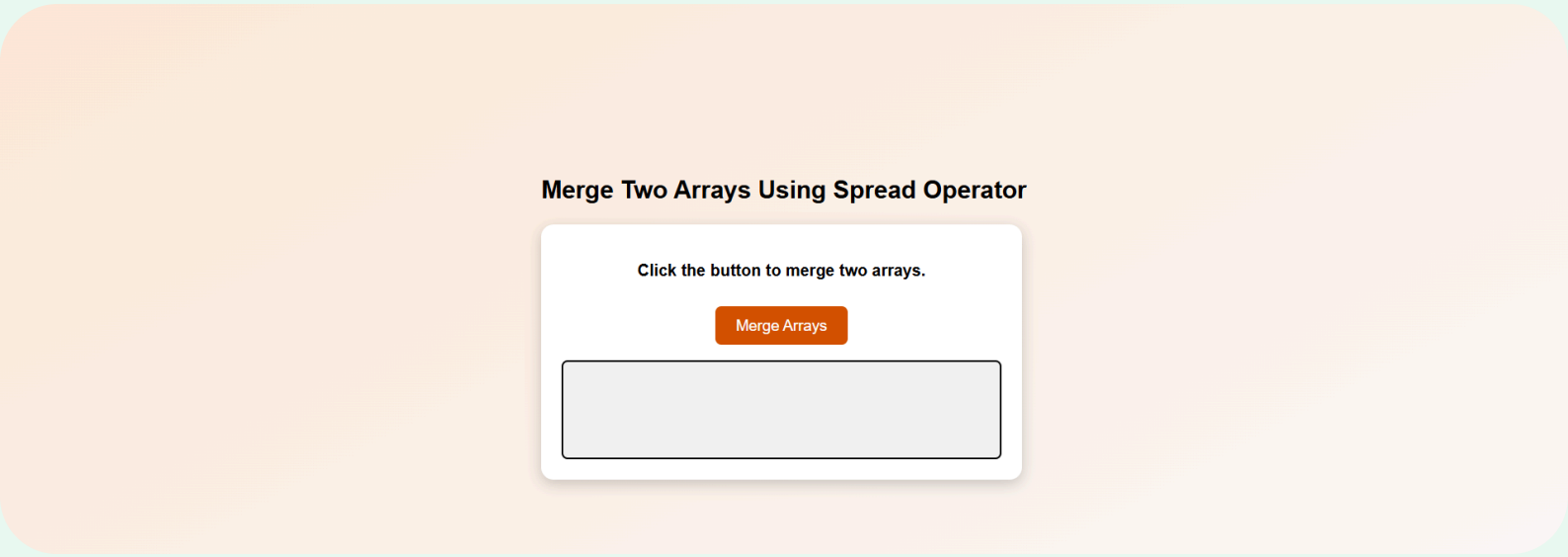


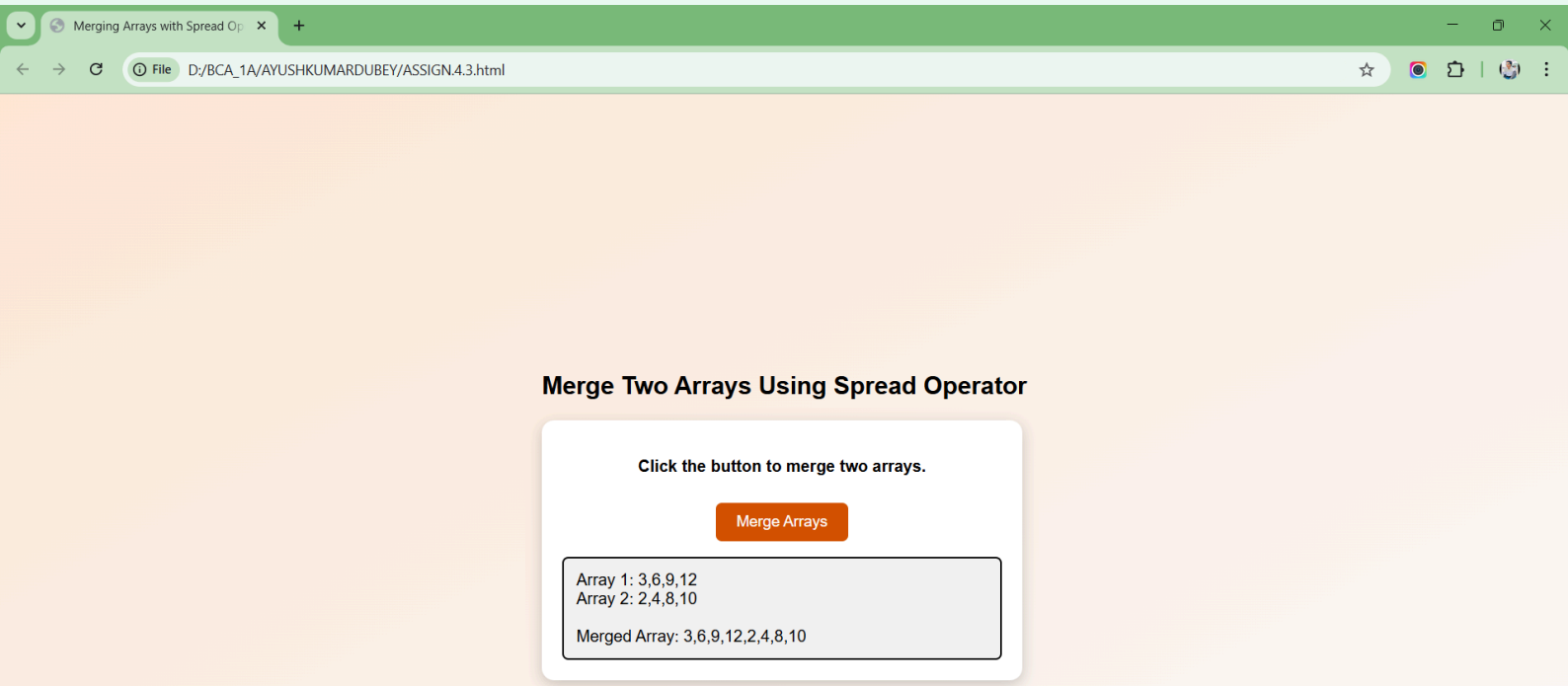
EXAMPLE:

```
const arr1 = [1, 2, 3];
const arr2 = [4, 5, 6];

const mergedArray = [...arr1, ...arr2];

console.log(mergedArray);
// Output: [1, 2, 3, 4, 5, 6]
```

Q. 4.3 Merge two arrays using the spread operator

output :



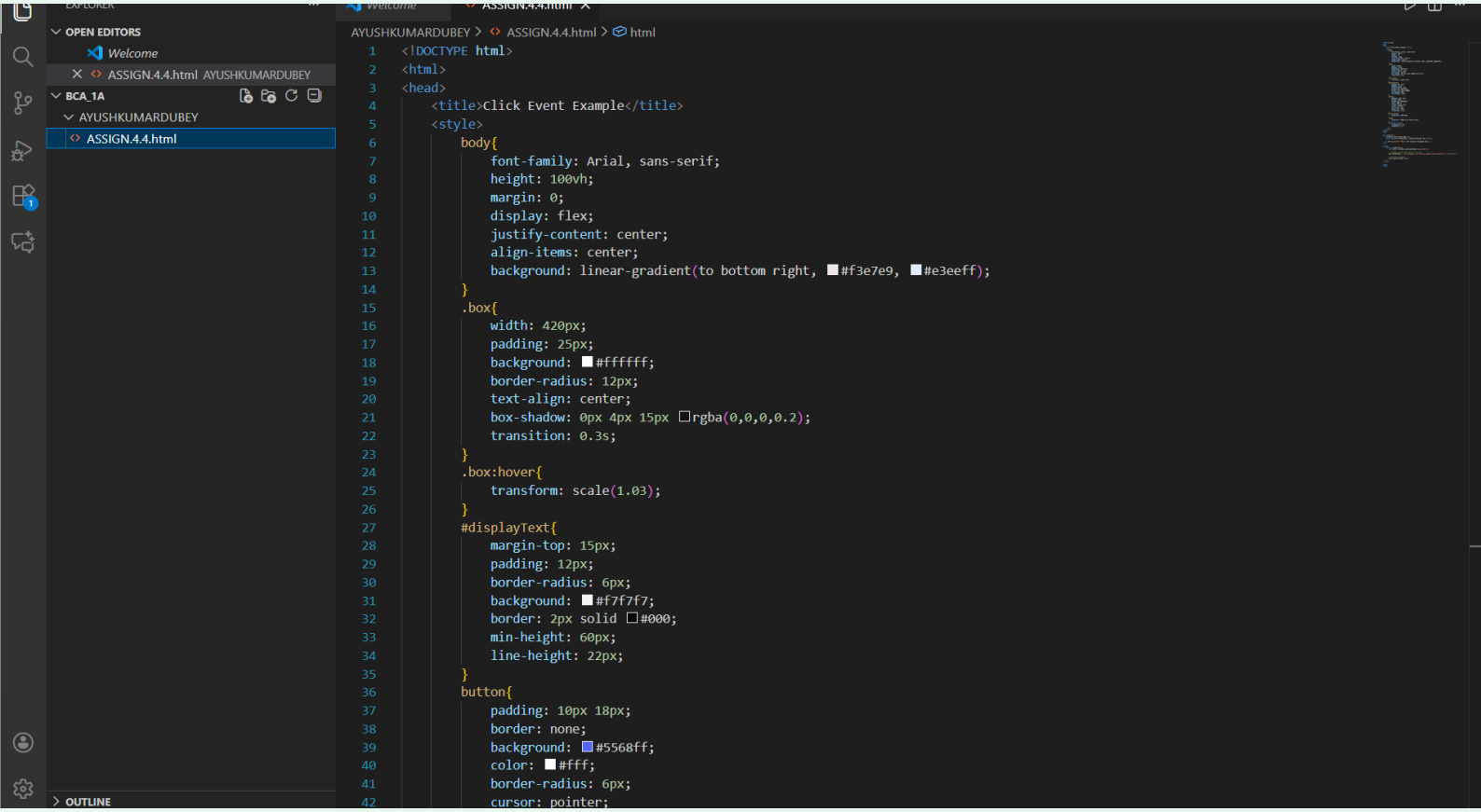


CONCLUSION:

Using the spread operator is an efficient and modern way to merge two arrays in JavaScript. It provides clean syntax preserves the original arrays, and produces a new combined array with minimal code.

Q. 4.4 Add a click event to a button that changes the text of a paragraph.

This task involves creating a button that updates the text inside a paragraph when clicked. By attaching a click event to the button through JavaScript, the paragraph content can be changed instantly. This demonstrates how events allow webpages to respond to user actions and make them more interactive.



Key Points:

- A click event runs a function when the button is pressed.

- The paragraph is selected using an ID or query selector.

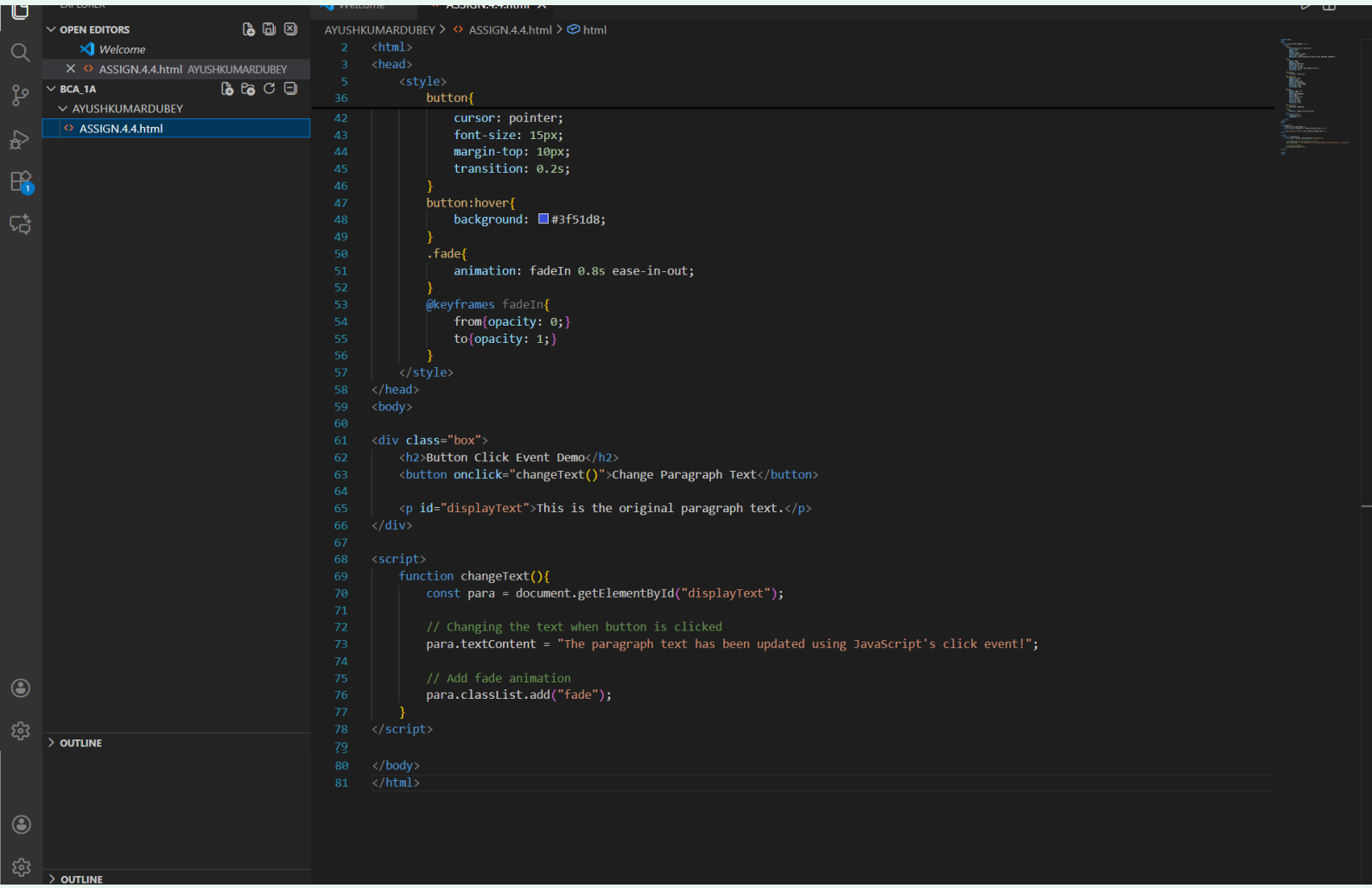
- The text is updated by modifying the paragraph's innerHTML or textContent.

- Helps understand basic DOM interaction.

- Useful for creating interactive web pages.

Q. 4.4 Add a click event to a button that changes the text of a paragraph.

This task involves creating a button that updates the text inside a paragraph when clicked. By attaching a click event to the button through JavaScript, the paragraph content can be changed instantly. This demonstrates how events allow webpages to respond to user actions and make them more interactive.



<!-- HTML -->

Original text

<button id="changeBtn">Change Text</button><!-- JavaScript -->

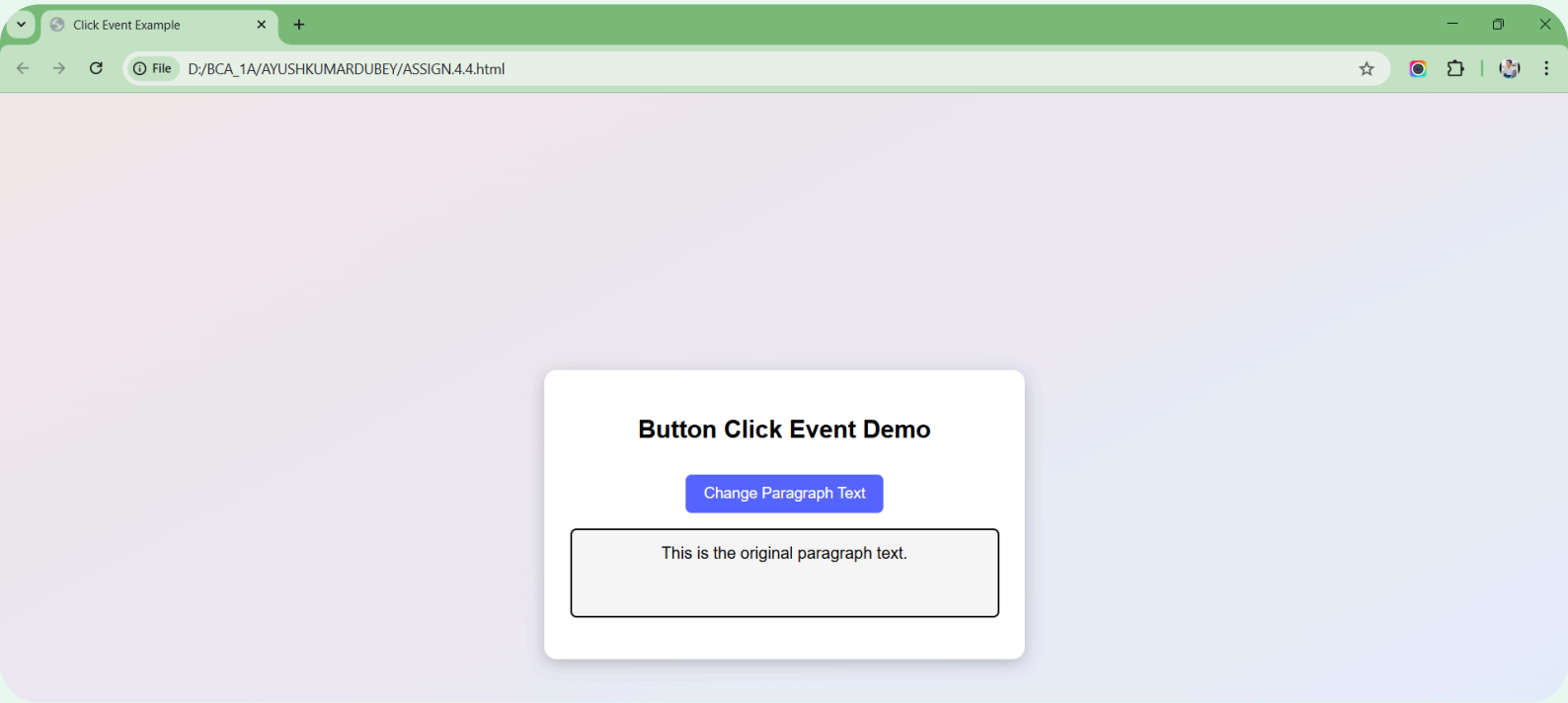
document.getElementById("changeBtn").addEventListener("click", function() {

document.getElementById("msg").textContent = "The text has been changed!";

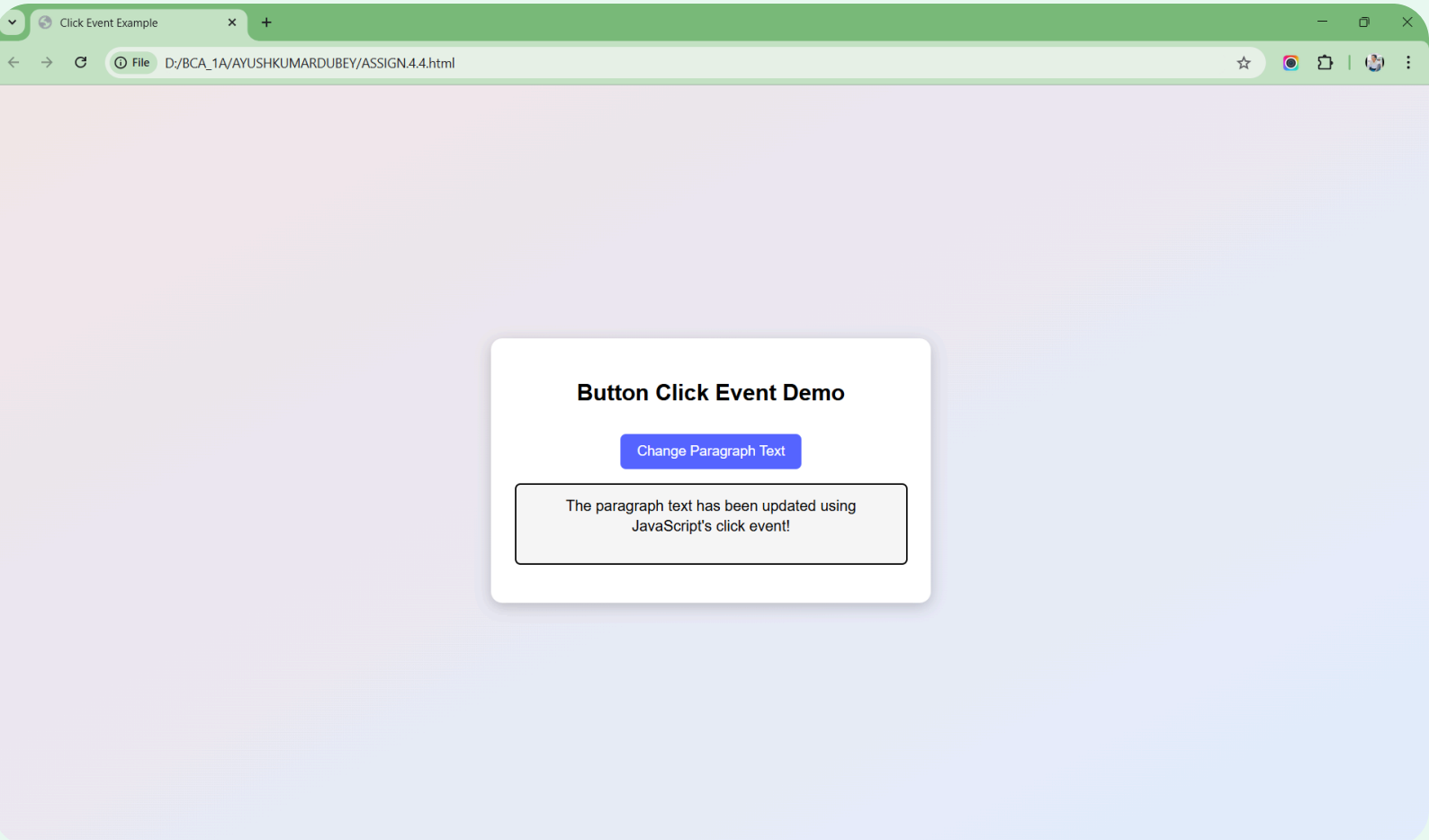
});

Q. 4.4 Add a click event to a button that changes the text of a paragraph.

OUTPUT:



After clicking on change paragraph text



CONCLUSION:

By adding a click event to the button, the webpage responds to user interaction and updates the paragraph's content. This shows how JavaScript connects elements on a page and allows simple actions to create dynamic changes.