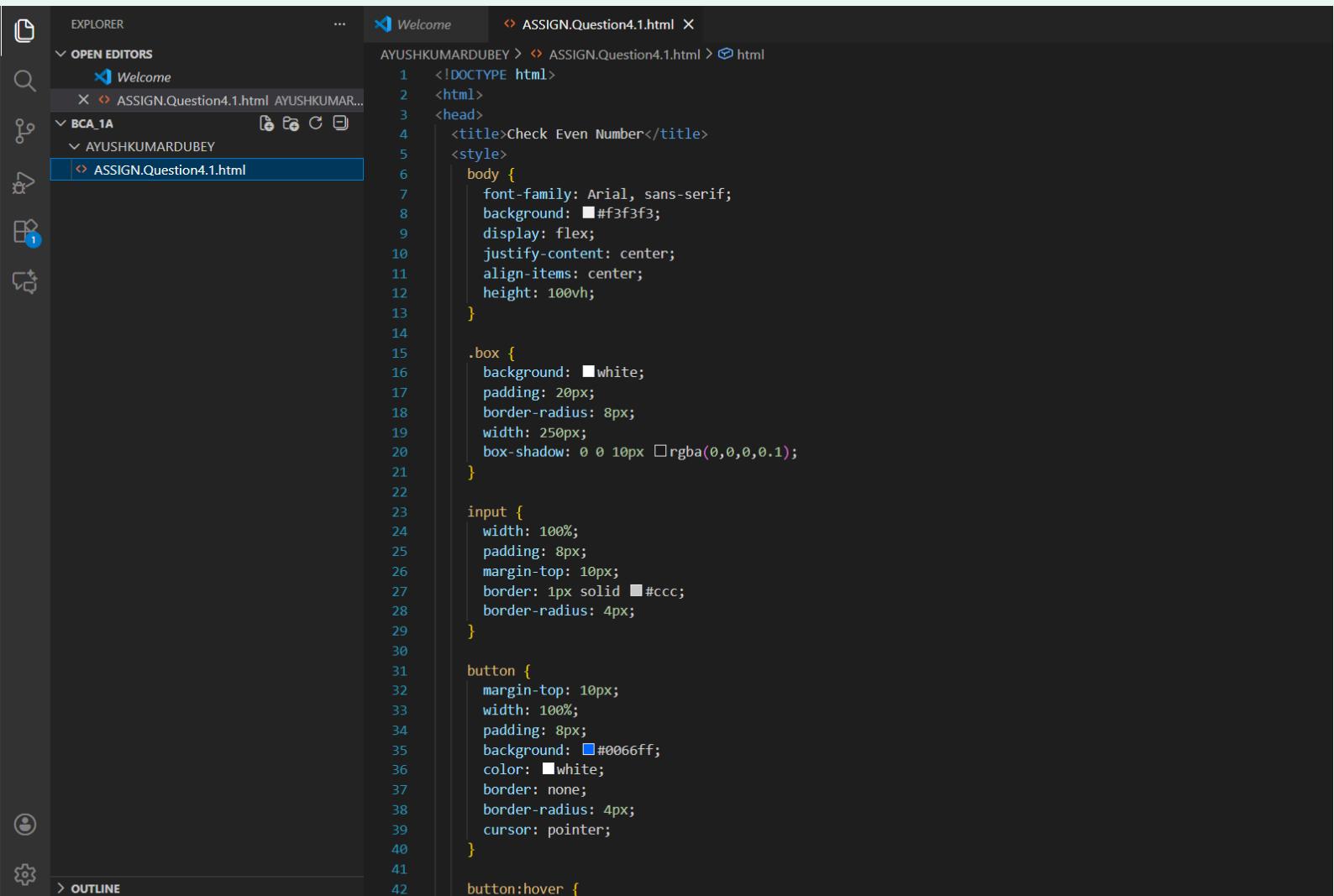


## Q.4.1 Write an arrow function isEven (num) that returns true if a number is even.

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.



The screenshot shows the Visual Studio Code interface. The left sidebar has icons for Explorer, Open Editors, BCA\_1A, and AYUSHKUMARDUBEY. The Open Editors section shows 'Welcome' and 'ASSIGN.Question4.1.html'. The main area is titled 'Welcome' and shows the content of 'ASSIGN.Question4.1.html'. The code is a CSS file with the following content:

```
<!DOCTYPE html>
<html>
<head>
<title>Check Even Number</title>
<style>
body {
    font-family: Arial, sans-serif;
    background: #f3f3f3;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.box {
    background: white;
    padding: 20px;
    border-radius: 8px;
    width: 250px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
}

input {
    width: 100%;
    padding: 8px;
    margin-top: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
}

button {
    margin-top: 10px;
    width: 100%;
    padding: 8px;
    background: #0066ff;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}

button:hover {
```

### 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.

```
43 |     background: #004bcc;
44 |
45 |
46 | #result {
47 |     margin-top: 15px;
48 |     font-weight: bold;
49 |     text-align: center;
50 |
51 | }
52 | </style>
53 | </head>
54 | <body>
55 |
56 | <div class="box">
57 |     <h3>Even Number Checker</h3>
58 |
59 |     <input type="number" id="numInput" placeholder="Enter a number">
60 |
61 |     <button onclick="checkEven()">Check</button>
62 |
63 |     <div id="result"></div>
64 | </div>
65 |
66 | <script>
67 |     // arrow function to check even number
68 |     const isEven = (num) => {
69 |         return num % 2 === 0;
70 |     };
71 |
72 |     function checkEven() {
73 |         const value = document.getElementById("numInput").value;
74 |         const result = document.getElementById("result");
75 |
76 |         if (value === "") {
77 |             result.textContent = "Please enter a number.";
78 |             return;
79 |         }
80 |
81 |         if (isEven(Number(value))) {
82 |             result.textContent = value + " is even ✓";
83 |         } else {
84 |             result.textContent = value + " is not even ✗";
85 |         }
86 |     }
87 | </script>
88 |
89 | </body>
90 | </html>
```

**Example:**

If the input is: "isEven(10)"

Calculation: "10 % 2 === 0"

Output: true

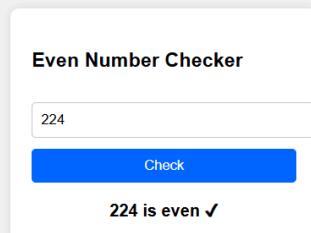
If the input is: "isEven(7)"

Calculation: "7 % 2 === 1"

Output: false

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

## OUTPUT

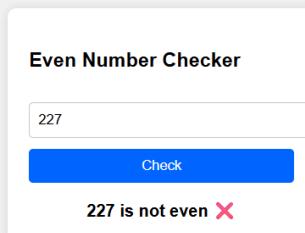


Even Number Checker

Check

224 is even ✓

Here's the result



Even Number Checker

Check

227 is not even ✗

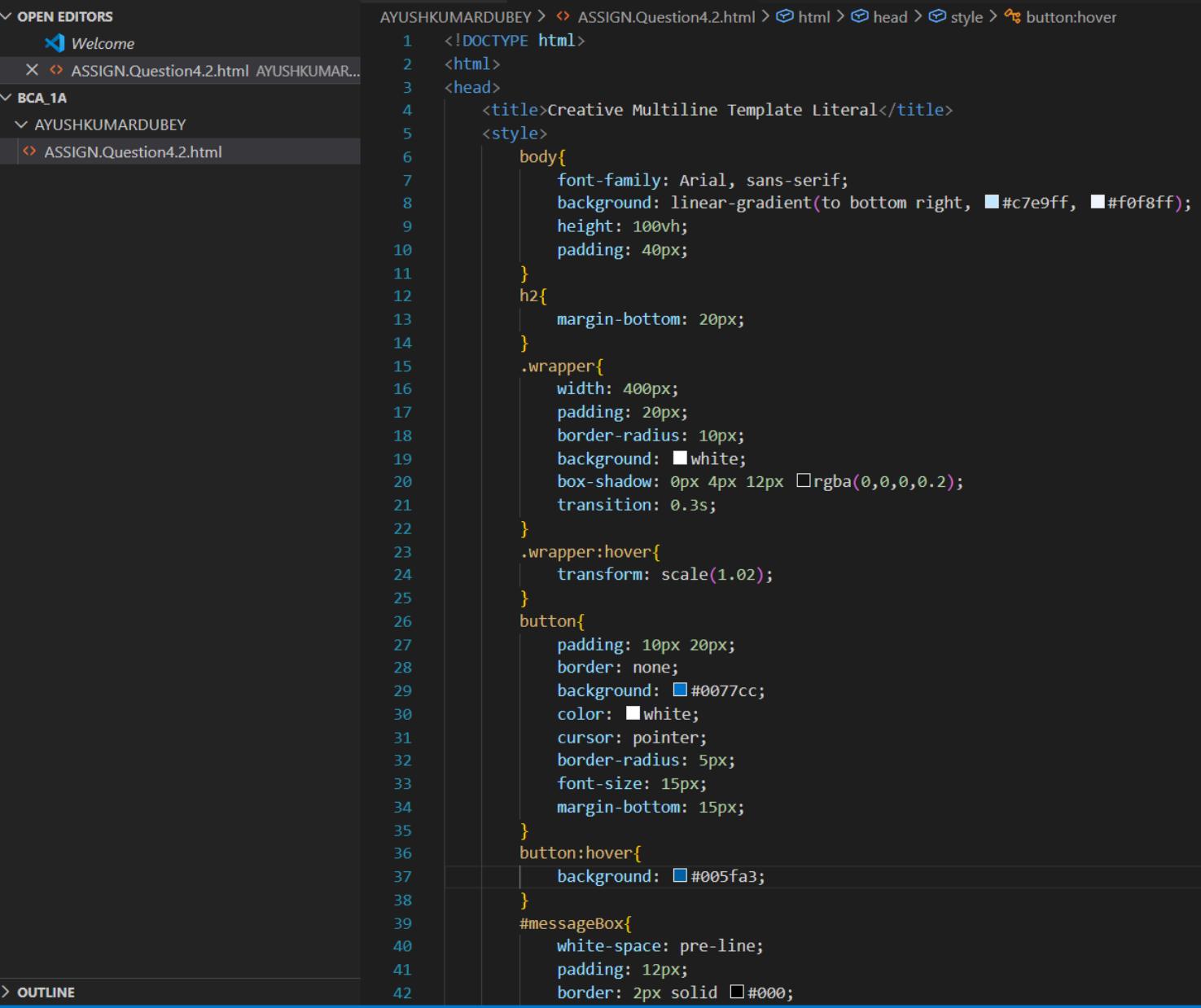
ON ODD NUMBER IT WON'T WORK

## 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.

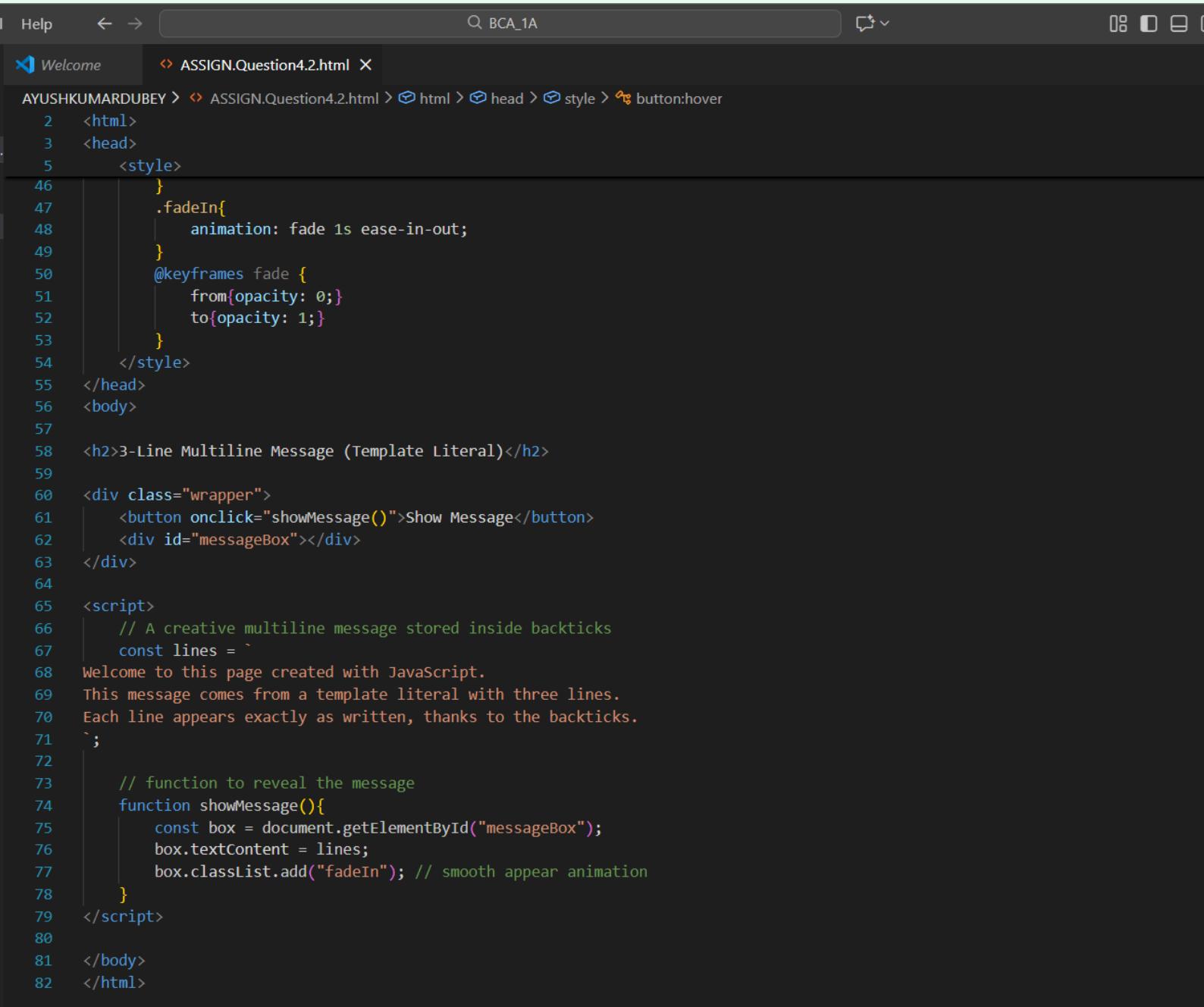


```
AYUSHKUMARDUBEY > ASSIGN.Question4.2.html > html > head > style > button:hover
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Creative Multiline Template Literal</title>
5      <style>
6          body{
7              font-family: Arial, sans-serif;
8              background: linear-gradient(to bottom right, #c7e9ff, #f0f8ff);
9              height: 100vh;
10             padding: 40px;
11         }
12         h2{
13             margin-bottom: 20px;
14         }
15         .wrapper{
16             width: 400px;
17             padding: 20px;
18             border-radius: 10px;
19             background: white;
20             box-shadow: 0px 4px 12px rgba(0,0,0,0.2);
21             transition: 0.3s;
22         }
23         .wrapper:hover{
24             transform: scale(1.02);
25         }
26         button{
27             padding: 10px 20px;
28             border: none;
29             background: #0077cc;
30             color: white;
31             cursor: pointer;
32             border-radius: 5px;
33             font-size: 15px;
34             margin-bottom: 15px;
35         }
36         button:hover{
37             background: #005fa3;
38         }
39         #messageBox{
40             white-space: pre-line;
41             padding: 12px;
42             border: 2px solid #000;
```

### 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.



```
AYUSHKUMARDUBEY > ASSIGN.Question4.2.html > html > head > style > button:hover
2  <html>
3  <head>
5   <style>
46    }
47    .fadeIn{
48      animation: fade 1s ease-in-out;
49    }
50    @keyframes fade {
51      from{opacity: 0;}
52      to{opacity: 1;}
53    }
54  </style>
55 </head>
56 <body>
57
58 <h2>3-Line Multiline Message (Template Literal)</h2>
59
60 <div class="wrapper">
61   <button onclick="showMessage()">Show Message</button>
62   <div id="messageBox"></div>
63 </div>
64
65 <script>
66   // A creative multiline message stored inside backticks
67   const lines = `
68 Welcome to this page created with JavaScript.
69 This message comes from a template literal with three lines.
70 Each line appears exactly as written, thanks to the backticks.
71 `;
72
73   // function to reveal the message
74   function showMessage(){
75     const box = document.getElementById("messageBox");
76     box.textContent = lines;
77     box.classList.add("fadeIn"); // smooth appear animation
78   }
79 </script>
80
81 </body>
82 </html>
```

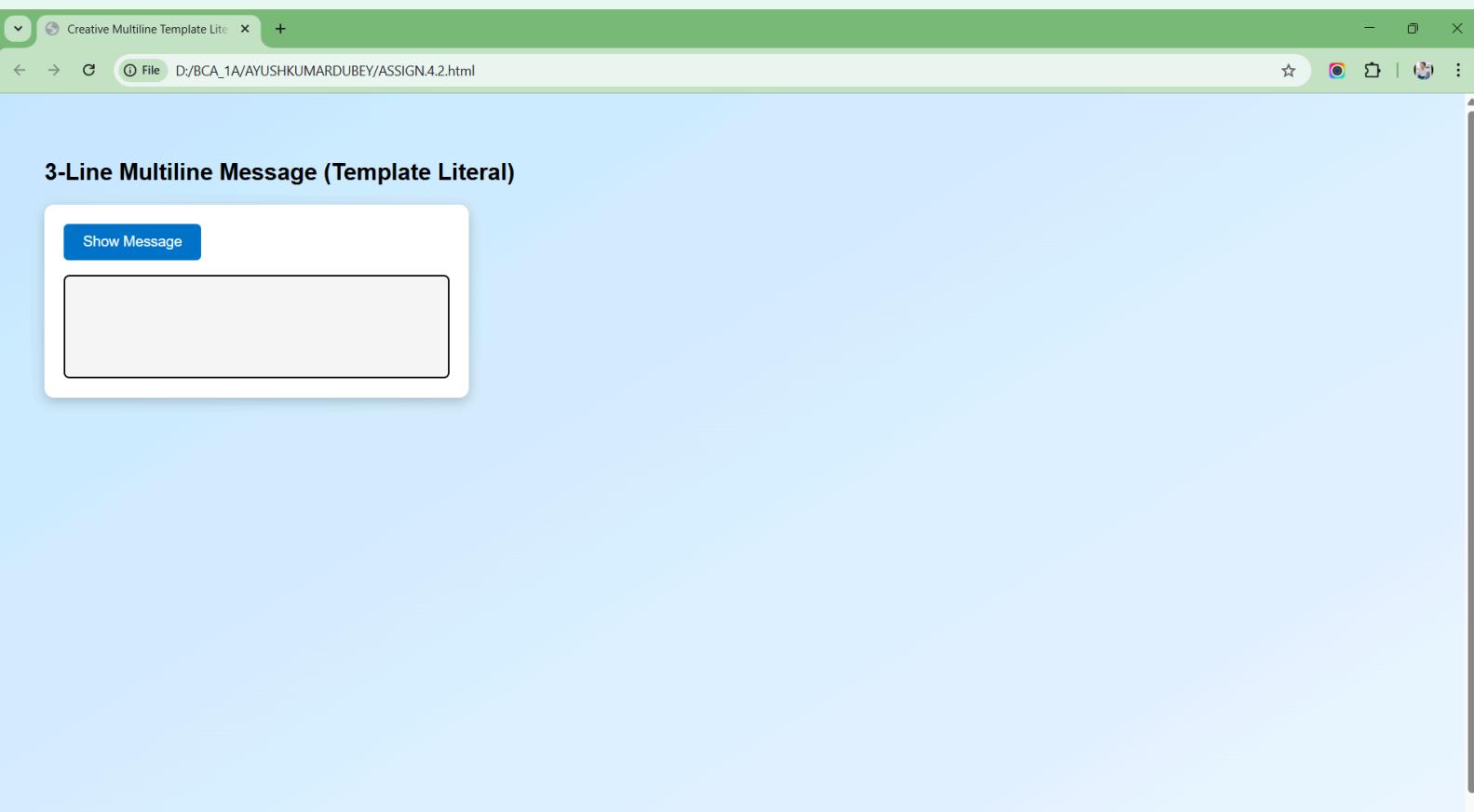
EXAMPLE:

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

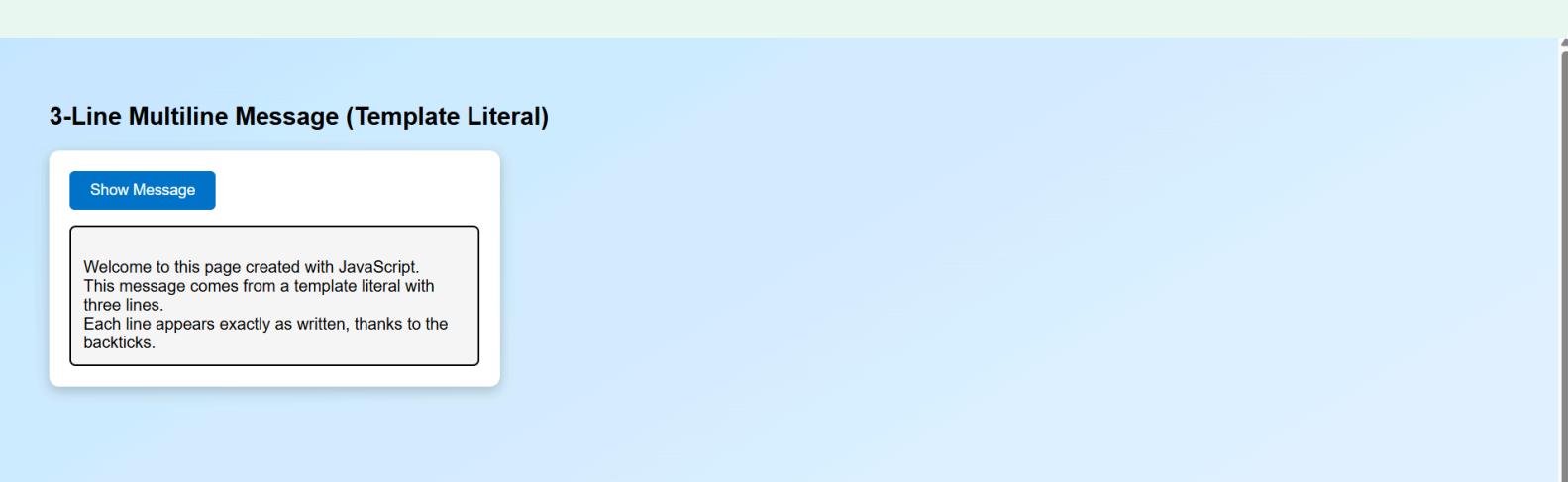
console.log(message);
```

# 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.

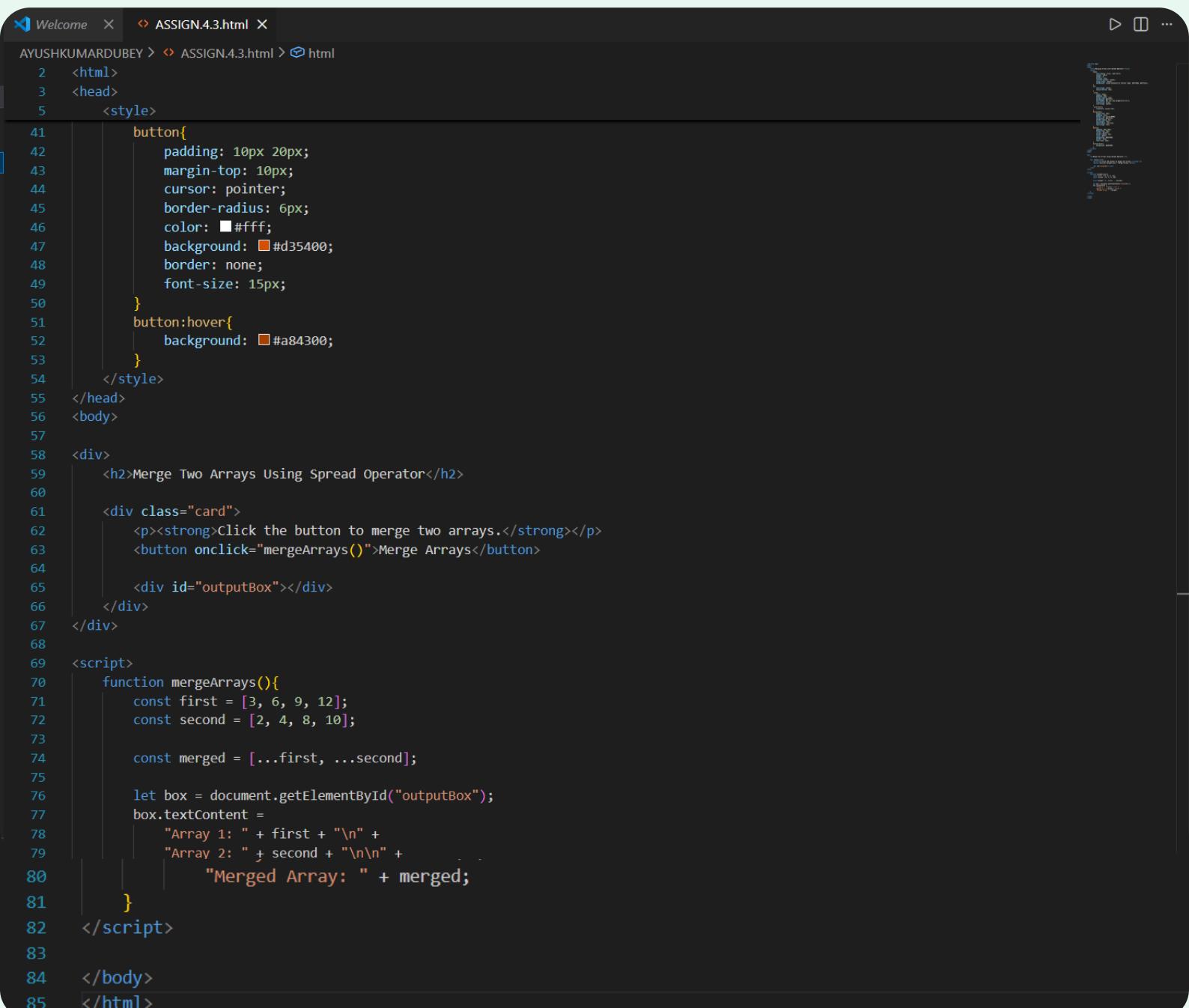
```
AYUSHKUMARDUBEY > ASSGN.4.3.html > html
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Merging Arrays with Spread Operator</title>
5  <style>
6      body{
7          font-family: Arial, sans-serif;
8          height: 100vh;
9          margin: 0;
10         display: flex;
11         justify-content: center;
12         align-items: center;
13         background: linear-gradient(to bottom right, #ffe8d6, #fafafa);
14     }
15     h2{
16         text-align: center;
17         margin-bottom: 20px;
18     }
19     .card{
20         width: 430px;
21         padding: 20px;
22         border-radius: 12px;
23         background: #ffffff;
24         box-shadow: 0px 4px 14px rgba(0,0,0,0.2);
25         transition: 0.25s;
26         text-align: center;
27     }
28     .card:hover{
29         transform: scale(1.02);
30     }
31     #outputBox{
32         margin-top: 15px;
33         padding: 12px;
34         border: 2px solid #000;
35         background: #f1f1f1;
36         border-radius: 6px;
37         min-height: 70px;
38         white-space: pre-line;
39         text-align: left;
40     }
41     button{
42         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.



The screenshot shows a code editor window with the following details:

- File Path: AYUSHKUMARDUBEY > ASSIGN.4.3.html > html
- Code Content:

```
2   <html>
3     <head>
4       <style>
5         button{
6           padding: 10px 20px;
7           margin-top: 10px;
8           cursor: pointer;
9           border-radius: 6px;
10          color: #fff;
11          background: #d35400;
12          border: none;
13          font-size: 15px;
14        }
15        button:hover{
16           background: #a84300;
17         }
18      </style>
19    </head>
20    <body>
21
22      <div>
23        <h2>Merge Two Arrays Using Spread Operator</h2>
24
25        <div class="card">
26          <p><strong>Click the button to merge two arrays.</strong></p>
27          <button onclick="mergeArrays()">Merge Arrays</button>
28
29          <div id="outputBox"></div>
30        </div>
31      </div>
32
33      <script>
34        function mergeArrays(){
35          const first = [3, 6, 9, 12];
36          const second = [2, 4, 8, 10];
37
38          const merged = [...first, ...second];
39
40          let box = document.getElementById("outputBox");
41          box.textContent =
42            "Array 1: " + first + "\n" +
43            "Array 2: " + second + "\n\n" +
44            "Merged Array: " + merged;
45        }
46      </script>
47
48      </body>
49    </html>
```

## EXAMPLE:

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

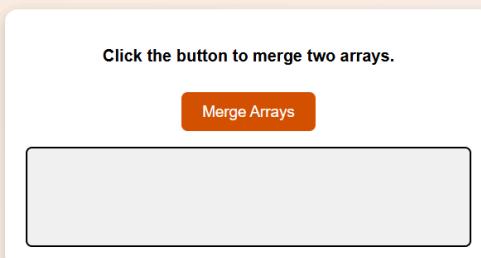
```
const mergedArray=[...arr1, ...arr2];
```

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

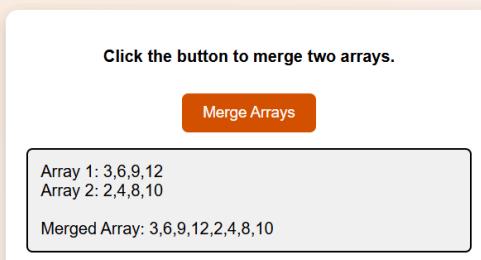
# Q. 4.3 Merge two arrays using the spread operator

output :

## Merge Two Arrays Using Spread Operator



## Merge Two Arrays Using Spread Operator

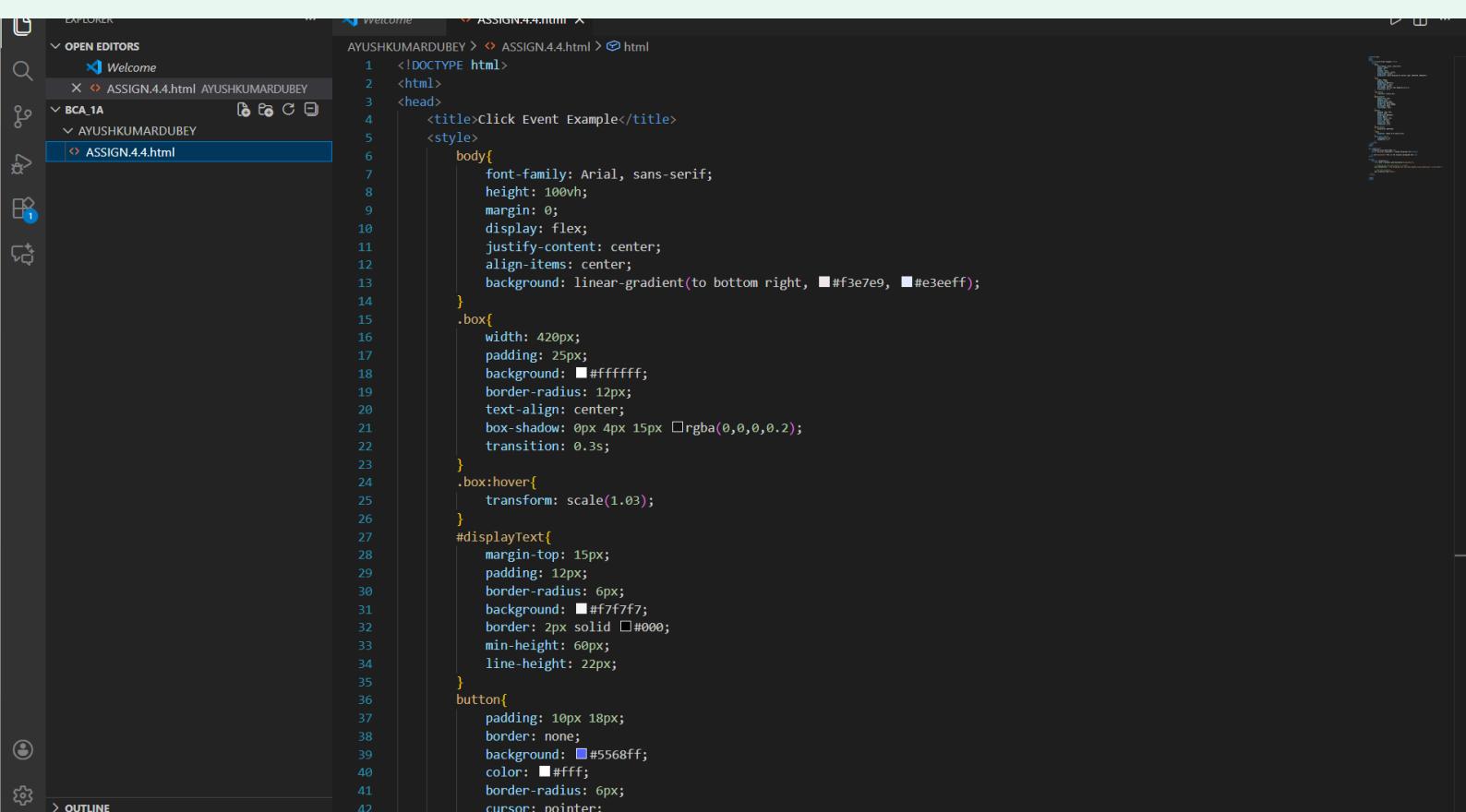


## CONCLUSION:

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.



```
<!DOCTYPE html>
<html>
<head>
<title>Click Event Example</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, #f3e7e9, #e3efff);
}
.box{
    width: 420px;
    padding: 25px;
    background: #ffffff;
    border-radius: 12px;
    text-align: center;
    box-shadow: 0px 4px 15px rgba(0,0,0,0.2);
    transition: 0.3s;
}
.box:hover{
    transform: scale(1.03);
}
#displayText{
    margin-top: 15px;
    padding: 12px;
    border-radius: 6px;
    background: #f7f7f7;
    border: 2px solid #0000;
    min-height: 60px;
    line-height: 22px;
}
button{
    padding: 10px 18px;
    border: none;
    background: #5568ff;
    color: #fff;
    border-radius: 6px;
    cursor: pointer;
}</style>

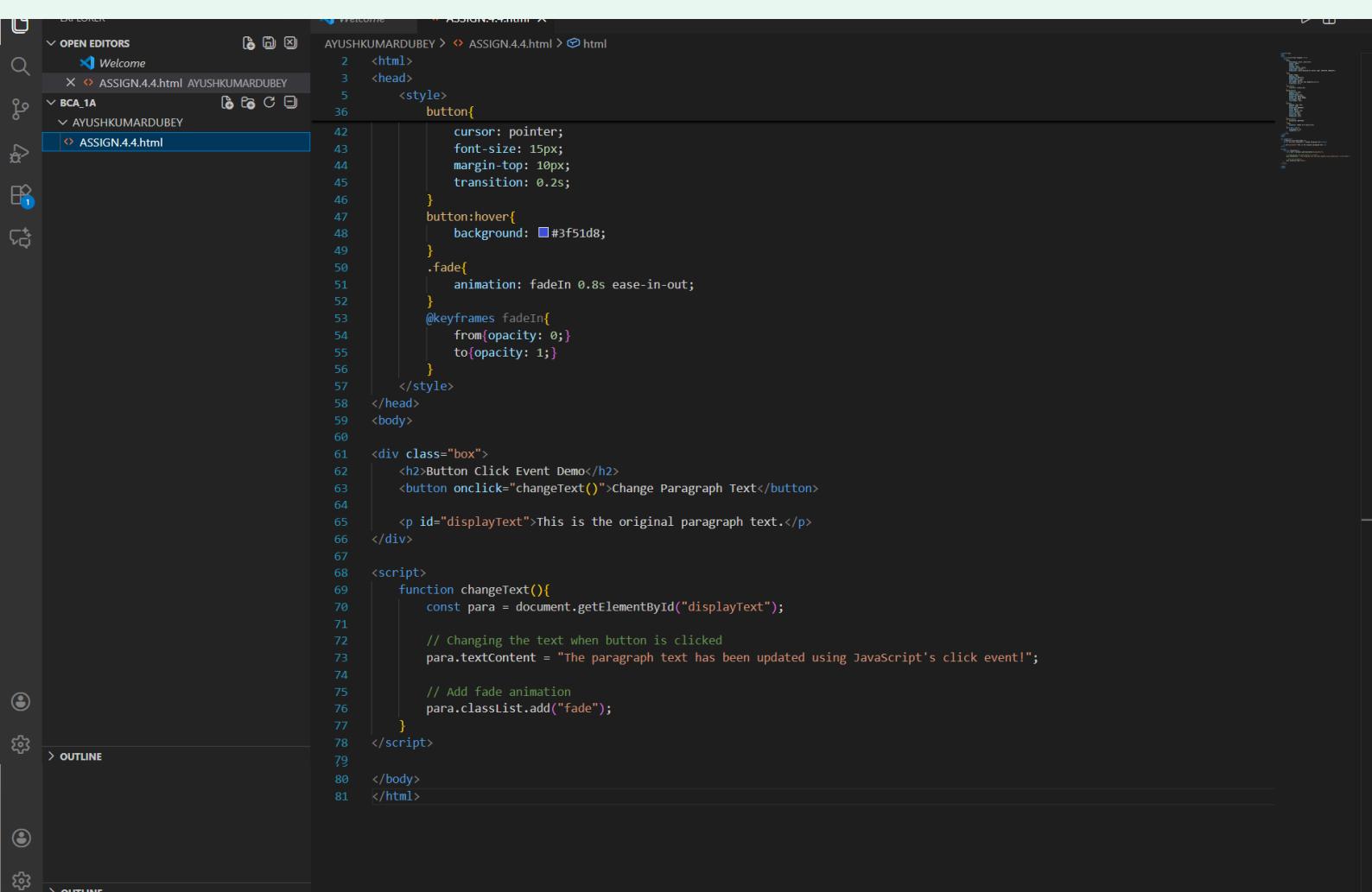
```

## 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.



The screenshot shows a code editor interface with the following details:

- OPEN EDITORS:** Welcome, ASSIGN.4.4.html, BCA\_1A, AYUSHKUMARDUBEY, ASSIGN.4.4.html (selected).
- File Content:**

```
2 <html>
3   <head>
4     <style>
5       button{
6         cursor: pointer;
7         font-size: 15px;
8         margin-top: 10px;
9         transition: 0.2s;
10      }
11      button:hover{
12        background: #3f51d8;
13      }
14      .fade{
15        animation: fadeIn 0.8s ease-in-out;
16      }
17      @keyframes fadeIn{
18        from{opacity: 0;}
19        to{opacity: 1;}
20      }
21    </style>
22  </head>
23  <body>
24
25    <div class="box">
26      <h2>Button Click Event Demo</h2>
27      <button onclick="changeText()">Change Paragraph Text</button>
28
29      <p id="displayText">This is the original paragraph text.</p>
30    </div>
31
32    <script>
33      function changeText(){
34        const para = document.getElementById("displayText");
35
36        // Changing the text when button is clicked
37        para.textContent = "The paragraph text has been updated using JavaScript's click event!";
38
39        // Add fade animation
40        para.classList.add("fade");
41      }
42    </script>
43
44  </body>
45 </html>
```
- OUTLINE:** (empty)

<!-- HTML -->

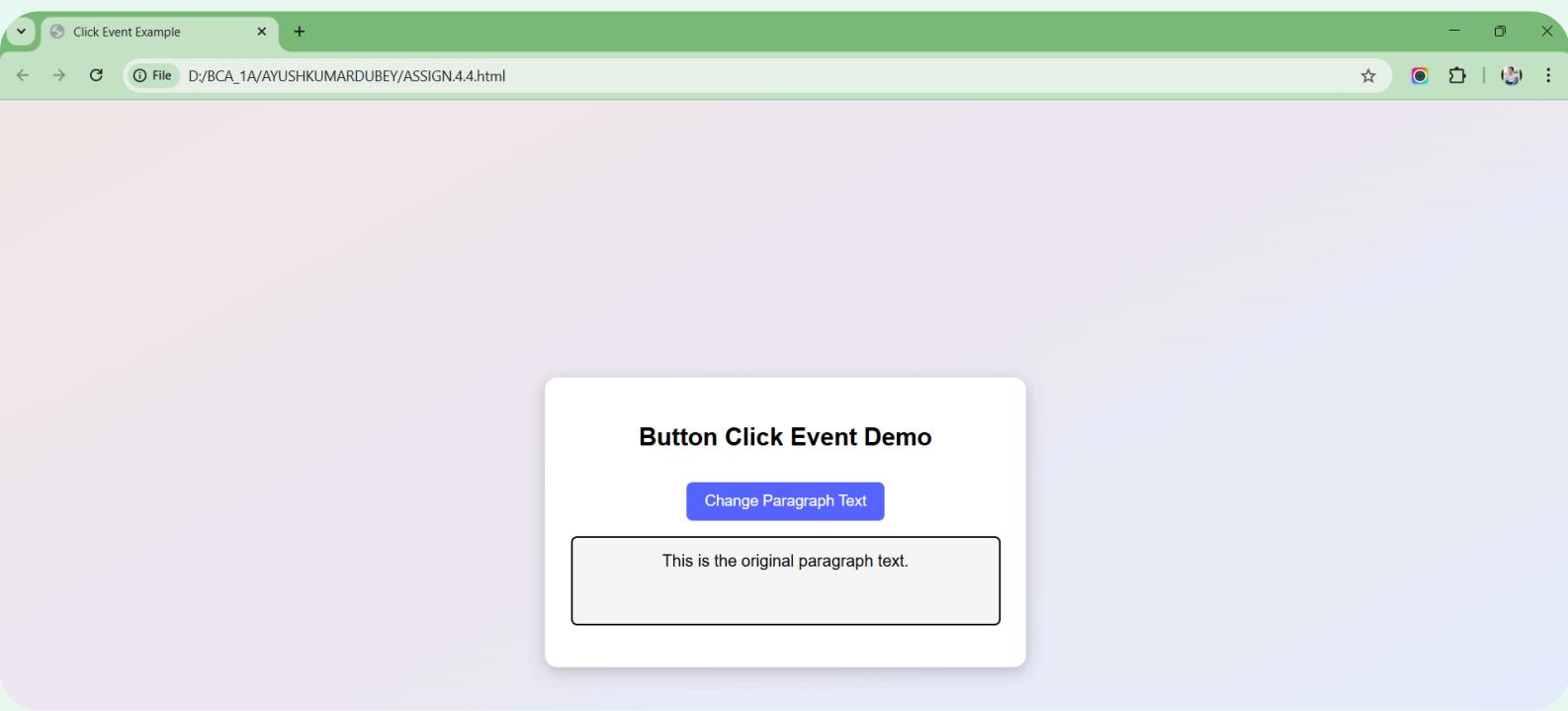
<p id="msg">Original text</p>

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

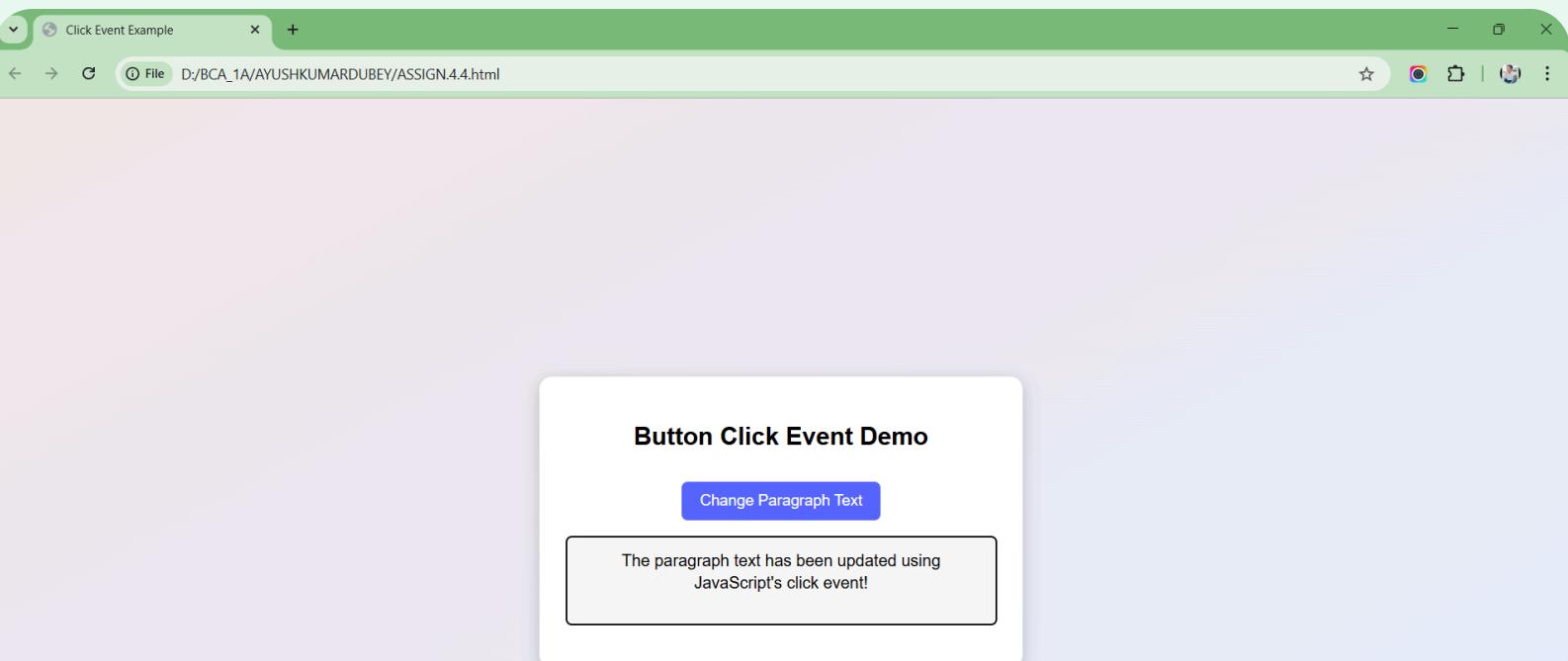
```
document.getElementById("changeBtn").addEventListener("click", function() {
  document.getElementById("msg").textContent = "The text has been changed!";
});
```

# 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.