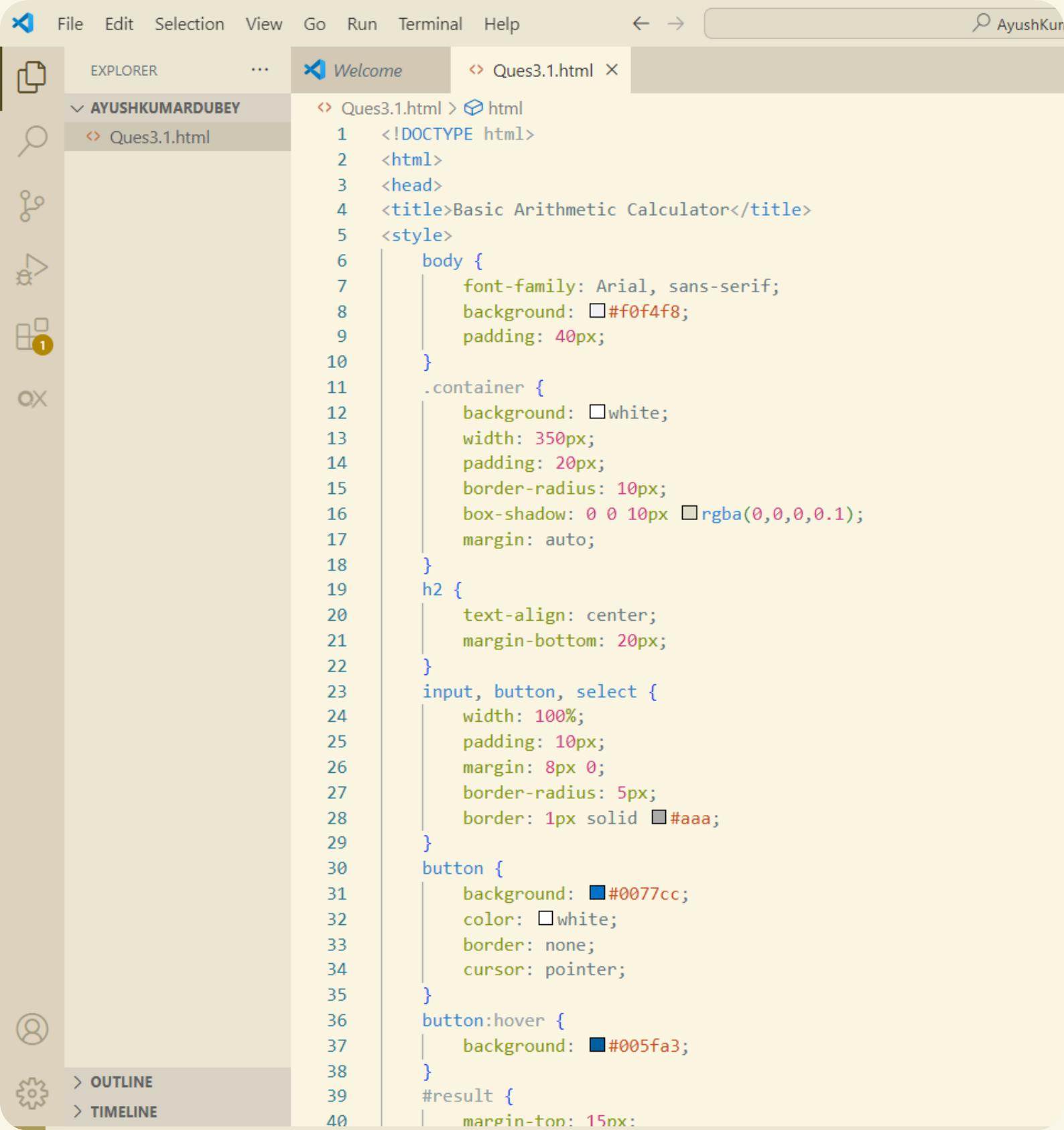


3.1 Create a JavaScript-enabled web page that performs basic arithmetic operations using input from users

Introduction

A JavaScript-based arithmetic page allows users to enter two numbers and choose an operation such as addition, subtraction, multiplication, or division. The browser performs calculations instantly without the need for external processing. This makes the page interactive and helps students understand how JavaScript interacts with HTML elements.



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

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- User Profile:** AyushKumardubey.
- Explorer:** Shows a folder named AYUSHKUMARDUBEY containing a file named Ques3.1.html.
- Code Editor:** Displays the content of Ques3.1.html, which includes an HTML structure and a corresponding CSS style sheet.

```
<!DOCTYPE html>
<html>
<head>
<title>Basic Arithmetic Calculator</title>
<style>
body {
    font-family: Arial, sans-serif;
    background: #f0f4f8;
    padding: 40px;
}
.container {
    background: white;
    width: 350px;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
    margin: auto;
}
h2 {
    text-align: center;
    margin-bottom: 20px;
}
input, button, select {
    width: 100%;
    padding: 10px;
    margin: 8px 0;
    border-radius: 5px;
    border: 1px solid #aaa;
}
button {
    background: #0077cc;
    color: white;
    border: none;
    cursor: pointer;
}
button:hover {
    background: #005fa3;
}
#result {
    margin-top: 15px;
}
</style>
<body>
<div class="container">
<h2>Basic Arithmetic Calculator</h2>
<form>
<input type="text" placeholder="First Number" />
<input type="text" placeholder="Second Number" />
<select>
<option value="add">AddSubtractMultiplyDivideCalculate</button>
<div id="result"></div>
</form>
</div>
</body>
```

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

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Includes icons for Explorer, Search, Share, and others.
- Sidebar:** Shows a tree view under "AYUSHKUMARDUBEY" with "Ques3.1.html" selected.
- Code Area:** Displays the HTML code for an arithmetic calculator.

```
2 <html>
3 <head>
5 <style>
39 #result {
40     margin-top: 20px;
41     font-weight: bold;
42     color: black;
43     text-align: center;
44 }
45 </style>
46 </head>
47
48 <body>
49
50 <div class="container">
51     <h2>Arithmetic Calculator</h2>
52
53     <label>Enter First Number:</label>
54     <input type="number" id="num1" placeholder="Number 1">
55
56     <label>Enter Second Number:</label>
57     <input type="number" id="num2" placeholder="Number 2">
58
59     <label>Select Operation:</label>
60     <select id="operation">
61         <option value="add">Addition (+)</option>
62         <option value="sub">Subtraction (-)</option>
63         <option value="mul">Multiplication (×)</option>
64         <option value="div">Division (÷)</option>
65     </select>
66
67     <button onclick="calculate()">Calculate</button>
68
69     <div id="result"></div>
70 </div>
71
72 <script>
73     function calculate() {
74         let a = document.getElementById("num1").value;
75         let b = document.getElementById("num2").value;
```
- Bottom Bar:** Shows "OUTLINE" and "TIMELINE" buttons.

Input Explanation → The webpage accepts:

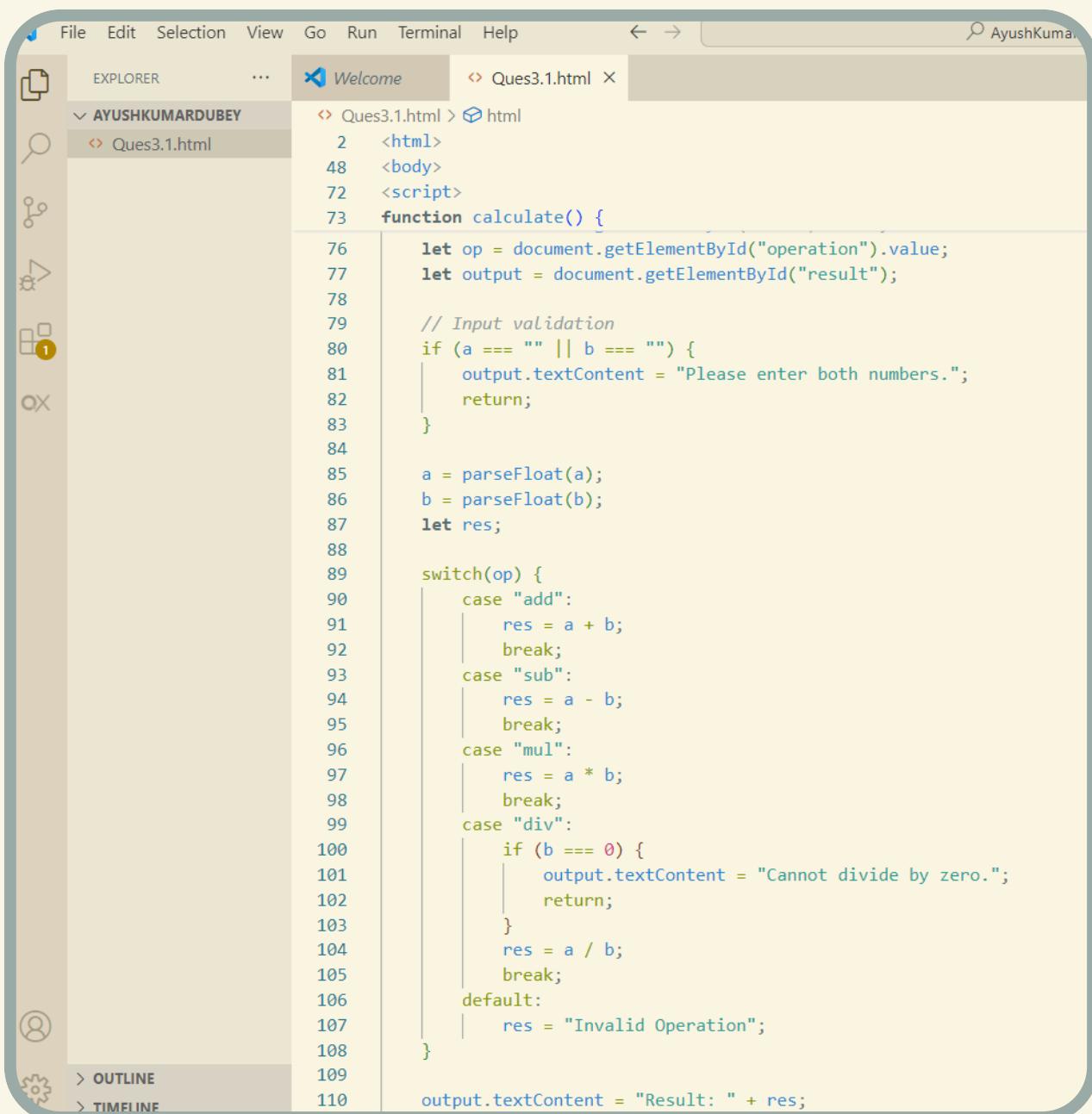
Input Purpose

Number 1: First value to be calculated

Number 2: Second value to be calculated

Operation Button: Chooses the type of arithmetic

JAVASCRIPT HERE



File Edit Selection View Go Run Terminal Help

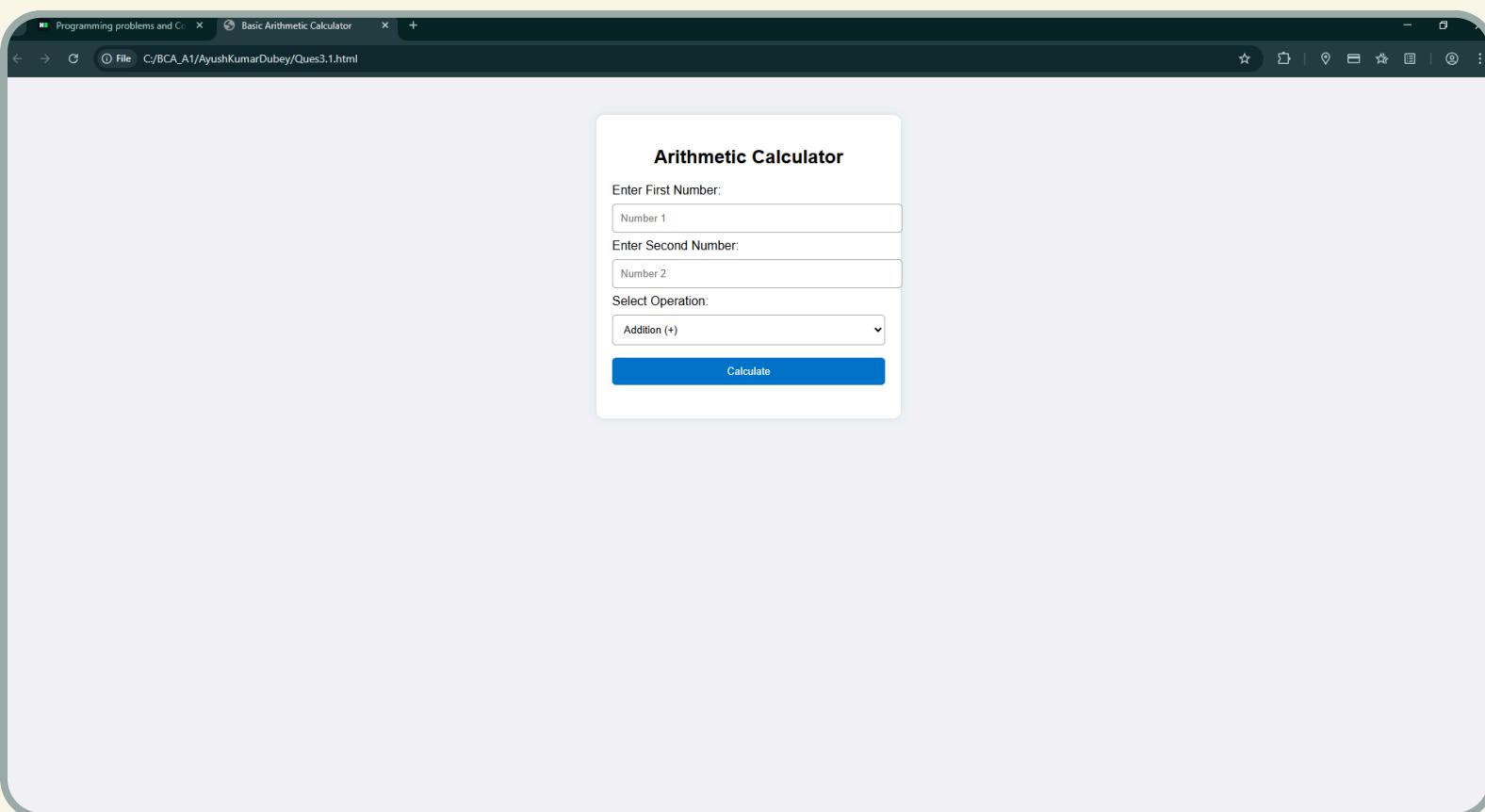
EXPLORER ... Welcome Ques3.1.html

AYUSHKUMARDUBEY Ques3.1.html

```
2   <html>
48  <body>
72  <script>
73  function calculate() {
76      let op = document.getElementById("operation").value;
77      let output = document.getElementById("result");
78
79      // Input validation
80      if (a === "" || b === "") {
81          output.textContent = "Please enter both numbers.";
82          return;
83      }
84
85      a = parseFloat(a);
86      b = parseFloat(b);
87      let res;
88
89      switch(op) {
90          case "add":
91              res = a + b;
92              break;
93          case "sub":
94              res = a - b;
95              break;
96          case "mul":
97              res = a * b;
98              break;
99          case "div":
100             if (b === 0) {
101                 output.textContent = "Cannot divide by zero.";
102                 return;
103             }
104             res = a / b;
105             break;
106         default:
107             res = "Invalid Operation";
108     }
109
110    output.textContent = "Result: " + res;
111 }
112 </script>
113 </body>
114 </html>
```

```
107     |     |     res = "Invalid Operation";
108     |
109     |     output.textContent = "Result: " + res;
110     |
111 }
```

OUTPUT



A screenshot of a mobile device showing the same arithmetic calculator application. The screen displays the "Arithmetic Calculator" title, input fields for "Enter First Number" (200) and "Enter Second Number" (2), a dropdown for "Select Operation" (Division (/)), and a blue "Calculate" button. Below the button, the result "Result: 100" is shown. Three orange circles highlight the "Calculate" button, the division operation selection, and the second number input field.

| To Increase or
Decrease Value

| To Switch operation
Ex:- (-) to (+)

A screenshot of a mobile device showing the arithmetic calculator application again. The screen shows the "Arithmetic Calculator" title, input fields for "Enter First Number" (200) and "Enter Second Number" (2), and a dropdown for "Select Operation" (Division (/)). The dropdown menu is open, showing options: Division (/), Addition (+), Subtraction (-), Multiplication (×), and Division (/). The "Division (/)" option is highlighted with a blue background. Three orange circles highlight the "Division (/)" operation in the dropdown, the second number input field, and the "Calculate" button.

| Types Of operations
which are available
use as accordingly ;

OUTPUT

Arithmetic Calculator

Enter First Number:

Enter Second Number:

Select Operation:

Calculate

Result: 100

Arithmetic Calculator

Enter First Number:

Enter Second Number:

Select Operation:

Calculate

Result: 6

It showcasing division

Here Example of Addition

Arithmetic Calculator

Enter First Number:

Enter Second Number:

Select Operation:

Calculate

Result: 12

Arithmetic Calculator

Enter First Number:

Enter Second Number:

Select Operation:

Calculate

Result: 2

Here Ex. For Multiplication

Subtracting two numbers

OUTPUT

Data Types Used

Data Type

number

string.

HTMLElement.

Role

Stores numeric user inputs

Used for identifying the selected operation

Interacts with input/output elements

Conclusion

This example demonstrates how to combine HTML and JavaScript to build a simple interactive calculator. It teaches key concepts such as DOM access, input handling, event-driven functions, and basic arithmetic logic. Students can extend the project by adding CSS, dropdown menus, or automated validation.

Q3.2 Write JavaScript to demonstrate conditional statements and looping (for, while, do...while).

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

- EXPLORER** sidebar on the left lists files: Ques3.1.html, Ques3.2.html, and temp.html.
- File Explorer** at the top shows the current file is "Ques3.2.html".
- Code Editor Content:**

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Conditional & Looping example</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background: #e9eff5;
            padding: 20px;
        }
        .container {
            width: 400px;
            background: #fff;
            margin: auto;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0px 0px 10px rgba(0,0,0,0.2);
        }
        h2 {
            text-align: center;
            color: #333;
        }
        .section-title {
            font-weight: bold;
            margin-top: 15px;
        }
        p {
            margin: 5px 0;
        }
        hr {
            margin: 12px 0;
        }
    </style>
</head>
<body>
<div class="container">
    <h2>JS Conditional & Looping Demo</h2>
    <script>
        // ----- Conditional Statement -----
        let number = 10;
        document.write("<p class='section-title'>Conditional Statement:</p>");

        if (number > 0) {
            document.write("<p>The number is positive.</p>");
        }
    </script>
</div>
</body>
```
- Bottom Navigation:** OUTLINE and TIMELINE buttons.

Conditional statements help a program make decisions based on conditions. Loops allow repeated execution of code until a stopping point is reached. Together, they enable logical flow and controlled repetition in JavaScript.

JAVASCRIPT HERE

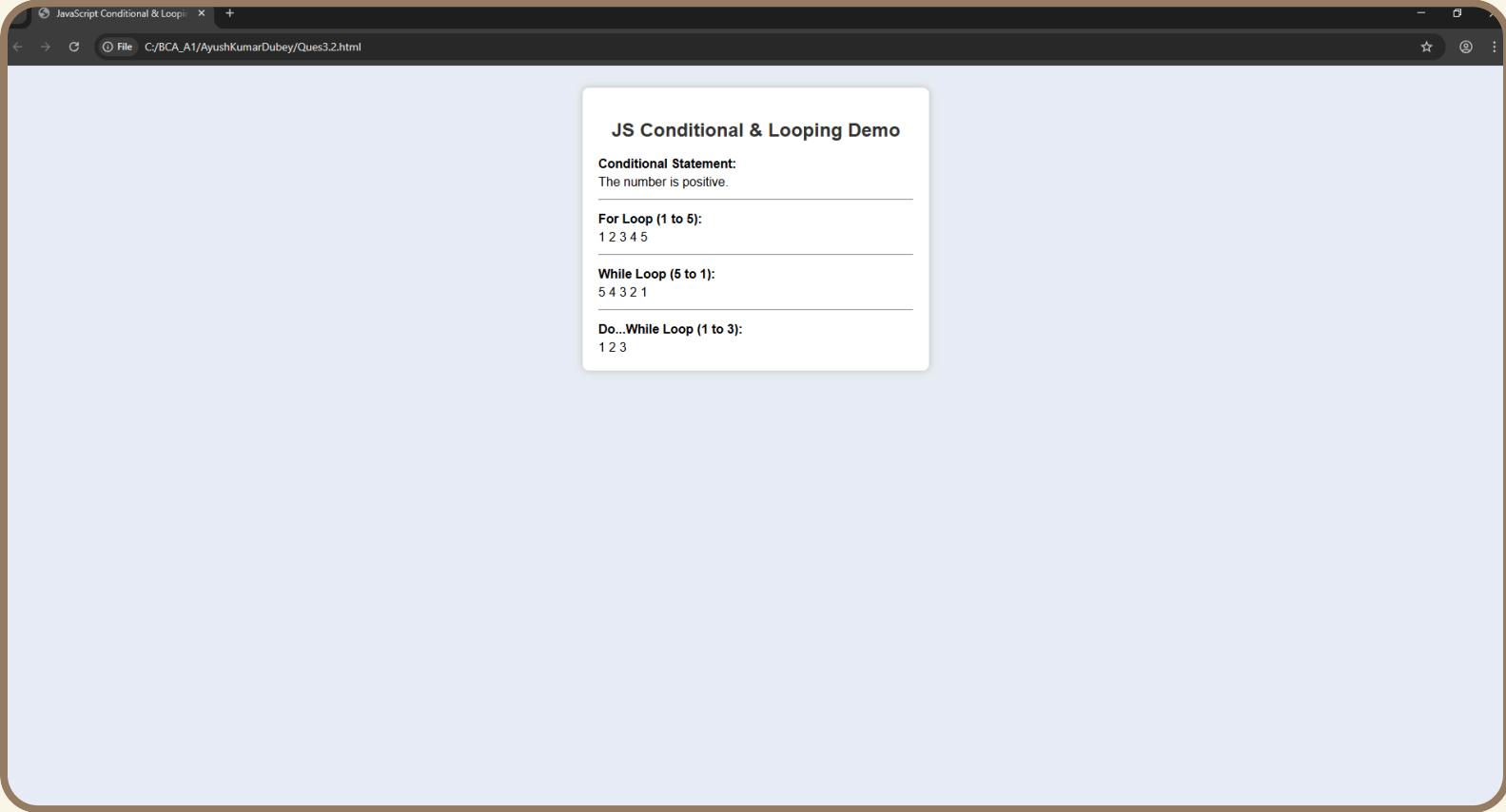
Ques3.1.html
Ques3.2.html
temp.html

```
2  <html>
36
37  <body>
38
39  <div class="container">
40      <h2>JS Conditional & Looping Demo</h2>
41
42      <script>
43          // ----- Conditional Statement -----
44          let number = 10;
45          document.write("<p class='section-title'>Conditional Statement:</p>");
46
47          if (number > 0) {
48              document.write("<p>The number is positive.</p>");
49          }
50          else if (number < 0) {
51              document.write("<p>The number is negative.</p>");
52          }
53          else {
54              document.write("<p>The number is zero.</p>");
55          }
56
57          document.write("<hr>");
58
59          // ----- For Loop -----
60          document.write("<p class='section-title'>For Loop (1 to 5):</p>");
61          for (let i = 1; i <= 5; i++) {
62              document.write(i + " ");
63          }
64
65          document.write("<hr>");
66          document.write("<p class='section-title'>While Loop (5 to 1):</p>");
67          let x = 5;
68          while (x >= 1) {
69              document.write(x + " ");
70              x--;
71          }
72          document.write("<hr>");
73          document.write("<p class='section-title'>Do...While Loop (1 to 3):</p>");
74          let y = 1;
75          do {
76              document.write(y + " ");
77              y++;
78          } while (y <= 3);
79      </script>
80  </div>
81  </body>
82  </html>
```

How it works

The program checks each condition. The first true condition produces the output. Only one block executes.

OUTPUT



A. For Loop

```
For (let i = 1; i <= 5; i++) {  
    console.log("For loop count: " + i);  
}
```

Explanation

Starts at 1, stops at 5

Increments by 1

Prints 5 lines

B. While Loop

```
let count = 1;  
while (count <= 5) {  
    console.log("While loop count: " + count);  
    count++;  
}
```

Explanation

Checks the condition before each repeated action

Stops when the condition becomes false

C. Do...While Loop

```
let num = 1;  
do {  
    console.log("Do...While loop count: " + num);  
    num++;  
} while (num <= 5);
```

Explanation

Executes the block at least once, even if the condition is false initially

OUTPUT

| 4. Summary of Data Types Used

Data Type	Used For
number	Loop counters, score values
string	Console messages
boolean	Internal conditional checking outcome

Conclusion

JavaScript's conditional statements allow programs to decide which instructions to execute based on specific criteria. Looping structures—for, while, and do...while—enable repetition of tasks efficiently. These tools are essential for handling logic, automation, and dynamic behavior in programming.



3.3 Create and invoke user-defined functions in JavaScript use var, let, and const for scope demonstration

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

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Includes icons for Save, Undo, Redo, Cut, Copy, Paste, Find, and Select All.
- Explorer:** Shows a tree view of files under "AYUSHK...". The file "Ques3.3.html" is selected and highlighted in yellow.
- Code Editor:** The main area displays the content of "Ques3.3.html".

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Functions & Scope Demo</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background: #eef3f8;
            padding: 20px;
        }
        .box {
            width: 420px;
            margin: auto;
            background: white;
            padding: 18px;
            border-radius: 8px;
            box-shadow: 0px 0px 8px rgba(0,0,0,0.2);
        }
        h2 {
            text-align: center;
        }
        p {
            margin: 8px 0;
        }
        hr {
            margin: 15px 0;
        }
    </style>
</head>
<body>
    <div class="box">
        <h2>JS Functions & Scope Demonstration</h2>
        <script>
            // -----
            // User-defined Functions
            // -----
            function greet(name) {
                return "Hello, " + name + "!";
            }

            function addNumbers(a, b) {
                return a + b;
            }
        </script>
    </div>
</body>

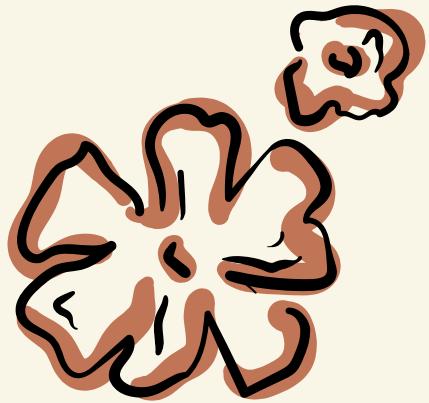
```
- Sidebar:** Includes "OUTLINE" and "TIMELINE" buttons.

User-defined functions allow programmers to bundle instructions into reusable blocks. JavaScript also provides different ways to declare variables—var, let, and const—each with its own scope rules. Understanding these helps prevent errors and improves code organization.

Creating and Invoking User-Defined Functions

A. Simple Function Example

```
function greet(name){  
    return "Hello, " + name + "!";  
}  
console.log(greet("James")); // Function call
```



Explanation

- * Defined using the function keyword

- * Accepts a parameter

- * Returns a value

- * Invoked using its name and passing an argument

```
49  
50     function showSquare(n) {  
51         return n * n;  
52     }  
53  
54     document.write("<p><b>Function Outputs:</b></p>");  
55     document.write("<p>" + greet("Student") + "</p>");  
56     document.write("<p>Sum of 5 and 3: " + addNumbers(5, 3) + "</p>");  
57     document.write("<p>Square of 6: " + showSquare(6) + "</p>");  
58  
59     document.write("<hr>");  
60  
61     // -----  
62     // var, let, const example  
63     // -----  
64  
65     document.write("<p><b>Scope Demonstration (var, let, const):</b></p>");  
66  
67     // var example (function-scoped)  
68     var x = 10;  
69     document.write("<p>var x = " + x + " (function scoped)</p>");  
70  
71     // let example (block-scoped)  
72     {  
73         let y = 20;  
74         document.write("<p>let y = " + y + " (block scoped)</p>");  
75     }  
76     // y is not accessible here (scope ends)  
77  
78     // const example (cannot be reassigned)  
79     const pi = 3.14;  
80     document.write("<p>const pi = " + pi + " (constant value)</p>");  
81  
82     // Trying to reassign const pi would cause an error  
83     </script>  
84  
85     </div>  
86  
87     </body>  
88     </html>
```

OUTPUT

The screenshot shows a browser window with the title "JavaScript Functions & Scope D". The address bar indicates the file path: "C:/BCA_A1/AyushKumarDubey/Ques3.3.html". The main content area displays a box titled "JS Functions & Scope Demonstration". Inside, under "Function Outputs:", it lists: "Hello, Student!", "Sum of 5 and 3: 8", and "Square of 6: 36". Below this, under "Scope Demonstration (var, let, const):", it shows: "var x = 10 (function scoped)", "let y = 20 (block scoped)", and "const pi = 3.14 (constant value)".

Using var, let, and const for Scope Demonstration

Keyword	Scope	Can Reassign?	Can Redeclare
var	Function	Yes	No
let	Block	Yes	No
const	Block	No	No

Conclusion

User-defined functions are essential for organizing reusable logic in JavaScript. Variable declarations—var, let, and const—play a crucial role in controlling scope and preventing unintended value changes. Understanding function creation, function invocation, and scope helps developers write clean, predictable, and professional-grade JavaScript code.

Q. 3.4 Handle HTML form validation using JavaScript events like onsubmit, on input, and use alert() for feedback

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

- Menu Bar:** Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Back, Forward, Search bar, and a button labeled "Ayu".
- Left Sidebar (Explorer):** Shows a tree view with files: Ques3.1.html, Ques3.2.html, Ques3.3.html, Ques3.4.html (selected), and temp.html.
- Right Sidebar (Outline):** Shows sections: OUTLINE and TIMELINE.
- Central Area:** A code editor window titled "Welcome" showing the content of "Ques3.4.html".

```
<!DOCTYPE html>
<html>
<head>
    <title>Form Validation with Events</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background: #eef3f8;
            padding: 20px;
        }
        .form-box {
            width: 350px;
            background: white;
            margin: auto;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0px 0px 8px rgba(0,0,0,0.2);
        }
        h2 {
            text-align: center;
        }
        input {
            width: 100%;
            padding: 10px;
            margin: 8px 0;
            border-radius: 5px;
            border: 1px solid #aaa;
        }
        button {
            width: 100%;
            padding: 10px;
            background: #0077cc;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }
        button:hover {
            background: #005fa3;
        }
        small {
            color: red;
            display: none;
        }
    </style>
</head>
<body>
```

Form validation ensures that users enter correct and complete information before the form is submitted. JavaScript provides several events to manage validation, including:



oninput → Runs as the user types



onsubmit → Runs when the user tries to submit the form



alert() → Displays messages and warnings

These events help prevent invalid data and give real-time feedback.

```
48
49 <body>
50
51 <div class="form-box">
52   <h2>Form Validation Demo</h2>
53
54   <form id="myForm" onsubmit="return validateForm()">
55
56     <label>Name:</label>
57     <input type="text" id="name" oninput="checkName()">
58     <small id="nameError">Name must be at least 3 characters</small>
59
60     <label>Email:</label>
61     <input type="text" id="email" oninput="checkEmail()">
62     <small id="emailError">Enter a valid email</small>
63
64     <label>Age:</label>
65     <input type="number" id="age" oninput="checkAge()">
66     <small id="ageError">Age must be 18 or above</small>
67
68     <button type="submit">Submit</button>
69   </form>
70 </div>
71
72 <script>
73   // ----- oninput validation -----
74
75   function checkName() {
76     let name = document.getElementById("name").value;
77     let err = document.getElementById("nameError");
78
79     if (name.length < 3) {
80       err.style.display = "block";
81       return false;
82     } else {
83       err.style.display = "none";
84       return true;
85     }
86   }
87
88   function checkEmail() {
89     let email = document.getElementById("email").value;
90     let err = document.getElementById("emailError");
91
92     if (!email.includes("@") || !email.includes(".")) {
93       err.style.display = "block";
94     }
95   }
96
97   function validateForm() {
98     let nameErr = document.getElementById("nameError");
99     let emailErr = document.getElementById("emailError");
100
101    if (checkName() && checkEmail()) {
102      alert("Form submitted successfully!");
103      return true;
104    } else {
105      return false;
106    }
107  }
108
109 </script>
```

3. Explanation of Events Used

A. oninput Event

Runs every time the user types.
Used for live validation.

Example:

```
oninput="checkName()"
```

B. onsubmit Event

Executes when the form is submitted.

Used for final form checking.

Example:

```
<form onsubmit="return validateForm()">
```

C. alert() Function

Shows pop-up messages.

Used to notify users about errors or success.

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

- Top Bar:** Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:** Ayush...
- Explorer Bar:** Shows files: AYUSHK... (selected), Ques3.1.html, Ques3.2.html, Ques3.3.html, Ques3.4.html, temp.html.
- Code Editor:** Displays an HTML file with embedded JavaScript. The code includes three validation functions: checkName(), checkEmail(), and checkAge(). It also contains an onsubmit event handler validateForm() which checks all three and shows an alert if successful.

```
<html>
<body>
<script>
    function checkName() {
        let name = document.getElementById("name").value;
        let err = document.getElementById("nameError");

        if (!name.includes(" ")) {
            err.style.display = "block";
            return false;
        } else {
            err.style.display = "none";
            return true;
        }
    }

    function checkEmail() {
        let email = document.getElementById("email").value;
        let err = document.getElementById("emailError");

        if (!email.includes("@") || !email.includes(".")) {
            err.style.display = "block";
            return false;
        } else {
            err.style.display = "none";
            return true;
        }
    }

    function checkAge() {
        let age = document.getElementById("age").value;
        let err = document.getElementById("ageError");

        if (age < 18) {
            err.style.display = "block";
            return false;
        } else {
            err.style.display = "none";
            return true;
        }
    }

    // ----- onsubmit validation -----

    function validateForm() {
        let validName = checkName();
        let validEmail = checkEmail();
        let validAge = checkAge();

        if (validName && validEmail && validAge) {
            alert("Form submitted successfully!");
            return true;
        } else {
            alert("Please correct the errors before submitting.");
            return false;
        }
    }
</script>
</body>
</html>
```

- Bottom Bar:** OUTLINE, TIMELINE.

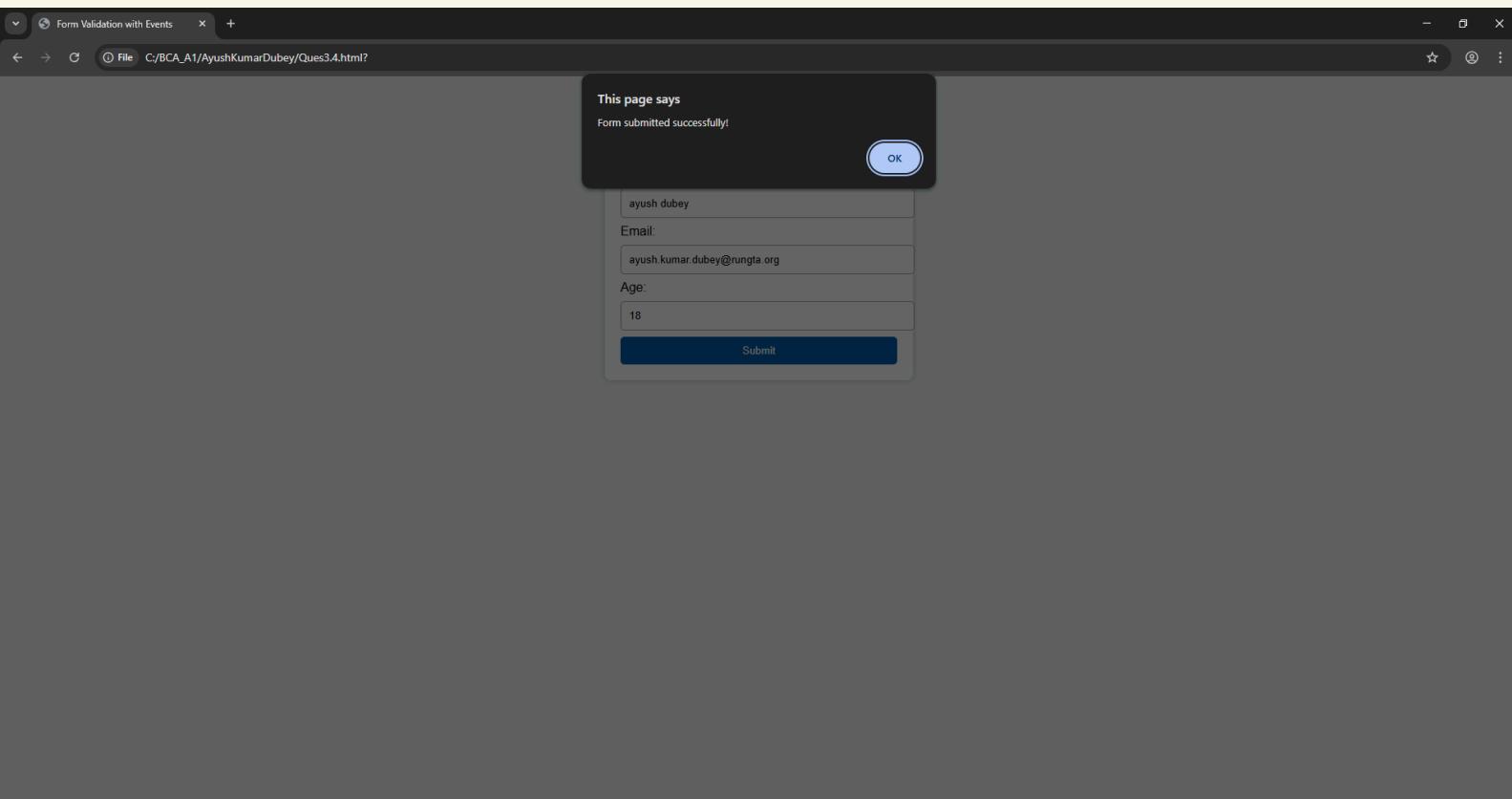
OUTPUT IMAGE

A screenshot of a web browser window titled "Form Validation with Events". The address bar shows the URL "C:/BCA_A1/AyushKumarDubey/Ques3.4.html?". The main content is a form titled "Form Validation Demo" with three input fields: "Name", "Email", and "Age", followed by a blue "Submit" button.

filled With User Data

A screenshot of a web browser window titled "Form Validation with Events". The address bar shows the URL "C:/BCA_A1/AyushKumarDubey/Ques3.4.html?". The main content is a form titled "Form Validation Demo" with three input fields: "Name" containing "ayush dubey", "Email" containing "ayush.kumar.dubey@rungta.org", and "Age" containing "18", followed by a blue "Submit" button.

FORM SUCCESSFULLY SUBMITTED EXAMPLE



Example

- * Name must have 3 or more characters
- * Email must match a proper format
- * All fields are required
- * Alerts appear when something is wrong



Conclusion

JavaScript events such as `oninput` and `onsubmit` provide powerful tools for building interactive form validation. Real-time feedback improves user experience, while final submit validation ensures accurate and clean data. Using `alert()` gives clear feedback before submitting the form.