**Module 2 Assignment : Exploring JavaScript Topics with EJS, Node.js, and Express**

Saurabh Kale

IFT 458/554: Middleware Programming & Database Security

Dinesh Sthapit

Sep 3rd, 2023

* Screenshot of Project init and packages installed-  
  A screen shot of a computer

  Description automatically generated
* Screenshot of code running-  
  A screenshot of a computer

  Description automatically generated
* Screenshot of ‘/calculate’ route-  
  A screenshot of a computer

  Description automatically generated
* Screenshot of app.js code

A screen shot of a computer

Description automatically generated

* Screenshot of app.js part 2-  
  A computer screen shot of a program code

  Description automatically generated
* Code-

// Student Name: Saurabh

// Student ID: 1223450319

// Date: 09/02/2023

// Import the required modules

const express = require('express');

const bodyParser = require('body-parser');

const path = require('path');

// Create an instance of express

const app = express();

// We use the 'body-parser' middleware to parse the incoming request bodies

app.use(bodyParser.urlencoded({ extended: false }));

// Set the view engine to ejs

app.set('view engine', 'ejs');

app.set('views', path.join(\_\_dirname, 'views'));

console.log('views', path.join(\_\_dirname, 'views'));

// The GET route for the form

app.get('/', (req, res) => {

// Render the form and pass in the current student data

res.render('index');

});

// create a route for user to enter the numbers

app.post('/calculate', (req, res) => {

const { num1, num2 } = req.body;

const sum = Number(num1) + Number(num2);

const difference = Number(num1) - Number(num2);

const product = Number(num1) \* Number(num2);

const quotient = Number(num1) / Number(num2);

res.render("result", { sum,difference, product, quotient});

});

// Start the server on port 4000

// Note we are advertising the service on port number 4000 and not 3000 this time

var port = 4000

// NOTE

// this quotes are replaced by back ticks ` next to key caps 1

app.listen(port, () => {

console.log('views', path.join(\_\_dirname, 'views'));

console.log(`Server is running on port ${port}`);

});

* Screenshot of index.ejs-  
  A screen shot of a computer

  Description automatically generated

Code-  
<!-- // Student Name: Saurabh

// Student ID: 1223450319

// Date: 09/02/2023 -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<form action="/calculate" method="POST">

<input type="number" name="num1" placeholder="Enter first number" required>

<input type="number" name="num2" placeholder="Enter second number" required>

<button type="submit">Calculate</button>

</form>

</body>

</html>

* Screenshot of result.ejs-  
  A screen shot of a computer

  Description automatically generated

Code-  
<!-- // Student Name: Saurabh

// Student ID: 1223450319

// Date: 09/02/2023 -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<form action="/calculate" method="POST">

<input type="number" name="num1" placeholder="Enter first number" required>

<input type="number" name="num2" placeholder="Enter second number" required>

<button type="submit">Calculate</button>

</form>

<p>

<h1>Calculation Results:</h1>

<p>Sum: <%= sum %> </p>

<p>Difference: <%= difference %></p>

<p>Product: <%= product %></p>

<p>Quotient: <%= quotient %></p>

</p>

</body>

</html>

* **ejs Code Analysis-**

.ejs Code Analysis:

JavaScript Code Embedding:

In the .ejs file, JavaScript code is embedded within <% and %> tags. For example, <%= sum %> is surrounded by these tags, indicating that it's JavaScript code that will be evaluated and its result will be inserted into the HTML.

Control Structures:

There are no control structures like loops or conditionals visible in the given .ejs file. It seems to primarily focus on rendering calculated results based on user input.

Variables:

Variables like sum, difference, product, and quotient are accessed and displayed within the HTML content using the <%= variableName %> syntax. These variables are expected to hold dynamic values calculated based on user input.

* **Html code analysis-**

Dynamic Content:

The HTML code contains placeholders where dynamic content from the .ejs file will be inserted. For example, <%= sum %> will be replaced with the calculated sum.

Form Submission:

There is a form element that takes user input for two numbers and submits the data to "/calculate" using the POST method. This form is used to send user input to the server for calculation.

Calculation Results:

Under the "Calculation Results" heading, you have paragraphs that display the results of calculations. These results are inserted dynamically using the <%= variableName %> syntax. In the HTML, they appear as placeholders, but the actual values will be filled in when the .ejs file is rendered.

Page Structure:

The HTML code follows the standard structure with a <head> section for metadata and a <body> section for the page content.

* Modify the .ejs file to include more complex JavaScript logic, such as conditional rendering or iteration over an array

A screenshot of a computer

Description automatically generated

Code-  
result.ejs-  
<!-- // Student Name: Saurabh

// Student ID: 1223450319

// Date: 09/02/2023 -->

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <form action="/calculate" method="POST">

        <input type="number" name="num1" placeholder="Enter first number" required>

        <input type="number" name="num2" placeholder="Enter second number" required>

        <button type="submit">Calculate</button>

    </form>

    <h1>Calculation Results:</h1>

    <p>Sum: <%= sum %> </p>

    <p>Difference: <%= difference %></p>

    <p>Product: <%= product %></p>

    <p>Quotient: <%= quotient %></p>

    <!-- Javascript logic for conditional operation -->

    <% if (sum%2==0) { %>

        <p >The sum <%= sum %> is even</p>

    <% } else { %>

        <p>The sum <%= sum %> is odd</p>

    <% } %>

    <!-- Javascript logic for iteration -->

    <h2>List of Numbers:</h2>

    <ul>

        <% for (let i = 1; i <= 10; i++) { %>

            <% if (i % 2 === 0) { %>

                <li style="color: blue;"><%= i %></li>

            <% } else { %>

                <li><%= i %></li>

            <% } %>

        <% } %>

    </ul>

</body>

</html>

**Analysis-**

* Conditional Operation:

In the provided .ejs code, there's a conditional operation that checks if the variable sum is even or odd. If sum is even, it displays a message indicating that the sum is even; otherwise, it displays a message indicating that the sum is odd.

* Iteration Over Numbers:

The .ejs code includes a loop that iterates from 1 to 10 using <% for (let i = 1; i <= 10; i++) { %> ... <% } %>. Inside the loop, there's another conditional operation that checks if each i is even or odd. If i is even, it styles the list item with blue text color; otherwise, it uses the default styling.

**HTML Analysis-**

* When the page is initially rendered, it will not display the results of the conditional operation related to the sum variable because sum is undefined until the form is submitted.
* After the user submits the form and the server processes the input, the .ejs file will be re-rendered with the updated sum value. If sum is even, it will display "The sum [value] is even" in a paragraph; if sum is odd, it will display "The sum [value] is odd" in a paragraph.
* The "List of Numbers" section will display a list of numbers from 1 to 10. Numbers that are even (divisible by 2) will be styled with blue text color, while odd numbers will have the default styling.