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

Diya Liza Varghese

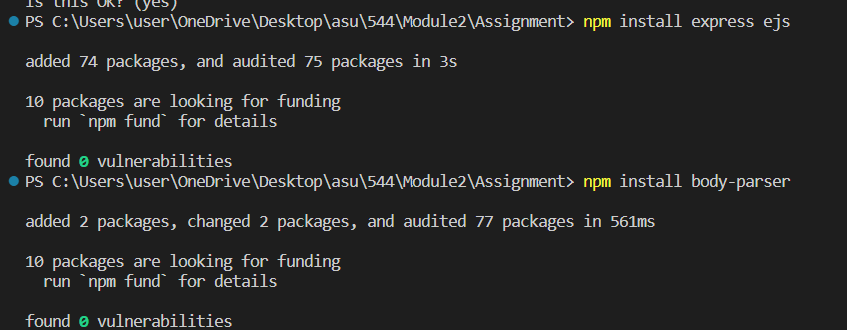
Information Technology, Polytechnic School, Arizona State University

IFT 544: Middleware Prog & Database Sec (2023 Fall)

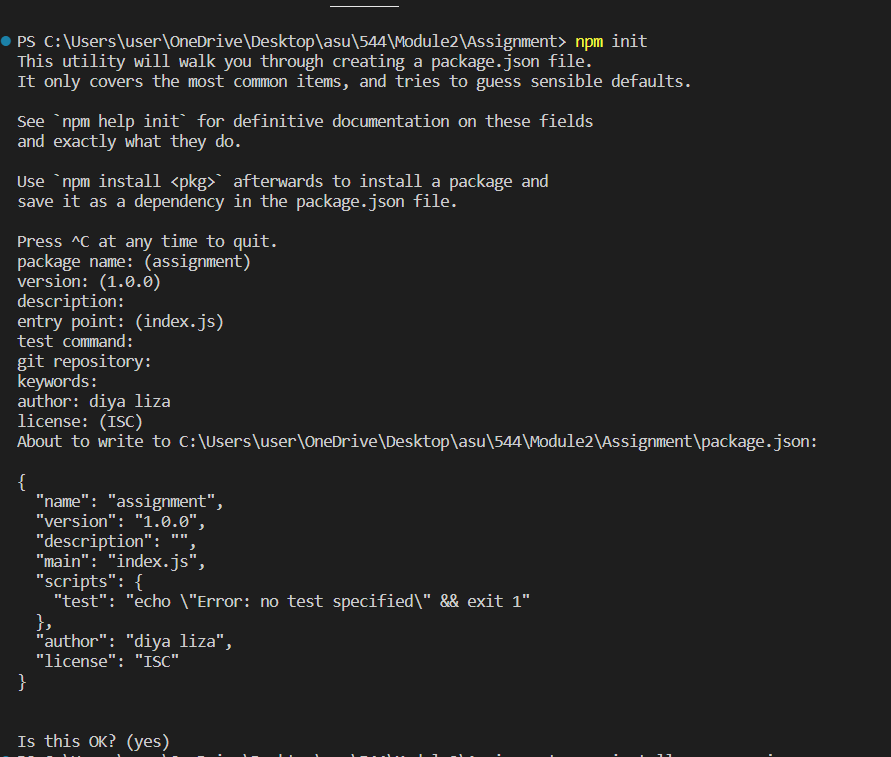
Dr. Dinesh Sthapit

September 03, 2023

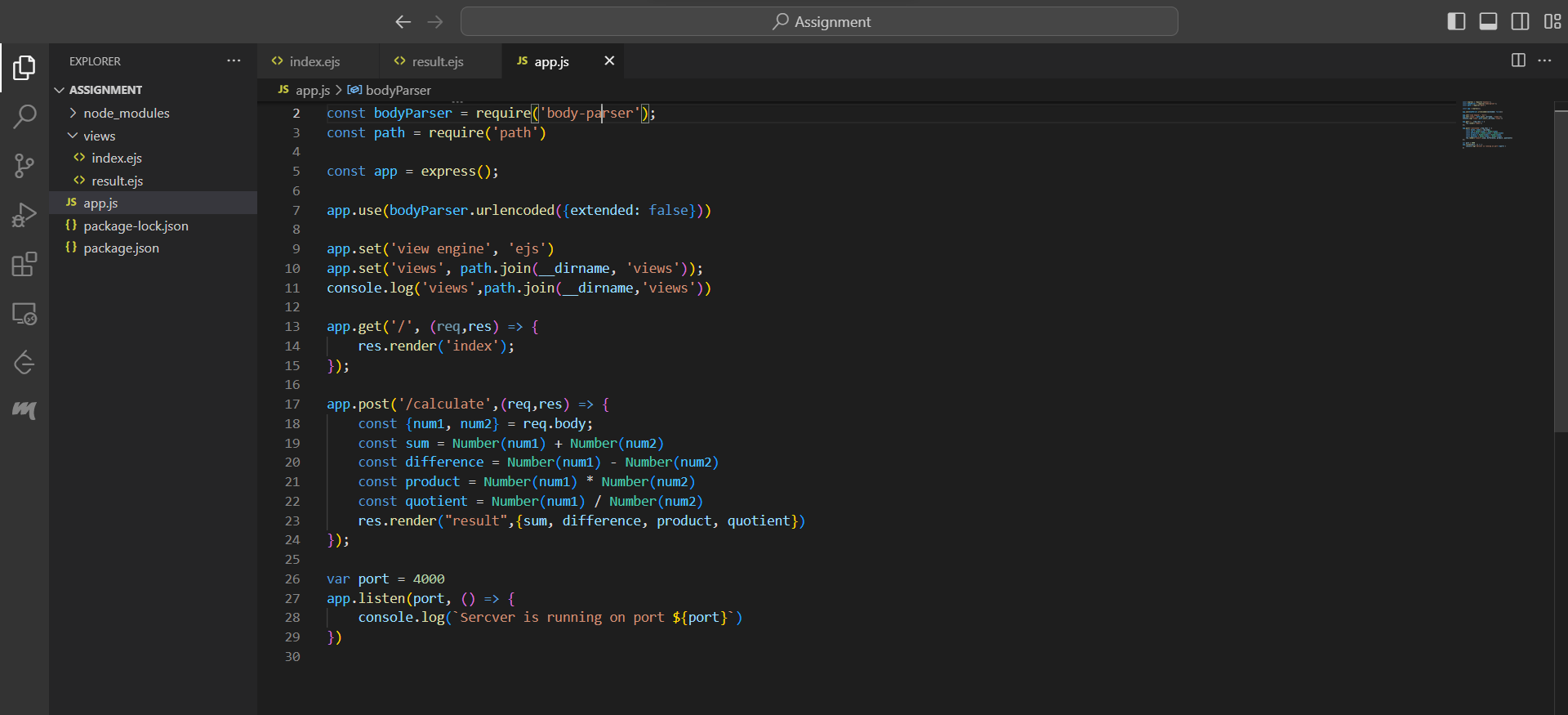
Installing pre-requisites:



Npm init:



App.js



const express = require('express');

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

const path = require('path')

const app = express();

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

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

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

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

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

    res.render('index');

});

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})

});

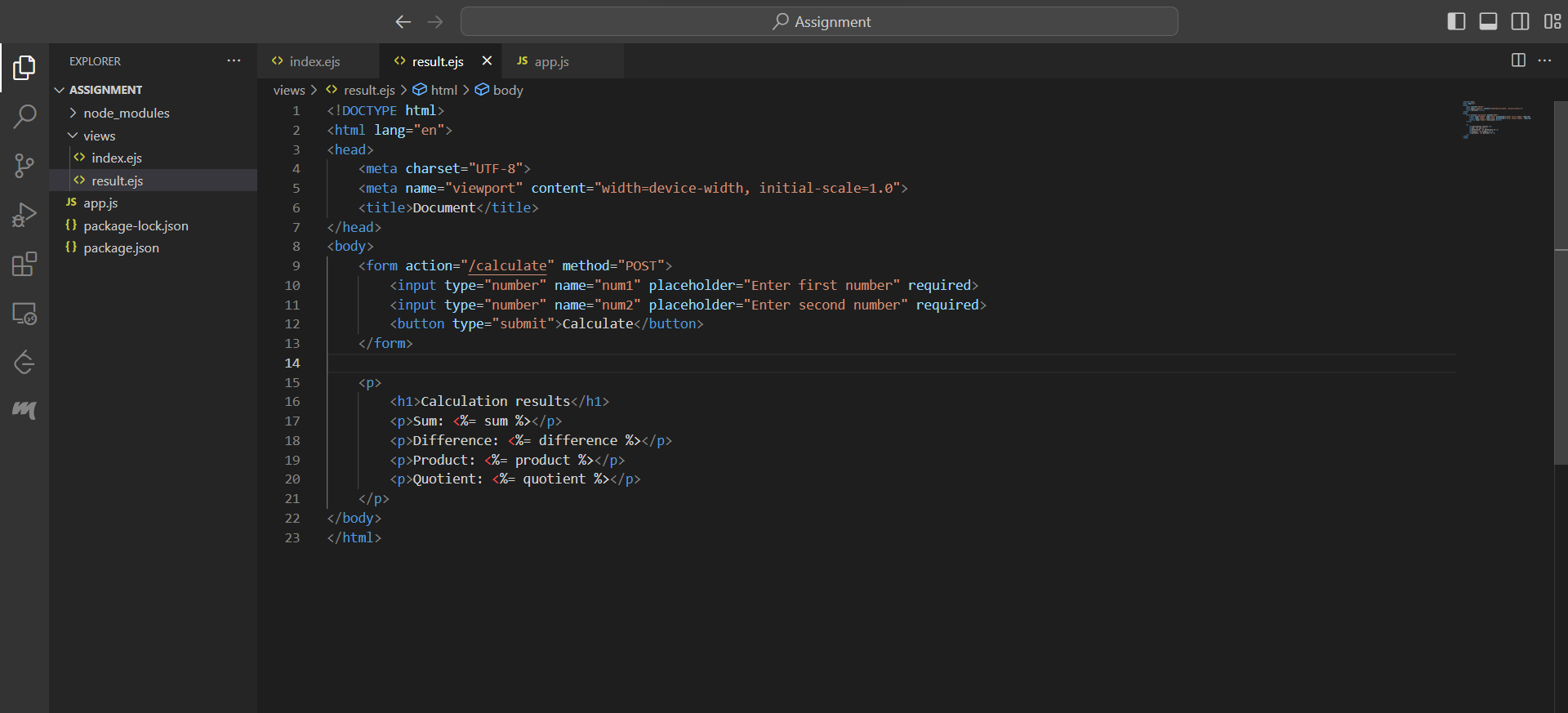
var port = 4000

app.listen(port, () => {

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

})

Result.ejs



<!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>

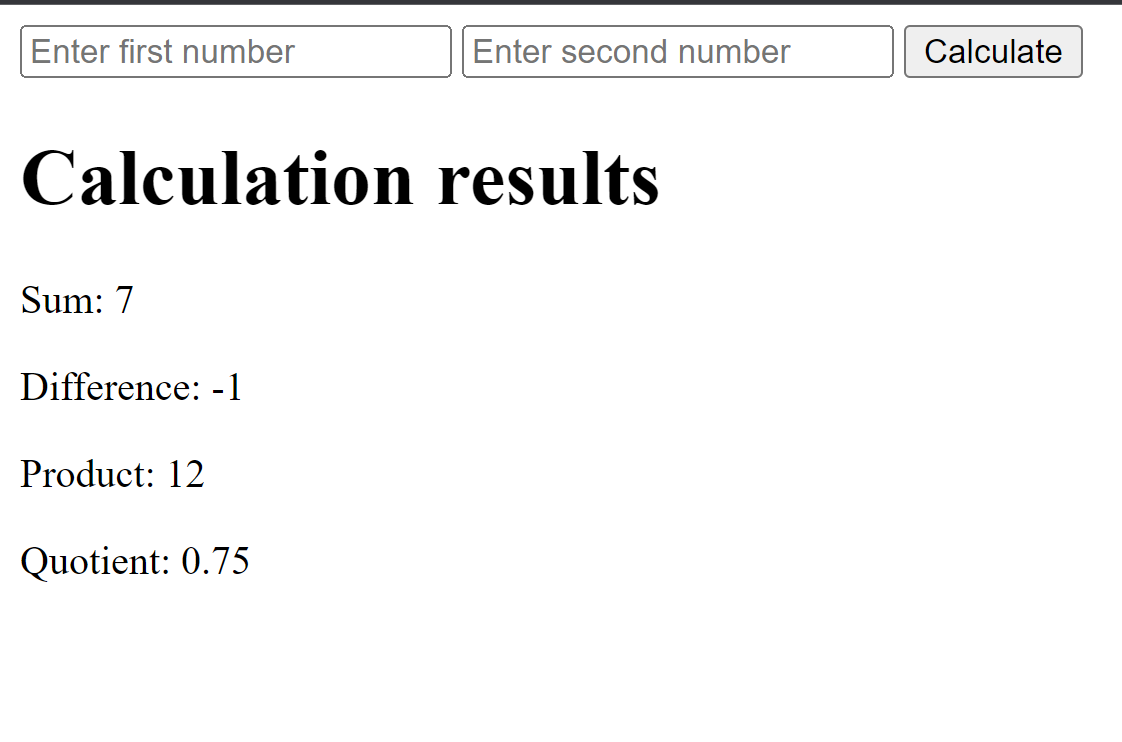
        <p>Quotient: <%= quotient %></p>

    </p>

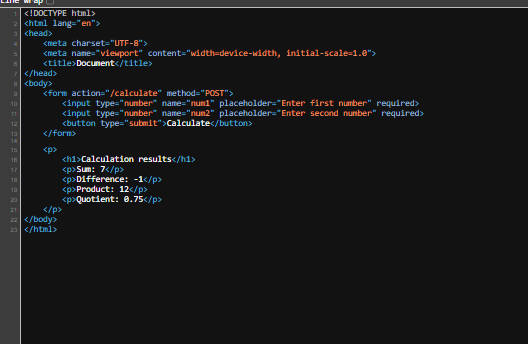
</body>

</html>

Localhost:4000



Html code:



The ejs code is a template file used for dynamically generating HTML element. This is made practical by using <% and %> tag. The code enclosed in these tags are:

  <h1>Calculation results</h1>

        <p>Sum: <%= sum %></p>

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

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

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

The <%= sum %> tag displays the sum of the 2 numbers num1 and num2. When these numbers are passed to the /calculate, the app.post() call calculates the 2 numbers num1 and num2 and returns their sum in the variable called ‘sum’. When it is returned, it is rendered with the other variables, so the in-line code can access the variable using the variable name. The same is the case for other variables also.

The advantage of using the embedded javaScript code is, that, whenever you change the values of num1 and num2, it is reflected in the elements by clicking calculate, it gives dynamic HTML element generation. In order to access a variable, the <% ..%> tag contains the = sign.

This .ejs file serves as a template that can be dynamically populated with calculation results from a server-side script. When a user submits the form, the server processes the input data, performs calculations, and then injects the calculated values into this template to display the results on the corresponding HTML page.

When we click the inspect the code, we get the actual HTML code of the page that is rendered currently. This will not contain the injected javasSript code as it is plain HTML. Here is the comparison of the part that is different in the ejs code and html code:

Ejs code:

<p>

<h1>Calculation results</h1>

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

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

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

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

</p>

Html code:

<p>

<h1>Calculation results</h1>

<p>Sum: 7</p>

<p>Difference: -1</p>

<p>Product: 12</p>

<p>Quotient: 0.75</p>

</p>

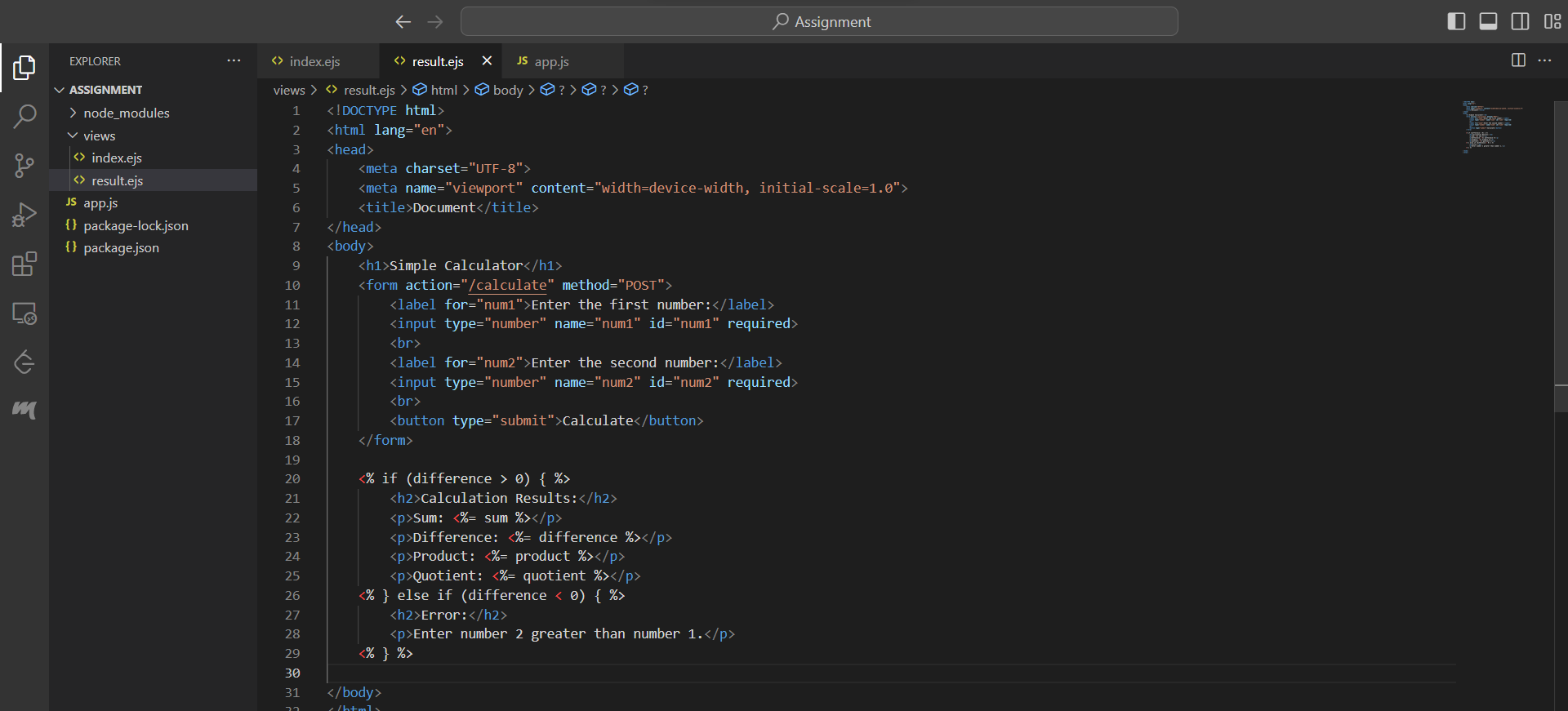
While comparing, the major change is that, the placeholder javaScript injections have been replaced by actual numbers that are calculated based on the numbers that are input by the user. The other html elements and tags do not change, they stay the same while the html code is seen. The key takeaway is that the .ejs file serves as a template that is filled in with actual data on the server side, resulting in the generation of dynamic HTML content when the page is loaded in the browser.

Advantages of EJS over static HTML code:

1. Generate content based on the dynamic demand. This can be very helpful in the websites that have their content changed frequently- for example, the news websites. It can keep changing by keeping the same layout.
2. Code reusability is a very important advantage of ejs over static webpage. It gives a clean and uniform look.
3. With the code reusability, comes the advantage of making quick changes. These changes can be brought out by making the changes only in one template and then, the rest of the web pages will change that makes use of the same template, thereby avoiding the effort of making changes everywhere.
4. It supports conditional rendering, so based on few conditions, we can render different contents. For example, if we want to render different contents for users with normal permissions and admin permissions, we can do the conditional rendering using ejs.

These are few of the many advantages ejs has over the static HTML rendering.

Optional Enhancement



<!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>

    <h1>Simple Calculator</h1>

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

        <label for="num1">Enter the first number:</label>

        <input type="number" name="num1" id="num1" required>

        <br>

        <label for="num2">Enter the second number:</label>

        <input type="number" name="num2" id="num2" required>

        <br>

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

    </form>

    <% if (difference > 0) { %>

        <h2>Calculation Results:</h2>

        <p>Sum: <%= sum %></p>

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

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

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

    <% } else if (difference < 0) { %>

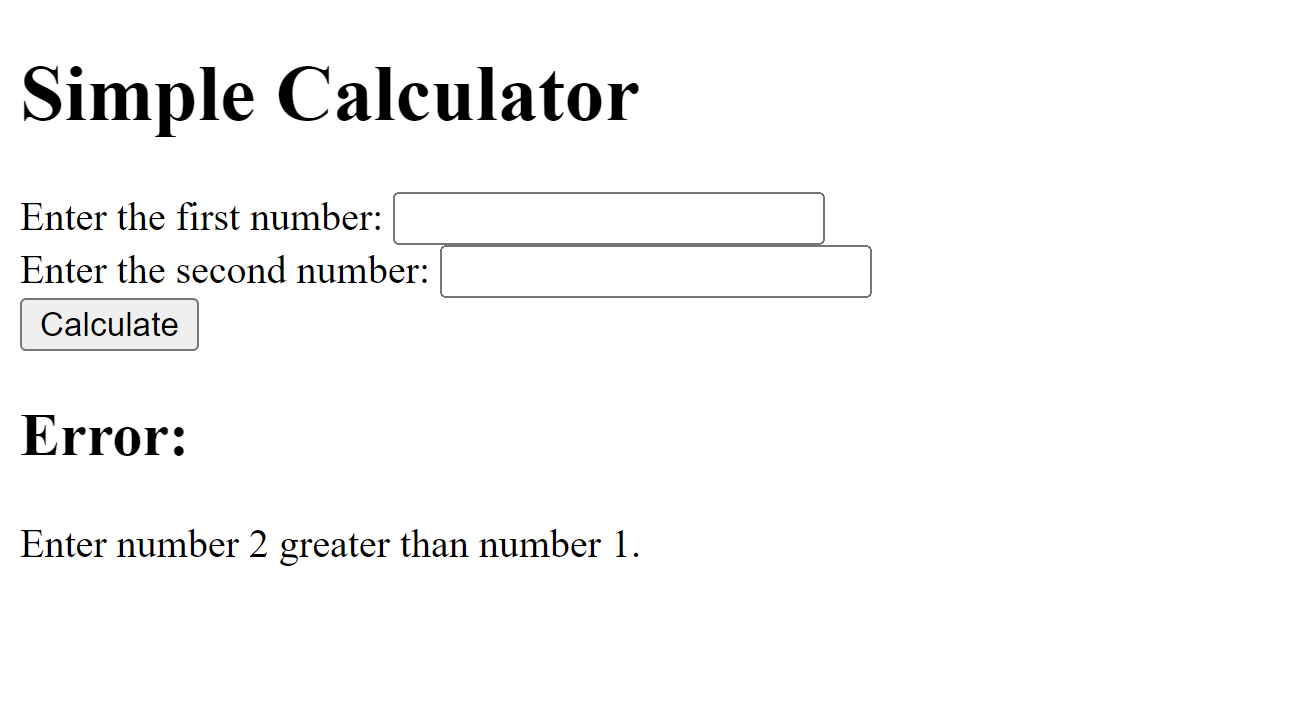
        <h2>Error:</h2>

        <p>Enter number 2 greater than number 1.</p>

    <% } %>

</body>

</html>



In this code, the code renders 2 different html code when the number 2 is greater than number 1 and when number 2 is less than number 1.

This is the html code that is generated when number 2 is greater than number 1:

