

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

Justin Dyer

Ira A. Fulton Schools of Engineering, Arizona State University

IFT 458 Middleware Prog & Database Sec (2023 Fall)

Professor Dinesh Sthapit

September 3rd, 2023

```

1 // Import the required modules
2 const express = require('express');
3
4
5 // create an instance of express
6 const app = express();
7
8
9 // set view engine to ejs
10 app.set('view engine', 'ejs');
11 app.set('views', path.join(__dirname, 'views'));
12 console.log('Views', path.join(__dirname, 'views'));

```

```

PS C:\Users\Pocke\Documents\IFT-458\Module 2\M2A2> npm install express ejs
added 74 packages, and audited 75 packages in 2s
10 packages are looking for funding
run 'npm fund' for details
found 0 vulnerabilities

```

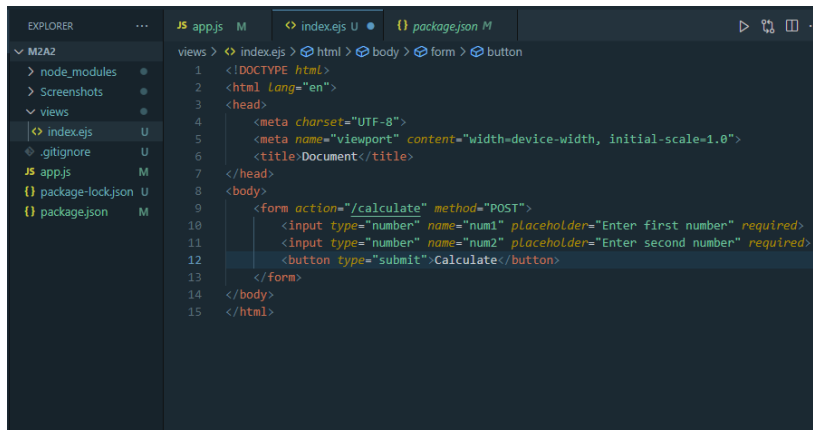
The first image shows the new app.js JavaScript file in the M2A2 folder along with the node_modules folder produced from installing express, ejs, and body-parser. I also wrote the code that imports express and creates an instance of express named app. I then use the instance to set the view engine to ejs.

```

1 // Student Name: Justin Dyer
2 // Student ID: 1216117409
3 // Date: 09/02/23
4
5 // Import the required modules
6 const express = require('express');
7 const bodyParser = require('body-parser');
8 const path = require('path');
9
10 // Create an instance of express
11 const app = express();
12
13 // We use the 'body-parser' middleware to parse the incoming request bodies
14 app.use(bodyParser.urlencoded({ extended: false }));
15
16 // Set the view engine to ejs
17 app.set('view engine', 'ejs');
18 app.set('views', path.join(__dirname, 'views'));
19 console.log('views', path.join(__dirname, 'views'));
20
21 // create a route for the home page
22 // The GET route for the form
23 app.get('/', (req, res) => {
24   // Render the form and pass in the current student data
25   res.render('index');
26 });
27
28 // create a route for user to enter the numbers
29 app.post('/calculate', (req, res) => {
30   const { num1, num2 } = req.body;
31   const sum = Number(num1) + Number(num2);
32   const difference = Number(num1) - Number(num2);
33   const product = Number(num1) * Number(num2);
34   const quotient = Number(num1) / Number(num2);
35   res.render('result', { sum, difference, product, quotient });
36 });
37
38 var port = 4000
39
40
41 // Start the server on port 3000
42 app.listen(port, () => {
43   console.log('Server is running on port ${port}');
44 });
45

```

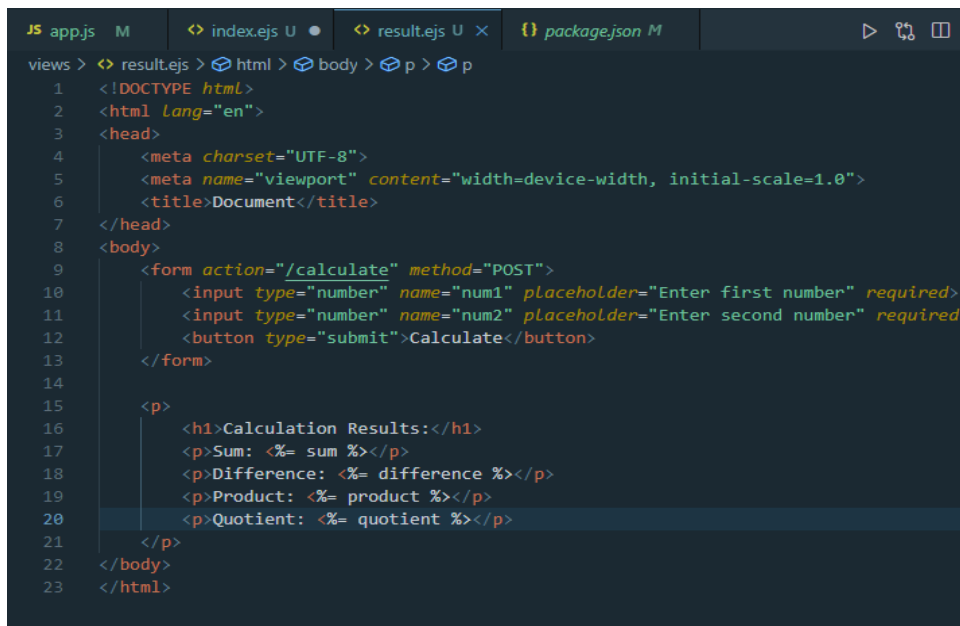
Next the code from the app.js file of module 1 assignment 2 was pasted into the app.js file. The GET route was updated to render the index page and the POST route was modified to the calculate page and the result is rendered to the result page. We changed the port to a variable named port with an assigned value of 4000 and told the app instance to listen to the port number assigned to var port.



The screenshot shows the VS Code interface with the Explorer on the left and the Editor on the right. The Explorer shows a project structure with folders like 'node_modules', 'Screenshots', 'views', and files like 'index.ejs', '.gitignore', 'app.js', 'package-lock.json', and 'package.json'. The Editor displays the content of 'index.ejs', which is an HTML template with a form for calculating the sum, difference, product, and quotient of two numbers. The form has two input fields for numbers and a submit button. The template includes a DOCTYPE declaration, HTML and head tags, and a body tag containing the form.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7 </head>
8 <body>
9   <form action="/calculate" method="POST">
10     <input type="number" name="num1" placeholder="Enter first number" required>
11     <input type="number" name="num2" placeholder="Enter second number" required>
12     <button type="submit">Calculate</button>
13   </form>
14 </body>
15 </html>
```

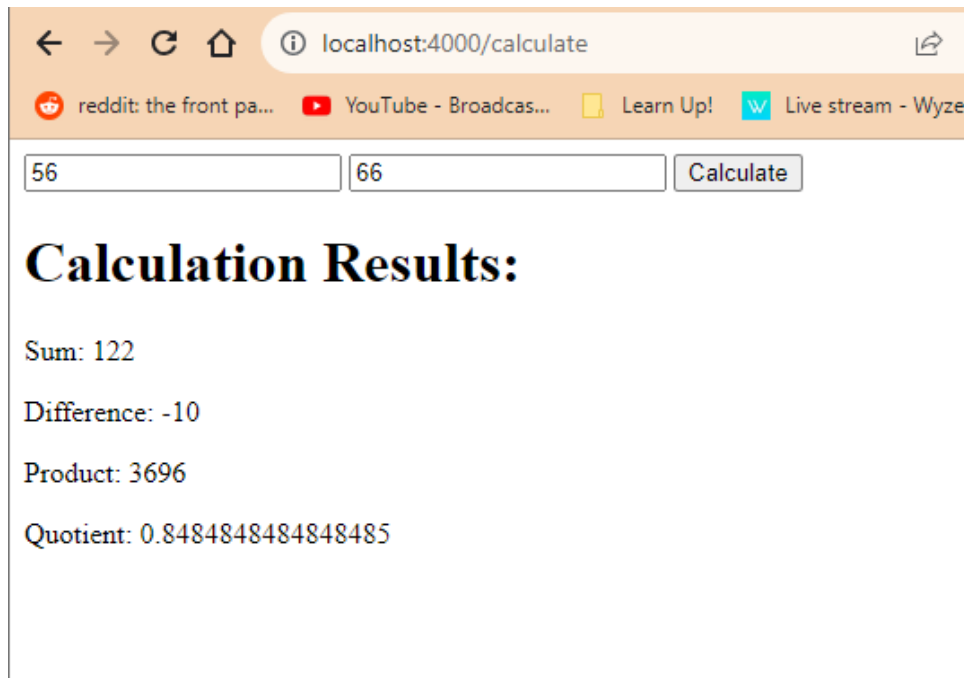
Here we have created the views folder and added an embedded JavaScript (EJS) file named 'index.ejs'. This is the index page that will display to the user allowing them to input numbers.



The screenshot shows the VS Code interface with the Explorer on the left and the Editor on the right. The Explorer shows a project structure with folders like 'node_modules', 'Screenshots', 'views', and files like 'index.ejs', '.gitignore', 'app.js', 'package-lock.json', and 'package.json'. The Editor displays the content of 'results.ejs', which is an HTML template for displaying calculation results. The template includes a DOCTYPE declaration, HTML and head tags, and a body tag containing a form for calculating the sum, difference, product, and quotient of two numbers. The form has two input fields for numbers and a submit button. The template also includes a section for displaying the results, with placeholders for the sum, difference, product, and quotient.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7 </head>
8 <body>
9   <form action="/calculate" method="POST">
10     <input type="number" name="num1" placeholder="Enter first number" required>
11     <input type="number" name="num2" placeholder="Enter second number" required>
12     <button type="submit">Calculate</button>
13   </form>
14
15   <p>
16     <h1>Calculation Results:</h1>
17     <p>Sum: <%= sum %></p>
18     <p>Difference: <%= difference %></p>
19     <p>Product: <%= product %></p>
20     <p>Quotient: <%= quotient %></p>
21   </p>
22 </body>
23 </html>
```

Then we create another EJS file named 'results.esj' that will display the results to the user after they submit calculations. This uses <%= %> tags to use the Javascript variables from our 'app.js' file to update the webpage dynamically after the user submits.



The screenshot shows a web browser at the URL `localhost:4000/calculate`. At the top, there are navigation buttons (back, forward, refresh, home) and a search bar. Below the search bar, there are several tabs: 'reddit: the front pa...', 'YouTube - Broadcas...', 'Learn Up!', and 'Live stream - Wyze'. The main content area has two input fields, one containing '56' and the other '66', followed by a 'Calculate' button. Below the inputs, the text 'Calculation Results:' is displayed in a large, bold font. Underneath, the following calculations are shown: 'Sum: 122', 'Difference: -10', 'Product: 3696', and 'Quotient: 0.84848484848485'.

The resulting page after entering 56 and 66 into the form and submitting. The numbers were passed into the arrow function when we POST, the `req.body` contains values `num1` and `num2` which are calculated into `const sum`, `difference`, `product`, and `quotient`. We then render the page with the EJS template from `'result.ejs'` and this uses the `const` values we passed in.

```
<p>
  <h1>Calculation Results:</h1>
  <p>Sum: <%= sum %></p>
  <p>Difference: <%= difference %></p>
  <p>Product: <%= product %></p>
  <p>Quotient: <%= quotient %></p>
</p>
</body>
```

This shows the EJS code within the `'results.esj'` file in visual studio code. Note the syntax

`<$= 'variable name here'$>`

```
<p>
  <h1>Calculation Results:</h1>
  <p>Sum: 122</p>
  <p>Difference: -10</p>
  <p>Product: 3696</p>
  <p>Quotient: 0.8484848484848485</p>
</p>
```

This is the same section in the chrome inspector view. We see that the formatting characters are gone and instead, we have the values of the const variables displayed as HTML. This shows the advantages of using EJS instead of static HTML. With EJS we can create templates for certain cases and display the necessary information dynamically using user input.

app.js

```
// Student Name: Justin Dyer

// Student ID: 1216117409

// Date: 09/02/23


// 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'));

// create a route for the home page

// The GET route for the form
app.get('/', (req, res) => {

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

    res.render('index.ejs');

});

// 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 });

})

var port = 4000

// Start the server on port 3000

app.listen(port, () => {

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

})
```

index.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>

</body>

</html>
```

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>
```