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

IFT 544: Middleware Prog & Database Sec

Dinesh Sthapit

3 Sep 2023

**Exercise 1: Variables:**

**Output:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Step1: Setup:**

Package.json and installation of express,ejs and body parser

A screenshot of a computer

Description automatically generated

**Step2: .ejs Code Analysis:**

1. Identify the parts of the file that contain JavaScript code, surrounded by <% and %> tags:

In EJS files, JavaScript code is embedded within <% %> tags. Here are the parts of .ejs files that contain JavaScript code:

Result.ejs:

<p>

<h1>Calculation Results:</h1>

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

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

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

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

</p>

2. Analyze how JavaScript code is embedded within the .ejs file to dynamically generate HTML content:

JavaScript code within the .ejs files is used to embed dynamic data into the HTML content. In result.ejs, JavaScript variables like sum, difference, product, and quotient are inserted into the HTML using <%= variableName %> syntax. This allows us to display calculation results dynamically on the webpage.

3. Identify and explain any control structures, such as loops or conditionals, used in the .ejs file:

there are no control structures like loops or conditionals used in index.ejs and result.ejs.

4. Observe how variables are accessed and displayed within the HTML content using <%= variableName %> syntax:

In result.ejs, we access variables like sum, difference, product, and quotient and display them in the HTML content as follows:

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

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

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

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

These variables are replaced with their actual values when the webpage is rendered.

5. Take note of any EJS-specific syntax or features used in the .ejs file:

The primary EJS-specific syntax used in the .ejs files is the <%= variableName %> syntax for embedding variables into the HTML content. This allows for dynamic content generation.

**Step3: HTML Code Analysis:**

1.Open the HTML file in a browser or a code editor:

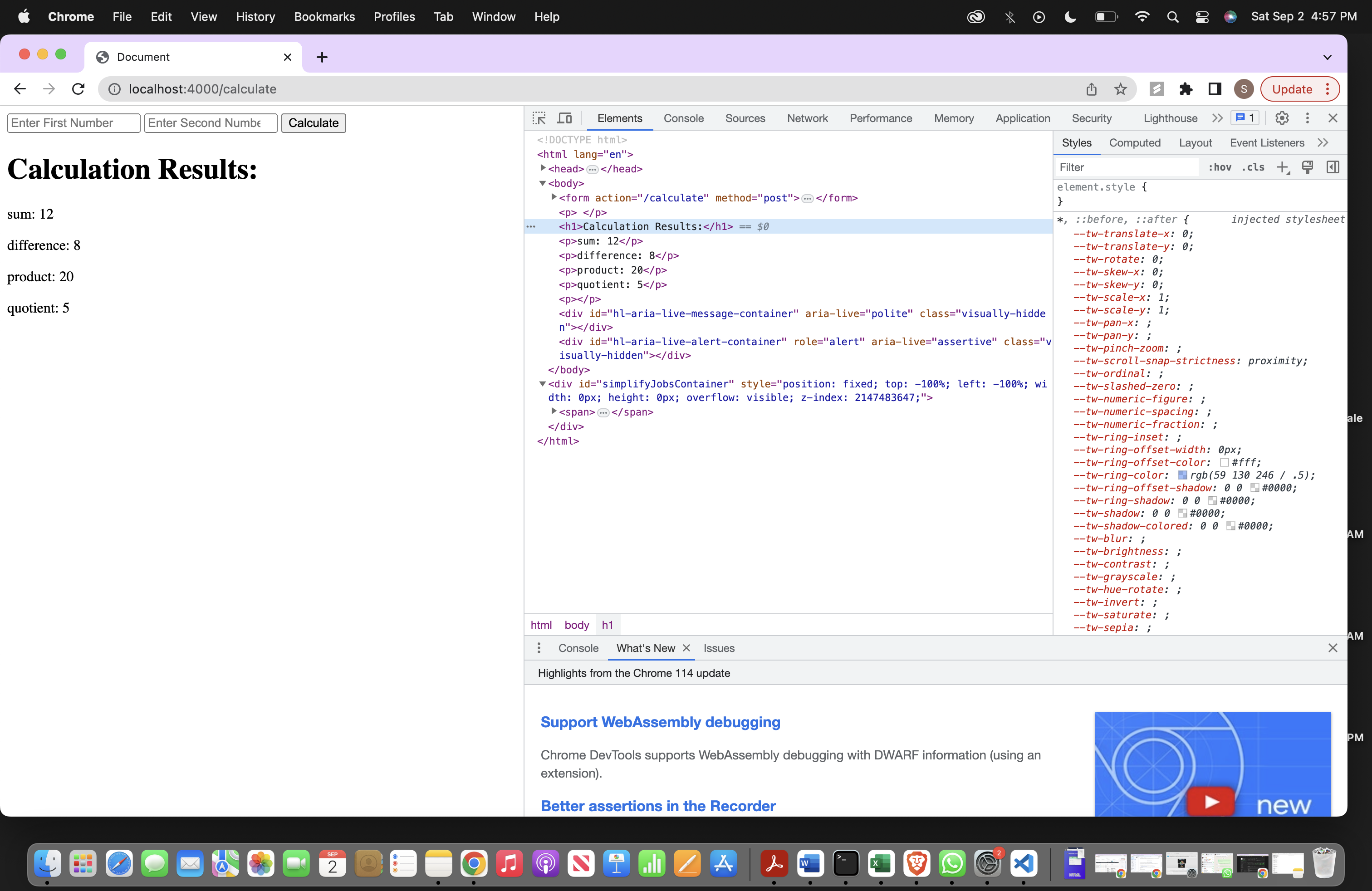
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

2. Inspect the rendered HTML code in the browser or view the HTML file's source code:



3.Identify the parts of the HTML code that were dynamically generated by the .ejs file:

HTML rendered code:

<p>

</p><h1>Calculation Results:</h1>

<p>sum: 12</p>

<p>difference: 8</p>

<p>product: 20</p>

<p>quotient: 5</p>

</p>

In "result.ejs" file contain dynamically generated content:

<p>

<h1>Calculation Results:</h1>

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

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

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

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

</p>

The <%= sum %> and similar expressions will be replaced with actual values when the template is rendered.

4.Compare the HTML code to the corresponding JavaScript code in the .ejs file:

In "result.ejs" file, you have HTML mixed with EJS code. The EJS code is enclosed in <% %> or <%= %>. It's used to inject dynamic content into the HTML.

For example, <%= sum %> in the HTML code corresponds to JavaScript code that calculates the 'sum' variable, which comes from the data sent from the server.

5.Identify any differences or modifications made to the original .ejs code in the rendered HTML:

The primary difference between the original .ejs code and the rendered HTML is that the EJS code in the .ejs file is replaced with actual values when the HTML is generated.

For instance, if you have <%= sum %> in the .ejs file, in the rendered HTML, it will appear as something like <p>sum: 12</p> if the 'sum' value is 12.

**Step4: documentation:**

1.Analysis Summary:

Variables code is a simple Express.js web application that consists of two views: index.ejs and result.ejs. It takes two numbers as input from a form, performs basic arithmetic operations, and displays the results on the result page.

2.Comparison: .ejs vs. HTML:

a. index.ejs (.ejs Code):

<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>

HTML Code Generated in the Browser:

<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>

A screenshot of a computer

Description automatically generated

b. result.ejs (.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 Generated in the Browser:

<p>

</p><h1>Calculation Results:</h1>

<p>sum: 12</p>

<p>difference: 8</p>

<p>product: 20</p>

<p>quotient: 5</p>

<p></p>

A screenshot of a computer

Description automatically generated

3.Observations and Differences:

In the .ejs code, we use <%= variable name%> to embed dynamic content (e.g., <%= sum %>) that will be replaced with actual values when the page is rendered.

The HTML code generated in the browser shows the results with actual values calculated on the server.

4.Benefits of Using .ejs Templates:

Dynamic Content: .ejs templates allow us to inject dynamic data into your HTML, making it easier to display results or user-specific information.

Reusability: .ejs templates can be reused across different pages, reducing code duplication.

Maintainability: It separates code logic (in JavaScript) from the presentation (in .ejs files), making it easier to maintain and update.

5. Scenarios for Using .ejs Templates:

E-commerce: To generate product listings with varying details.

Content Management Systems: For creating and editing content with consistent layouts.

Multi-page Forms: To create complex, multi-step forms with dynamic content.

**Step5: Optional Enhancement:**

Modified result.ejs:

<p>

<h1>Calculation Results:</h1>

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

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

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

<!-- <p>quotient: <%= quotient %></p> -->

<p>

<% if (isFinite(quotient)) { %>

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

<% } else { %>

<p>cannot divide by zero.</p>

<% } %>

</p>

</p>

In this modification, I have added a conditional check to determine if the quotient is a finite number. If it is, we display the quotient; otherwise, we display a message saying it's undefined.

A screenshot of a computer

Description automatically generated

2. Analyze the corresponding changes in the rendered HTML code and document your findings.

I have added a conditional statement using <% if (condition) {%> ... <% } else { %> ... <% } %> to determine whether the quotient is finite or not.

If the quotient is defined, it displays the value of quotient.

If the quotient is undefined, it displays a message, “Cannot divide by zero.”