Full Stack Development Lab Instructions
ENSF381: Lab05

Java Script (String, Arrays, functions)

Created by: Mahdi Jaberzadeh Ansari (He/Him)
Schulich School of Engineering
University of Calgary
Calgary, Canada
mahdi.ansari1@ucalgary.ca

Week 6, February 12/14, 2024

Contents

1 Introduction

1.1 Objectives

Lab 5 is designed to deepen your understanding of JavaScript, an essential language in web development. This lab will focus on three fundamental concepts of JavaScript:

- **Strings:** You will explore various methods to manipulate strings, understanding how to effectively handle text data in your applications.
- **Arrays:** The lab will introduce you to array operations, including creation, modification, and iteration over array elements. This is crucial for managing lists of data in web development.
- **Functions:** You will learn about defining and invoking functions in JavaScript, which are essential for structuring your code into reusable blocks and managing its flow.

1.2 Prerequisites

1. Basic understanding of computer operations.
2. A web browser (Chrome, Firefox, Safari, etc.) to view your HTML file.
3. GitHub account.

1.3 Forming groups

- In this lab session, you **MUST** work with a partner (groups of three or more are not allowed).
- The main goal of working in a group is to learn:
  - how to do teamwork
  - how to not be a single player
  - how to not be bossing
  - how to play for team
  - how to tolerate others
  - how to behave with colleagues
  - how to form a winner team
  - and ...
- Working with a partner usually gives you the opportunity to discuss some of the details of the topic and learn from each other. Also, it will give you the opportunity to practice one of the popular methods of program development called pair programming. In this method, which is normally associated with the "Agile Software Development" technique, two programmers normally work together on the same workstation (you may consider a Zoom session for this purpose). While one partner, the driver, writes the code, the other partner, acting as an observer, looks over his or her shoulder, making sure the syntax and solution logic are correct. Partners should switch roles frequently in such a way that both have equivalent opportunities to practice both roles. **Please note that you MUST switch roles for each exercise.**
- When you have to work with a partner:
  - Choose a partner that can either increase your knowledge or transfer your knowledge. (i.e., do not find a person with the same programming skill level!)
  - Please submit only one lab report with both names. Submitting two lab reports with the same content will be considered copying and plagiarism.

1.4 Before submission

- For most of the labs, you will receive a DOCX file that you need to fill out the gaps. Make sure you have this file to fill out.
- All your work should be submitted as a single file in PDF format. For instructions about how to provide your lab reports, study the posted document on the D2L called How to Hand in Your Lab assignment.
- Please note that if it is group work, only one team member must submit the solution in D2L. For ease of transferring your marks, please mention the group member's name and UCID in the description window of the submission form.
- If you have been asked to write code (HTML, CSS, JS, etc.), make sure the following information appears at the top of your code:
  - File Name
  - Assignment and exercise number
  - Your names in alphabetic order
  - Submission Date:

  Here is an example for CSS and JS files:

  ```
  <!--
  =========================================================
  Name         : lab5_exe_B.html
  Assignment   : Lab 5, Exercise B
  Author(s)    : Mahdi Ansari, William Arthur Philip Louis
  Submission   : May 21, 2030
  Description  : JS Basics.
  =========================================================
  -->
  ```

- Some exercises in this lab and future labs will not be marked. Please do not skip them, because these exercises are as important as the others in learning the course material.
- In courses like ENSF381, some students skip directly to the exercises that involve writing code, skipping sections such as "Read This First," or postponing the diagram-drawing until later. That's a bad idea for several reasons:
  - "Read This First" sections normally explain some technical or syntax details that may help you solve the problem or may provide you with some hints.
  - Some lab exercises may ask you to draw a diagram, and most of the students prefer to hand-draw them. In these cases, you need to scan your diagram with a scanner or an appropriate device, such as your mobile phone, and insert the scanned picture of your diagram into your PDF file (the lab report). A possible mobile app to scan your documents is Microsoft Lens, which you can install on your mobile device for free. Please make sure your diagram is clear and readable; otherwise, you may either lose marks or it could be impossible for TAs to mark it at all.
  - Also, it is better to use the Draw.io tool if you need to draw any diagram.
  - Drawing diagrams is an important part of learning how to visualize data transfer between modules and so on. If you do diagram-drawing exercises at the last minute, you won't learn the material very well. If you do the diagrams first, you may find it easier to understand the code-writing exercises, so you may be able to finish them more quickly.
- **Due Dates:**

  - You must submit your solution until 11:59 p.m. on the same day that you have the lab session.
  - Submissions until 24 hours after the due date get a maximum of 50% of the mark, and after 24 hours, they will not be evaluated and get 0.

1.5 Academic Misconduct/Plagiarism

- Ask for help, but don't copy.
  - You can get help from lab instructor(s), TAs, or even your classmates as long as you do not copy other people's work.
  - If we realize that even a small portion of your work is a copy from a classmate, both parties (the donor and the receiver of the work) will be subject to academic misconduct (plagiarism). More importantly, if you give exercise solutions to a friend, you are not doing him or her a favor, as he or she will never learn that topic and will pay off for this mistake during the exam or quiz. So, please do not put yourself and your friend in a position of academic misconduct.
  - You can use ChatGPT, but please note that it may provide similar answers for others too, or even the wrong answers. For example, it has been shown that AI can hallucinate, proposing the use of libraries that do not actually exist [1]. So, we recommend that you imagine ChatGPT as an advanced search engine, not a solution provider.
  - In order to find out who is abusing these kinds of tools, we will eventually push you toward the incorrect responses that ChatGPT might produce. In that case, you might have failed for the final mark and be reported to administration.
  - If we ask you to investigate something, don't forget to mention the source of your information. Reporting without reference can lead to a zero mark even by providing a correct answer.

1.6 Marking Scheme

- You should not submit anything for the exercises that are not marked.
- In Table 1, you can find the marking scheme for each exercise.

Table 1: Marking scheme

Exercise	Marks
A	5 marks
B	15 marks
C	15 marks
D	15 marks
Total	50 marks

1.7 Complains

- Your grades will be posted one week following the submission date, which means they will be accessible at the subsequent lab meeting.
- Normally, the grades for individual labs are assessed by a distinct TA for each lab and section. Kindly refrain from contacting all TAs. If you have any concerns regarding your grades, please direct an email to the TA responsible for that specific lab.

[1] https://perma.cc/UQS5-3BBP

2 Exercise A (Creating a repository)

2.1 Initialize the Project

Like last time, we are going to clone an existing project from a remote repository. Cloning is particularly useful when you want to create a local copy of a remote repository to work on. Follow these steps:

- Create a new repository on GitHub named `Lab5`. Remember to:
  - Check the box for `Add a README file`. This file serves as an introduction and overview of your project.
  - Select `MIT License` for your project's license. This is a permissive license that allows others considerable freedom with your code while still crediting you.
- After creating the repository, it should contain two files: a `README.md` and a `LICENSE` file. The `README.md` is where you can write about your project, and the LICENSE file contains the license text.

2.2 Inviting Collaborators to Your GitHub Repository

Collaborating with others on a project is a key aspect of software development. GitHub allows you to add collaborators to your repository. Here's how you can invite your colleagues to the repository you created in the previous steps:

- First, log in to your GitHub account and navigate to the repository you want to share. In this case, it's the `Lab5` repository.
- On your repository's page, locate the `Settings` tab near the top of the page and click on it. This will take you to the repository settings.
- In the settings menu, on the left side of the page, you will find a section called `Manage access`. Click on it.
- On the `Manage access` page, you will see a button labeled `Invite a collaborator`. Click on this button.
- GitHub will prompt you to enter the GitHub username or email address of the person you want to invite. Enter the username or email of your colleague.
- Once you enter the username or email, GitHub will suggest the matching account. Click on the account to select it.
- After selecting the collaborator, you can set the role for them. For most cases, the `Write` access level is sufficient. This allows them to push changes to the repository but doesn't allow actions like deleting the repository.
- Click on `Add collaborator` to send the invitation. Your colleague will receive an email from GitHub with a link to accept the invitation.
- Inform your collaborator to check their email and accept the invitation to collaborate on your repository.

Once your colleague accepts the invitation, they will have access to the repository and can start collaborating with you on the project. This process is crucial for teamwork in software development, allowing multiple people to work together seamlessly on the same codebase.

Please note: *In the subsequent sections, each exercise should be completed by a single individual. It is imperative to rotate the role of the developer. This means that if Person X performs the steps in Exercise B, then Exercise C must be undertaken by Person Y, and this pattern should continue accordingly. We will check the commits on GitHub to give you marks.*

2.3 Using GitHub's WebIDE for Development

In this subsection, we will learn to use GitHub's WebIDE to develop our project directly on GitHub without the need for a local development environment. This approach simplifies the setup process and allows you to code directly in your browser. Follow these steps to start coding with GitHub's WebIDE:

- Navigate to your `Lab5` repository on GitHub.
- To open your repository in GitHub's WebIDE, simply replace the `github.com` part of your repository's URL with `github.dev` in your browser's address bar. For example, if your repository's URL is `https://github.com/yourusername/Lab5`, change it to `https://github.dev/yourusername/Lab5`. Press Enter to load the WebIDE.
- The GitHub WebIDE will open, showing your repository's files and folders in a code editor interface similar to Visual Studio Code. This interface allows you to edit files, commit changes, and manage branches directly from your browser.
- Start by exploring the interface. On the left side, you'll see a file explorer where you can navigate through your project files. You can open any file by clicking on it, which will open the file in the editor pane.
- You can edit any file by simply typing in the editor pane. The WebIDE provides syntax highlighting and basic code completion features to assist you in writing code.
- To create a new file, right-click on the folder in the file explorer where you want the file to be and select `New File`. Enter the file name (e.g., `index.html`) and press Enter. You can now start coding in your new file.
- Please edit the `README.md` file and add your team members' full names and UCIDs in a table.
- After making changes to your files, you can commit these changes directly from the WebIDE. Click on the Source Control icon (resembling a branching diagram) on the left sidebar. Here, you can stage changes, write a commit message, and commit the changes to your repository.
- Remember to regularly commit your changes with meaningful commit messages. This practice is essential for version control and will help your collaborators understand the changes you make.
- One significant limitation of using GitHub's WebIDE is the absence of a live server-like feature for immediate testing of changes. To view your changes, you typically need to pull the updated code to your local machine. However, integrating Continuous Integration/Continuous Deployment (CI/CD) pipelines can automate this process. When set up, CI/CD allows you to see the changes reflected on a live server almost immediately after they're pushed. For the next exercise, ensure that you have pulled the latest version of the repository using a local Git client (e.g. `SourceTree`), so you are ready to proceed with the hands-on practice.

2.4 Deliverable

1. Add the address of your GitHub repository to your answer sheet file. Please make sure the repository is public and do not delete it until the end of the semester.
2. Make a screenshot from your WebIDE and add it to your answer sheet file.

3 Exercise B (JavaScript Basics: Strings, Operators, and Variables)

3.1 Objective

Learn to integrate JavaScript with HTML to manipulate strings, perform arithmetic operations, and use variables.

3.2 Task Overview

You will create an HTML file and write JavaScript code within it to perform various operations involving strings, operators, and variables.

3.3 Steps

1. **Create an HTML File**:

   - Open your text editor and create a new file.
   - Save the file as `lab5_exe_B.html`.
   - Write the basic HTML structure. Your file should look like this:

   ```
   <!DOCTYPE html>
   <html lang="en">
   <head>
       <meta charset="UTF-8">
       <meta name="viewport" content="width=device-width, initial-
           scale=1.0">
       <title>JavaScript String and Variable Exercise</title>
   </head>
   <body>
       <!-- JavaScript will be linked here -->
   </body>
   </html>
   ```

2. **Add a Script Tag**:

   - Inside the `<body>` tag of your HTML, add a `<script>` tag at the bottom.
   - Your HTML should now look like this:

   ```
   <body>
       <!-- Content goes here -->

       <script>
           // JavaScript code will go here
       </script>
   </body>
   ```

3. **Declare Variables**:

   - Inside the `<script>` tag, declare variables `firstName`, `lastName`, and `age`.
   - Assign some real values to them (use your own real name). For example:

   ```
   let firstName = "John";
   let lastName = "Doe";
   let age = 25;
   ```

4. **Concatenate Strings**:

   - Create a new variable `fullName` and concatenate `firstName` and `lastName`.
   - Use `console.log` to display `fullName`.

   ```
   let fullName = firstName + " " + lastName;
   console.log(fullName);
   ```

5. **Calculate Birth Year**:

   - Declare a variable `birthYear` and calculate the birth year.
   - Assume the current year is 2024 and subtract `age` from it.
   - Log `birthYear` using `console.log`.

   ```
   let currentYear = new Date().getFullYear();
   let birthYear = currentYear - age;
   console.log(birthYear);
   ```

6. **String Manipulation**:

   - Log the length of `fullName`.
   - Convert `fullName` to uppercase and log it.

   ```
   console.log(fullName.length);
   console.log(fullName.toUpperCase());
   ```

7. **Template Literals**:

   - Use template literals to log a detailed message.

   ```
   console.log(`Hello, my name is ${fullName} and I am ${age} years
       old. I was born in ${birthYear}.`);
   ```

8. **Bonus: Conditional Operator**:

   - Use a ternary operator to check if `age` is over 18.
   - Declare `canVote` and set its value based on the condition.
   - Log the voting eligibility.

   ```
   let canVote = age > 18 ? true : false;
   console.log(`I am eligible to vote: ${canVote}`);
   ```

3.4 Testing Your Code

- Open `lab5_exe_B.html` in a web browser.
- Right-click on the page and select "Inspect" or "Developer Tools" to open the console.
- You should see the outputs of your `console.log` statements in the console.

3.5 Deliverables

- Commit and push your `lab5_exe_B.html` file with the embedded JavaScript code into GitHub.
- Make a screenshot from your browser console with all the outputs and include it in the answer sheet file.

4 Exercise C (JavaScript and the DOM)

4.1 Objective

Learn to manipulate the Document Object Model (DOM) in JavaScript. access HTML elements and dynamically change their content and style. This exercise will also involve teamwork, requiring collaboration between two members.

4.2 Read Me First

4.2.1 Understanding the DOM

The DOM is a programming interface for web documents. It represents the page so that programs can change the document structure, style, and content. The DOM represents the document as a tree of objects; each object corresponds to a part of the page (like elements, attributes, and text).

4.2.2 Finding DOM Elements

In JavaScript, there are several methods to find or select DOM elements. These methods allow you to access and manipulate elements within an HTML document. Here's a comprehensive list of common methods for finding DOM elements:

1. **getElementById**
   - Selects an element by its ID. Please note that the ID must be unique for each element, and two elements cannot have a similar ID.
   - Example: `document.getElementById('exampleId')`
2. **getElementsByClassName**
   - Selects all elements that have a specific class name.
   - *Returns a live HTMLCollection.*
   - Example: `document.getElementsByClassName('exampleClass')`
3. **getElementsByTagName**
   - Selects all elements with a specific tag name.
   - *Returns a live HTMLCollection.*
   - Example: `document.getElementsByTagName('p')`
4. **querySelector**
   - Selects the first element that matches a specified CSS selector.
   - Example: `document.querySelector('.exampleClass')`
5. **querySelectorAll**
   - Selects all elements that match a specified CSS selector.
   - Returns a static NodeList.
   - Example: `document.querySelectorAll('div.exampleClass')`
6. **getElementsByName**
   - Selects elements by their `name` attribute.
   - Often used with radio buttons.

   - Example: `document.getElementsByName('exampleName')`

7. **closest**
   - Used on an element to find the nearest ancestor that matches a given selector.
   - Example: `element.closest('.parentClass')`

8. **matches**
   - Checks if an element would be selected by a specified selector string.
   - Example: `element.matches('.matchClass')`

9. **children**
   - Gets the child elements of a specified element.
   - Returns an HTMLCollection of child elements.
   - Example: `element.children`

10. **parentElement**
    - Gets the parent element of a specified element.
    - Example: `element.parentElement`

11. **nextSibling** and **previousSibling**
    - Gets the next or previous sibling element.
    - Example: `element.nextSibling`, `element.previousSibling`

12. **getElementsByTagNameNS** (for XML namespaces)
    - Similar to `getElementsByTagName`, but for elements with a specified XML namespace.
    - Example: `document.getElementsByTagNameNS('http://www.w3.org/1999/xhtml', 'div')`

13. **getAttribute** and **setAttribute**
    - To get or set attributes of an element.
    - Example: `element.getAttribute('href')`, `element.setAttribute('href', 'https://example.com')`

Each of these methods serves different purposes and can be chosen based on the specific requirements of your task. Remember that methods like `getElementsByClassName` and `getElementsByTagName` return a live HTMLCollection, which means it automatically updates when the document changes. In contrast, `querySelectorAll` returns a static NodeList, which does not reflect changes made to the DOM after it is generated.

4.2.3 An Example

Here is an example that shows how we can use these methods. There is a `DOMExample.html` file in the D2L that you can run and see how this code works.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DOM Selection Testing</title>
</head>
<body>
```

```
    <div id="uniqueId">Element with Unique ID</div>
    <div class="commonClass">First Element with Common Class</div>
    <div class="commonClass specialClass">Second Element with Common Class
        and Special Class</div>
    <p>First Paragraph</p>
    <p>Second Paragraph</p>
    <input type="text" name="inputName" value="First Input by Name">
    <input type="text" name="inputName" value="Second Input by Name">
    <button onclick="testDOMMethods()">Test DOM Methods</button>
    <p id="msg" style="color:red; visibility: hidden;">Please check the
       Console to see extracted elements.</p>
    <script>
        function testDOMMethods() {
            console.log('getElementById:', document.getElementById('
               uniqueId'));
            console.log('getElementsByClassName (commonClass):', document.
               getElementsByClassName('commonClass'));
            console.log('getElementsByClassName (specialClass):', document
               .getElementsByClassName('specialClass'));
            console.log('getElementsByTagName (p):', document.
               getElementsByTagName('p'));
            console.log('querySelector (commonClass):', document.
               querySelector('.commonClass'));
            console.log('querySelectorAll (commonClass):', document.
               querySelectorAll('.commonClass'));
            console.log('getElementsByName (inputName):', document.
               getElementsByName('inputName'));
                        document.getElementById('msg').style.visibility =
                           'visible';
        }
    </script>
</body>
</html>
```

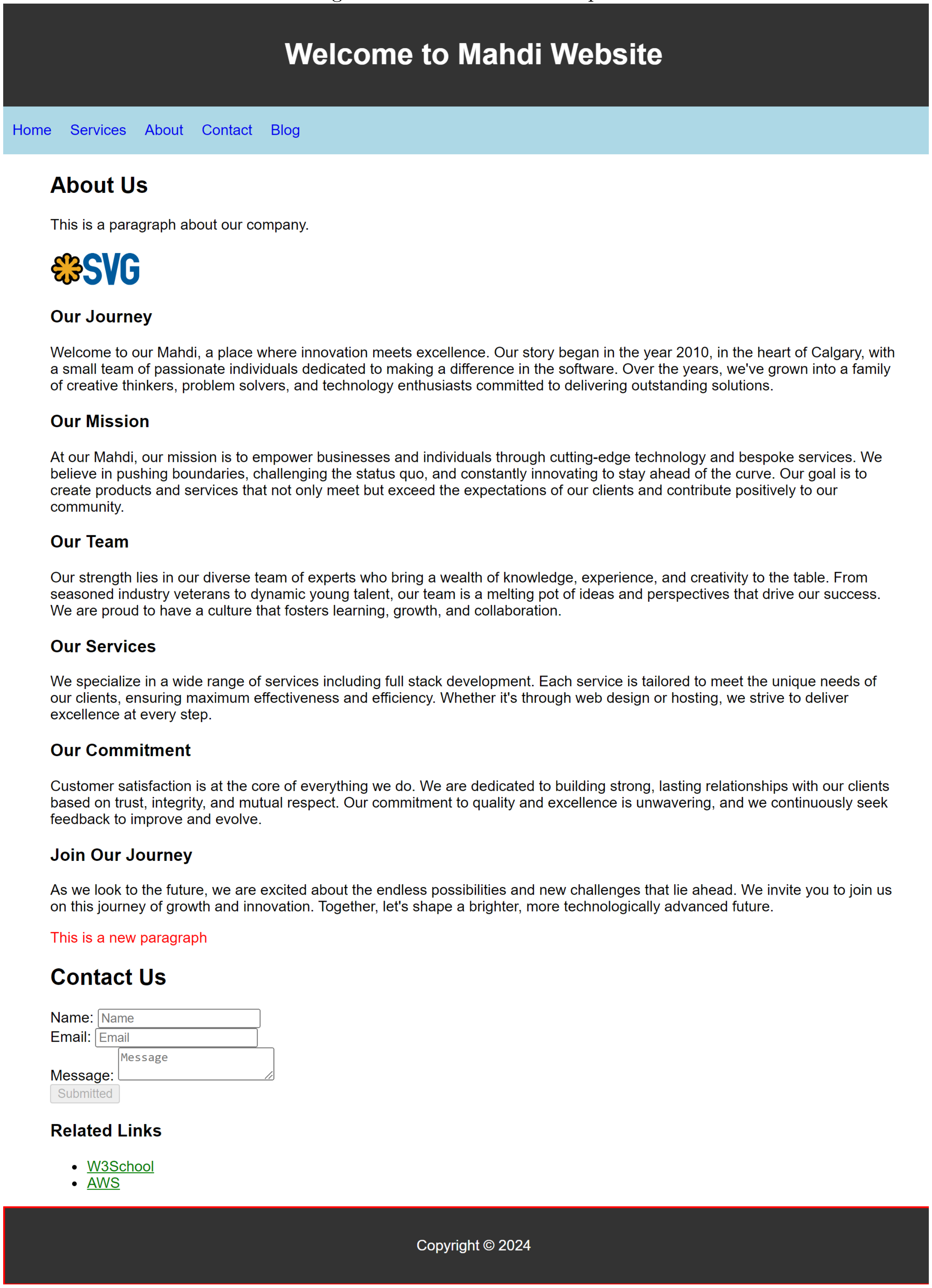
4.3 Task Overview

In this lab exercise, you will be manipulating the DOM of a provided HTML file using JavaScript. Open the file `lab5_exe_C.html` and write your code in the provided function within the `<script>` tag. Figure 1 shows the expected final output of your HTML file.

4.3.1 Tasks

1. Set the text content of all elements with the class `companyName` to “Your Name” and remove their borders. For example I set it to “Mahdi”.
2. Change the background color of the navigation bar to a color of your choice.
3. Hide the section titled “What is full-stack?”.
4. Create a new `<p>` element with the text “This is a new paragraph” with red color and append it to the end of the “About Us” section.
5. Add placeholders to input elements based on their corresponding label IDs. For example, the place holder for the `<input type="text" id="name" name="name" />` must be `Name`.
6. Set the `src` attribute of the image to point to the *./images/example.svg*.

Figure 1: Exercise C final output



Welcome to Mahdi Website

Home Services About Contact Blog

About Us

This is a paragraph about our company.

Our Journey

Welcome to our Mahdi, a place where innovation meets excellence. Our story began in the year 2010, in the heart of Calgary, with a small team of passionate individuals dedicated to making a difference in the software. Over the years, we've grown into a family of creative thinkers, problem solvers, and technology enthusiasts committed to delivering outstanding solutions.

Our Mission

At our Mahdi, our mission is to empower businesses and individuals through cutting-edge technology and bespoke services. We believe in pushing boundaries, challenging the status quo, and constantly innovating to stay ahead of the curve. Our goal is to create products and services that not only meet but exceed the expectations of our clients and contribute positively to our community.

Our Team

Our strength lies in our diverse team of experts who bring a wealth of knowledge, experience, and creativity to the table. From seasoned industry veterans to dynamic young talent, our team is a melting pot of ideas and perspectives that drive our success. We are proud to have a culture that fosters learning, growth, and collaboration.

Our Services

We specialize in a wide range of services including full stack development. Each service is tailored to meet the unique needs of our clients, ensuring maximum effectiveness and efficiency. Whether it's through web design or hosting, we strive to deliver excellence at every step.

Our Commitment

Customer satisfaction is at the core of everything we do. We are dedicated to building strong, lasting relationships with our clients based on trust, integrity, and mutual respect. Our commitment to quality and excellence is unwavering, and we continuously seek feedback to improve and evolve.

Join Our Journey

As we look to the future, we are excited about the endless possibilities and new challenges that lie ahead. We invite you to join us on this journey of growth and innovation. Together, let's shape a brighter, more technologically advanced future.

This is a new paragraph

Contact Us

Name: Name
Email: Email
Message: Message
Submitted

Related Links

- W3School
- AWS

Copyright © 2024

7. Add a new item in the navigation menu with the text “Blog” and link it to `#`.
8. Upon clicking the “Submit” button in the contact form, change the button text to “Submitted” and make it disabled.
9. For each item in the “Related Links” aside, change the text color to `green`.
10. Add a solid border to the footer with red color and 2 pixels width.

4.4 Deliverables

1. The final `lab5_exe_C.html` file must be pushed into your GitHub repository.
2. Make a screenshot from your final output and include it in the answer sheet file.

5 Exercise D (JavaScript Arrays and Functions)

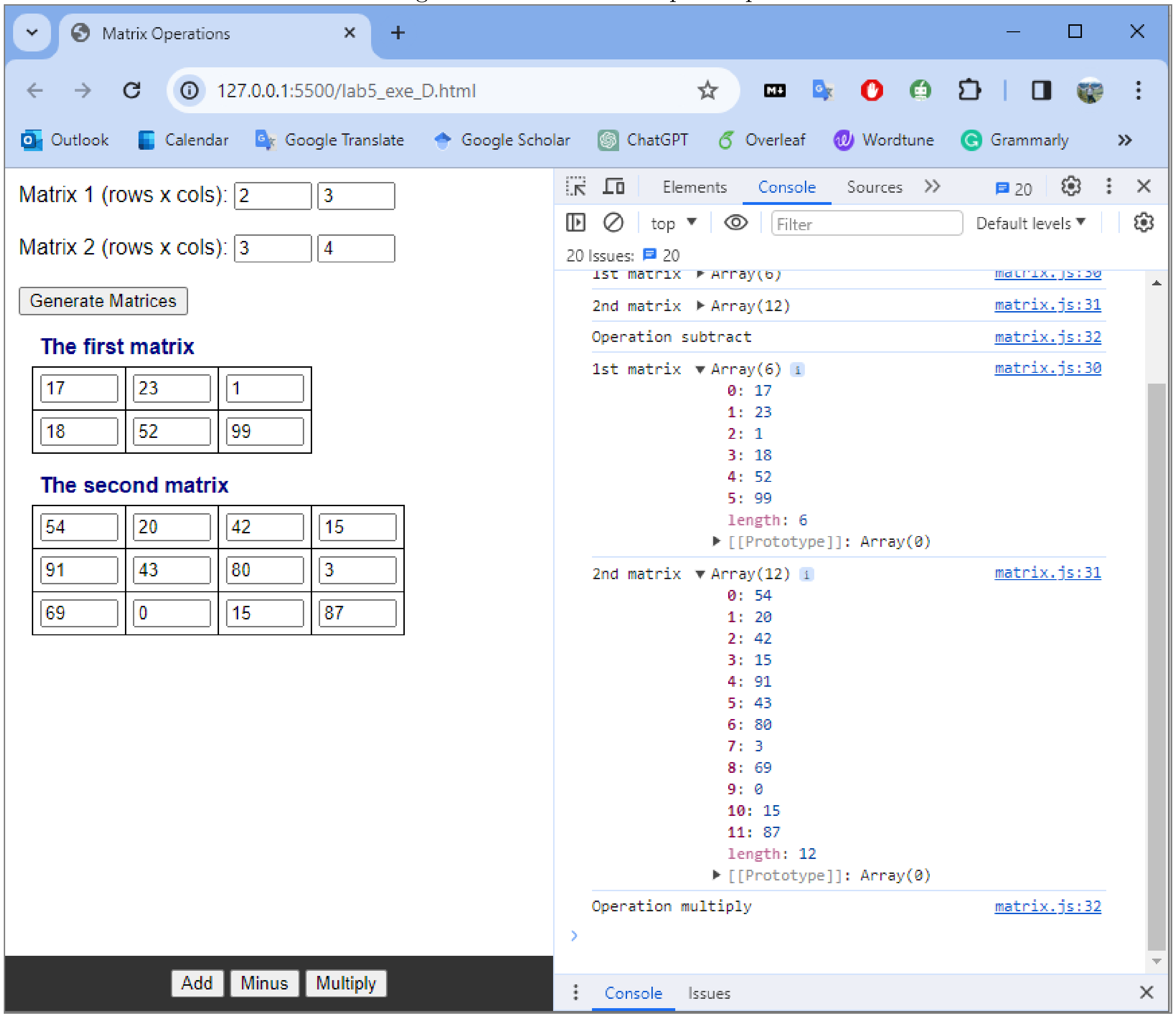
5.1 Objective

The goal of this lab exercise is to practice manipulating and performing operations on arrays in JavaScript. You will be completing a partially implemented file named `matrix.js`.

5.2 Tasks

1. Open the file `lab5_exe_D.html` in a web browser. This file contains the interface for matrix input and operation selection. Figure 2 shows a sample execution of the `lab5_exe_D.html` file.
2. Your task is to complete the implementation of the `matrix.js` file, which is linked to the HTML file. Focus on the following functions:
   - Matrix Calculation Functions:
     - `function addMatrices(matrix1, matrix2)`: Implement matrix addition.
     - `const subtractMatrices = function (matrix1, matrix2)`: Implement matrix subtraction.
     - `const multiplyMatrices = (matrix1, matrix2)`: Implement matrix multiplication. Remember to check the dimensions of the matrices to ensure that the operation is possible.
     - Please note that we have different possibilities for defining functions or methods in JavaScript.
   - `showResult2D(title, containerId, dataArray)`:
     - Implement this function to display the result of matrix operations. The `dataArray` parameter is a 2D array representing the result matrix. Use the `showResult` function as a reference for implementation.
   - After implementing the matrix operations, remove the sample result array and use your implemented functions. Display the results using the `showResult2D` function that you have implemented.
   - Ensure that your functions include checks for the feasibility of the operations (e.g., matrix dimension compatibility) and print suitable messages for impossible situations.
3. Test your implementation thoroughly to ensure accuracy and robustness.

Figure 2: Exercise D sample output



Matrix 1 (rows x cols): 2 3

Matrix 2 (rows x cols): 3 4

Generate Matrices

The first matrix

17	23	1
18	52	99

The second matrix

54	20	42	15
91	43	80	3
69	0	15	87

```
1st matrix  ▸ Array(6)                          matrix.js:30
2nd matrix  ▸ Array(12)                         matrix.js:31
Operation subtract                              matrix.js:32
1st matrix  ▾ Array(6)                          matrix.js:30
                0: 17
                1: 23
                2: 1
                3: 18
                4: 52
                5: 99
                length: 6
              ▸ [[Prototype]]: Array(0)
2nd matrix  ▾ Array(12)                         matrix.js:31
                0: 54
                1: 20
                2: 42
                3: 15
                4: 91
                5: 43
                6: 80
                7: 3
                8: 69
                9: 0
                10: 15
                11: 87
                length: 12
              ▸ [[Prototype]]: Array(0)
Operation multiply                              matrix.js:32
```

5.3 Deliverable

1. After completing the implementation, save your changes to `matrix.js` and push the file to your GitHub repository.
2. Add a snapshot of one execution to your answer sheet file.
3. Add the completed code of the (`showResult2D`, `addMatrices`, `subtractMatrices`, `multiplyMatrices`) functions to your answer sheet file.