ENSF-381: Full Stack Web Development Laboratory

Ahmad Abdellatif and Novarun Deb
Department of Electrical & Software Engineering
University of Calgary

Lab 4

Objectives

Welcome to the ENSF381 course lab! Below are the detailed instructions for creating and understanding JavaScript. The main objective of this lab is to understand the fundamental JavaScript concepts and their practical applications. We will develop skills in working with variables, strings, and dates, creating and using functions, and managing structured data with arrays, objects, and destructuring them. By completing the tasks and testing their code in the browser's developer console, students will gain hands-on experience in writing dynamic and efficient JavaScript code.

Groups

Lab instructions must be followed in groups **of two students**.

Submission

You must submit the complete source code, ensuring it can be executed without any modifications. Also, if requested by the instructor, you may need to submit the corresponding documentation file (e.g., word and image). Only one member of the group needs to submit the assignment, but the submission must include the names and UCIDs of all group members at the top of the code.

Deadline

Lab exercises must be submitted by **11:55 PM on the same day as the lab session**. Submissions made within 24 hours after the deadline will receive a maximum of 50% of the mark. Submissions made beyond 24 hours will not be evaluated and will receive a grade of zero.

Academic Misconduct

Academic Misconduct refers to student behavior which compromises proper assessment of a student's academic activities and includes: cheating; fabrication; falsification; plagiarism; unauthorized assistance; failure to comply with an instructor's expectations regarding conduct required of students completing academic assessments in their courses; and failure to comply with exam regulations applied by the Registrar.

For more information on the University of Calgary Student Academic Misconduct Policy and Procedure and the SSE Academic Misconduct Operating Standard, please visit: https://schulich.ucalgary.ca/current-students/undergraduate/student-resources/policies-and-procedures

Exercise 1: Exploring JavaScript Fundamentals

Objective: We will practice fundamental JavaScript concepts, including working with variables, strings, template literals, and dates, to develop a strong foundation in the language. You will execute the tasks in your browser's developer console to observe the output.

Create an HTML File

1. Open a text editor.
2. Create a new file and save it as exercise1.html.
3. Add the basic structure of an HTML document:

```
<!DOCTYPE html>
<html>
<head>
    <title>Lab4 - Exercise 1</title>
</head>
<body>
    <h1>JavaScript Lab</h1>
    <script>
        // Write your JavaScript code here
    </script>
</body>
</html>
```

Using the Developer Console

Before starting with Part 1, we need to familiarize ourselves with the browser's developer console, where we will view the JavaScript outputs.

1. **Open the Developer Console**:
    - In your web browser (e.g., Chrome, Firefox, or Edge), right-click anywhere on the page and select Inspect or Inspect Element**.**
    - Navigate to the Console tab in the developer tools panel that appears.
2. **Write and Test JavaScript Code**: JavaScript code placed inside the <script> tag of your HTML file will run automatically when the page loads, with any outputs or errors appearing in the console.
3. **Clear the Console**: to make it easier to track your work, clear the console between tests by clicking the Clear Console button (a trash can icon) or pressing Ctrl+L (Windows/Linux) or Cmd+K (Mac).

Variable Declaration

1. Declare three variables using `let`, and `const`. Assign each a different value.
2. Change the value of the variable declared using `let` and observe the behaviour.
3. Try to reassign the value of the `const` variable and note what happens.
4. Comment out the line written in the last step (i.e., Step 3).

String Operations

1. Declare two string variables:

```
string1 = "Hello, World!"
string2 = "JavaScript is fun!"
```

2. Use a method to find the length of string1 and display it in the console.
3. Concatenate string1 and string2 and log the result in the console.
4. Replace the word “fun” in string2 with “awesome” and display the modified string in the console.

String Templates

1. Declare the following variables for the tasks in this part:

```
name = "Alice"
age = 25
```

2. Create a string using backticks (`` ` ``). The string should read:
   `"My name is Alice and I am 25 years old"`
   Hint: Use the variables `name` and `age` inside the string by inserting them with the `${variableName}` syntax.
3. Log the resulting template string in the console.

Working with Dates

1. Create a variable to store the current date using JavaScript's Date object. This is a new concept, so you may need to research the Date object online for this lab**.**
2. Log the full date in the console.
3. Use methods from the Date object to:
   - Extract and log the year (e.g., 2025).
   - Extract and log the month as a number (e.g., 2 for February).
   - Extract and log the day of the week as a number (e.g., 1 for Monday).
   - Convert the day of the week to its name (e.g., “Monday”). Hint: Use an array of weekday names and the day number as an index.

Submission

1- Fill out the Answer_sheet.docx, including the names and UCIDs of all group members.
2- Create a new GitHub repository and upload the code for Exercise 1 to the new repository.

Exercise 2: Building a Simple Calculator

Objective: In this exercise, we will work on creating and experimenting with JavaScript functions to deepen your understanding of their concepts and usage. Follow the instructions carefully and test your code using the browser's developer console.

Declaring Functions

1. Repeat the steps from "Create an HTML File" in Exercise 1 to create a new file named exercise2.html. In the newly created file, write a function named add that accepts two numbers as parameters and returns their sum.
2. Write another function named subtract that accepts two numbers as parameters and returns their difference.
3. Use your browser's developer console to call each function with sample inputs and confirm the results.

Using Default Parameters

1. Copy and modify your add function to include a default value for the second parameter (e.g., 0). **You need to comment out the "add" and "subtract" functions declared in the previous section "Declaring Functions".**
2. Similarly, modify the subtract function to include a default value for the second parameter.
3. Test the updated functions by calling them with one and two arguments in the developer console. Observe the behaviour when you omit the second argument.

Implementing Arrow Functions

1. Rewrite the add and subtract functions using arrow function syntax.
2. Define two new functions, multiply and divide, using arrow function syntax.
   - The multiply should return the product of two numbers.
   - The divide function should return the quotient of two numbers. Ensure that division by zero is prevented.

Test all four functions in the developer console to verify their functionality.

Understanding Callback Functions

1. Create a function named calculator that accepts three parameters:
   - Two numbers.
   - A callback function, which represents the mathematical operation to be performed (e.g., addition, subtraction, multiplication, or division).
2. Inside calculator, call the provided callback function with the two numbers and return the result.
3. Test your calculator function in the developer console by passing it:
   - Numbers and the add function as the callback.

- Numbers and the multiply function as the callback.

Submission:
Upload the code for Exercise 2 to the same repository you created for Exercise 1.

Exercise 3: Managing Data with JavaScript

Objective: The objective of this exercise is to help students practice managing and manipulating data in JavaScript using arrays, objects, and destructuring. We will learn how to create, update, and access structured data, enabling them to build dynamic and organized programs.

Arrays and Basic Methods

1. Repeat the steps from "Create an HTML File" in Exercise 1 to create a new file named exercise3.html.
2. Declare an array named classRoster containing the following student names: Alice, Tom, Charlie, Diana, Evan
3. Use the toString method to convert the array into a string and log it to the console.
4. Log the initial classRoster to show that it remains unchanged after using toString.
5. Use a method (e.g., push) to add two more students: Fiona and Nancy.
6. Remove the first student in the list (Alice) using the shift method. **Do not** remove a name by directly editing the array declaration. Hint: the shift method is used to remove the first element from an array and return it.
7. Log the updated array to the console and confirm all changes.
8. Log the length of the updated classRoster array to confirm the total number of students.

Objects with Nested Structures

1. Create an object named `classInfo` with the following properties:
    a. `className`: A string, `'ENSF381: Full-Stack Web Development'`.
    b. `instructor`: A string, `'Dr. Smith'`.
    c. `students`: The `classRoster` array created earlier.
    d. `details`: A nested object with these properties:
        - `semester`: `'Winter'`
        - `year`: `2025`
2. Add a new property, `schedule`, to the `classInfo` object. This property should be an array containing the class meeting days: Monday, Wednesday, Friday.
3. Update the `instructor` property to 'Dr. Abdellatif' or 'Dr. Deb' depending on which section you are enrolled in. Use the appropriate syntax for this change (e.g., classInfo.instructor = 'Dr. Abdellatif'). **Do not** directly re-declare the object.
4. Log the values of the `className`, instructor, and `students` properties to the console.
5. Access and log the `semester` property from the nested details object.
6. Log the updated `classInfo` object to confirm all changes.
7. Destructure the `className` and `students` properties from `classInfo` into individual variables. Then, log these variables to the console to confirm their values.
8. Destructure the `semester` and year `properties` from the nested details object in classInfo into variables. Then, log these variables to confirm they are correctly assigned.
9. Destructure the first two student names from the `classRoster` array into variables named *`student1`* and *`student2`*.

10. After destructuring, `student1` and `student2` should contain the first two names, and `remainingStudents` should include the rest.
11. Log `student1`, `student2`, and `remainingStudents` to confirm their values.

Submission:

1. Upload the code for Exercise 3 to the same repository you created for Exercise 1.
2. Compress both Exercise 1, Exercise 2, and Exercise 3 into a single file and upload the compressed file to D2L. Also, upload the completed Answer_sheet.docx.