CMPS 350 Web Development Fundamentals
Lab 5 – JavaScript Fundamentals

Objective

The objective of this lab is to introduce you to various JavaScript concepts, including variables and expressions, control structures, arrays, functional programming, arrow functions, array methods, spread operators, and objects.

Overview

This Lab is divided into four parts:

- Exercise 1: A warm-up exercise to reinforce basic JavaScript syntax.
- Exercise 2: Practice arrays and control structures.
- Exercise 3: Practice arrow functions and array functions.
- Exercise 4: Create a simple shopping app using the concepts learned in exercises 1–3.

Preparation

1. If not already done, install the latest version of Node.js on your laptop from https://nodejs.org/en/.
2. Install Code Runner extension so that you can right-click and run .js files.
3. Install Prettier - Code formatter extension to keep your JS code nicely formatted.

Exercise 1 – Basic Syntax Warm up JS exercises

1. Add a **`Lab5-JS`** folder to your repository.
2. Create a JavaScript file named **`exercise1.js`** inside the Lab5-JS folder.
3. Using a `while loop`, write a JavaScript program that displays odd numbers from 1 to 100.
4. Rewrite the first program using a `for loop`.
5. Declare the following array: `const cars = ["Toyota", "Honda", "BMW"]`.
    - Add "Volvo" to the end of the array.
    - Add "Mercedes" to the beginning of the array.
    - Create a **`display`** function that takes an array as an argument and displays the array elements using a **`for-of`** loop.
    - Call the **`display`** function to display the cars array.
    - Sort the array alphabetically and print it again.

Exercise 2 – Array Functions

1. Declare the following matrix (i.e., array of arrays):

```
const matrix = [[2, 3], [34, 89], [55, 101, 34], [34, 89, 34, 99]]
```

2. Implement and test the following functions:
    - **flatten**: gets a matrix and returns a single-dimensional flat array.
    - **max**: gets an array and returns its maximum value (tip: use reduce function).

- **sort**: gets an array and returns a sorted array in descending order (from big to small).
- **square**: gets an array and returns an array with squared values.
- **average**: gets an array and returns its average.
- **distinct**: gets an array and returns an array without duplicate elements.
- **sum**: computes the sum of array elements that are greater than 40. You should write everything as a single statement.

Expected output:

```
Original array:
[ [ 2, 3 ], [ 34, 89 ], [ 55, 101, 34 ], [ 34, 89, 34, 99 ] ]

Flattened:
[ 2, 3, 34, 89, 55, 101, 34, 34, 89, 34, 99 ]

Max value:
101

Sorted in descending order:
[ 101, 99, 89, 89, 55, 34, 34, 34, 34, 3, 2 ]

Sum of elements greater than 40:
433

Unique elements:
[ 2, 3, 34, 89, 55, 101, 99 ]

Sum of unique elements:
383

Square of unique elements:
[ 4, 9, 1156, 7921, 3025, 10201, 9801 ]
```

Exercise 3 – Objects

1. Create a new project named **student-management**
2. Create a JavaScript file named **app.js**
3. Open your terminal and type the following command: **`npm install --save prompt-sync`** to download and install the **`prompt-sync`** package, that allows you to read user input from the terminal.
4. Open the `package.json` and add the following line **`"type" : "module",`**
5. Import the **prompt-sync package** inside the app.js

```
import promptSync from 'prompt-sync';
const prompt = promptSync()
```

6. Implement a function named **getStudents** that prompts the user to input the name and gender of 5 students. Within the function, assign a random age between 17 and 35, and a random grade between 0 and 100 for each student. Store the information for each student in an array of objects,

where each object represents a student and contains their name, gender, age, and grade. Return the array of student objects.

7. Inside the app.js create an empty array named **students** and call the **getStudents**.

   Here is an example of how the students array should look like after calling getStudents students:

```
[
  { name: 'Ahmed', gender: 'Male', grade: 85, age: 23 },
  { name: 'Sara', gender: 'Female', grade: 77, age: 31 },
  { name: 'Hassen', gender: 'Male', grade: 92, age: 29 },
  { name: 'Zahra', gender: 'Female', grade: 63, age: 20 },
  { name: 'Abdulahi', gender: 'Male', grade: 54, age: 22 }
]
```

8. Then write and test functions that return the following:
   a. The youngest student in the class
   b. The oldest student in the class
   c. The average student age in the class
   d. The median student age in the class
   e. The mean of the student grades
   f. The variance of the student grades

   Note: Solve the above questions using array functions instead of traditional loops.

Mean	Variance
$\bar{x} = \frac{1}{n}\sum_{i=0}^{n-1} x_i$	$\sigma^2 = \frac{1}{n-1}\sum_{i=0}^{n-1}(x_i - \bar{x})^2$

   g. Get students by gender to return either male of female students.
   h. Display the student information sorted in alphabetical order by name
   i. Display the student information sorted in descending order by grade
   j. Determine if there are any students with failing grades (below 60)
   k. Find the student(s) with the highest grade
   l. Find the student(s) with the highest grade among the female students
   m. Find the average grade for male students
   n. Display the student information with an additional property that indicates whether the student has a passing grade (60 or higher).

Exercise 4 – Shopping App

In the following practice exercise, you will apply the concepts from the previous exercises to create a simple shopping app that allows users to add an item to the cart, change item quantity, delete an item, and display the invoice.

The app has a list of products such as:

```
const products = [
      { id: 1, name: 'Apple 14 Pro Max', price: 4500},
      { id: 2, name: 'iPad Pro 12.9-inch', price: 5600},
      { id: 3, name: 'Samsung Galaxy S14', price: 3900},
      { id: 4, name: 'Microsoft Surface Book 3', price: 6700}
      { id: 5, name: 'Sony PlayStation 5', price: 3500}
      { id: 6, name: 'Dell XPS 13', price: 4500}
      { id: 7, name: 'LG 65-inch OLED TV', price: 9800}
      { id: 8, name: 'Bose QuietComfort 35 II', price: 1800}
]
```

Menu

What would you like to do?

1. Add Product
2. Change quantity
3. Delete product
4. Display invoice

Option 1 – Add Item

Display the list of available products and allow the use to enter the desired products and quantities to add the shopping cart (the user can add multiple products to the cart). In case an item already exists in the cart, you should increment only the quantity to avoid adding duplicate items to the cart.

Option 2 – Change Quantity

The user will be presented with a list of items currently in their cart along with their quantities. The user can then select an item to update and provide a new quantity for that item. The corresponding item in the cart will be updated with the new quantity. If the selected item does not exist in the cart, display a message indicating that the item could not be found in the cart.

Option 3 – Delete Item

Display the list of items in the cart and allow them to input the product id to be removed from the cart. If the item exists in the cart, remove it, and display a message to confirm its removal. If the item does not exist, display a message indicating that the item could not be found in the cart.

Option 4 – Display Invoice

Display the cart items along with their quantities and prices. Also display the total price for all items in the cart. Highlight the most expensive and the least expensive items in the cart with an asterisk and a double asterisk, respectively.

Your implementation should be modular and it should use array functions, arrow functions, objects and other JS features.