CMPS 350 Web Development Fundamentals
Lab 5 – JavaScript Fundamentals

Objective

The objective of this Lab is to practice the following JavaScript skills:

- Variables and expressions
- Control structures
- Arrays
- Functional programming including:
    - **Arrow functions**
    - **Array methods** (`map`, `reduce`, `filter`, `flat`, `splice`, `sort`…)
    - **Spread operator**
- Objects

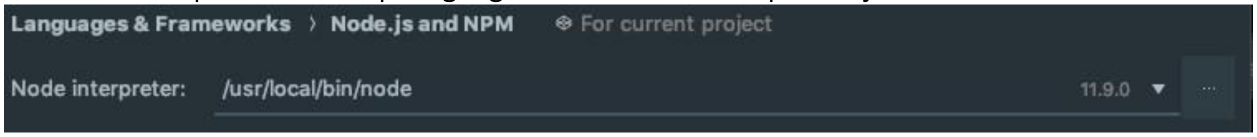
Overview

This Lab has four parts:

- **Exercise 1**: Warm-up exercise to practice basics JavaScript syntax.
- **Exercise 2:** Practice arrays and control structures.
- **Exercise 3:** Practice arrow functions and array functions.
- **Practice**: create a simple shopping application that allows you to practice the concepts from exercises 1 to 3.

Preparation

1. If not already done, install the latest LTS version of Node.js on your laptop from https://nodejs.org/en/
2. Configure WebStorm to use the installed Node.js:
    - File | Settings | Languages and Frameworks | Node.js and NPM for Windows
    - WebStorm | Preferences | Languages and Frameworks | Node.js and NPM for macOS



Languages & Frameworks 〉 Node.js and NPM ⊕ For current project

Node interpreter: /usr/local/bin/node 11.9.0

3. In the terminal type "`node -v`" you should see the version of the Node.js you have installed. The version should be 16+

Exercise 1 – Basic Syntax Warm up JS exercises

1. Add `Lab5-JS` folder to your repository.
2. Create a JavaScript file named **`excercise1.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. Consider the following array declaration: `let cars = ["Saab", "Volvo", "BMW"];`

- Add **"Toyota"** to the end of the array
- Add **"Mercedes"** to the beginning of the array.
- Create a ***displayArray*** function that takes an array as an argument and prints the array elements using a ***for-of*** loop. Call the ***displayArray*** function to display the cars array.
- Sort the array alphabetically and print it again.

Exercise 2 – Array Functions

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

Use the above array and implement and test the following functions:

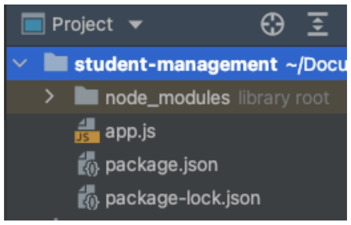
- **flatten:** gets a matrix (i.e., array of arrays) and returns a single dimensional flat array.
- **max**: gets an array and returns its maximum value (tip: use reduce function).
- **sort:** gets an array and returns a sorted array in descending order (from big to small).
- **square:** gets an array and returns an array with squared values.
- **average:** gets an array and returns its average.
- **removeDuplicates:** gets an array and returns an array without duplicate elements.
- **filter:** Find the sum of all the number in the array that are greater than 40. You should write everything as one 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 i prompt-sync ]`** in order to get the **`'prompt-sync'`** NPM package, that allows you to read user input from the terminal. This command will create multiple files inside your project. We will talk about those files in-depth when we discuss about Node.js.



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. Create an array named **students**
   Create a loop that asks the user to add 5 students. You should only ask the user the **name** and **gender** of the student. The age and grade of the students should be randomly generated. The age should be between [**17 - 35**] and the grade should be between [**0 and 100**] inclusive.

   Sample : the students array after adding the five students should look something like this

```
[
  { name: 'Hani', gender: 'Male', grade: 92, age: 19 },
  { name: 'Ahmed', gender: 'Male', grade: 81, age: 30 },
  { name: 'Sara', gender: 'Female', grade: 99, age: 24 },
  { name: 'Emany', gender: 'Female', grade: 50, age: 26 },
  { name: 'Issa', gender: 'Male', grade: 70, age: 24 }
]
```

7. The write the code that answers the following questions
   a. Find the student with the highest grade
   b. Find both the youngest and oldest student in the class
   c. Find the average and median student age in the class
   d. Calculate the mean and variance of the student grades.
   Note : Do not use the traditional loops to solve the above questions. You should use the arrow function.

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

Practice – Shopping App

In the following practice exercise, you will apply the concepts that you have seen above and create a simple shopping application with a cart that allows users to add item along with their price, change item quantities, delete items, and display a cart's total invoice.

Menu

What would you like to do?

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

Option 1 – Add Item
The user should be able to enter an item name along with a quantity and price to be added to their cart. Keep track of the newly introduced items by the end-user. If an item exists in the cart, you should just increment the quantity of that item, with the newly given quantity and price. Do not add duplicate item to the cart .

Option 2 – Change Quantity
A list of the items in the end-user's cart along with their quantities will be displayed. After selecting an item, the new quantity is input from the end-user and used to update the quantity of the corresponding item in the cart. Non-existing items are ignored.

Option 3 – Delete Item
A list of the items in the end-user's cart is displayed and the end-user can provide an item name to be removed from the cart. Non-existing items are ignored.

Option 4 – Display Invoice
The list of items in the end-user cart are displayed along with their quantities and prices. Also at the bottom the total price for all items in the cart is shown. Highlight the most expensive and the least expensive items in the cart with an asterisk and a double asterisk, respectively.