## JS and DOM functions

### Exercise 1: **Project Title:** Fitness Tracker - Simple Activity Tracking Application

The Fitness Tracker is a web-based application designed to help users easily track and organize their fitness activities. Built using HTML and JavaScript, this application provides a user-friendly interface for adding, viewing, and counting fitness activities efficiently.

**Requirements:**

1. **Activity Input:** Users are required to log their fitness activities by providing both an activity description and its corresponding duration in minutes. The application enforces the "required" attribute for the activity description field and ensures that the duration is a positive number. A button "**Add Activity**" facilitates the addition of each fitness activity.
2. **Activity Table:** All entered activities are neatly organized in a table format, presenting both the activity description and its corresponding duration in separate columns.
3. **Total Duration Display:** A dynamic display below the input section showcases the total duration of all logged fitness activities.

**Functionality**:

**addActivity()**: This function ensures the input fields are validated, requiring both the activity description and duration to be provided. Once validated, it adds a new row to the table, displaying the entered activity description and its duration.

### Exercise 2: **Title:** BMI Calculator Application

**Project Description:**

Create a BMI (Body Mass Index) calculator application which allows users to calculate their BMI based on their weight, height, and gender. BMI is a measure that indicates whether an individual has a healthy body weight for their height. The application should provide users with their BMI value, classification (e.g., underweight, normal weight, overweight, or obese), and a message indicating the healthy BMI range for their gender.

**﻿**

**Requirements:**

1. **Input fields:**

* The application should feature a form with id as **bmi-form** where users can input their gender, weight (in kg), and height (in cm).
* The gender field should have a dropdown menu with options for selecting male or female where values has to be **male** or **female**.
* The weight and height fields should have input fields with type as number where users can enter their weight and height, respectively.

2. **BMI Calculation:**

* Upon clicking the "Calculate" **button**, the **calculateBMI()** function has to be called and it should calculate the user's BMI using the provided weight and height and gender values.
* The BMI calculation formula, **bmi = weight / (height \* height)**.
* The calculated BMI should be rounded to two decimal places for accuracy.

3. **Color-Coded Display:**

* The application should visually represent the BMI classification by displaying the result in different colors.
* **bmi < 18.5** : Underweight - text color should be in blue
* **bmi >= 18.5** and **bmi < 24.9** : Normal weight - text color should be in green
* **bmi >= 25** and **bmi < 29.9** : Overweight - text color should be in orange
* **bmi >= 30 :** Obese - text color should be in red

4. **Gender-Specific Information:**

* The application should provide gender-specific information regarding the healthy BMI range for men and women.
* Depending on the selected gender, a message should be displayed.

5. **Validation:**

* If any of the input fields are empty, the application should display the user to fill out all fields before proceeding with the calculation as shown in the below output.

### Exercise 3: TIP CALCULATOR

**Progression:**

This HTML and JavaScript code creates a simple tip calculator web application. The user enters the bill amount and selects a tip percentage from a dropdown menu. Upon clicking the "Calculate Tip" button, the application computes the tip amount and total amount, displaying the results below the form.

**Requirements:**

1. The application should take input for the bill amount as a numeric value.
2. The user is required to choose a tip percentage from a predefined dropdown list, where percentage options are available in increments of 5, including values such as 5%, 10%, 15%, and 20%, with corresponding option values of 5, 10, 15, and 20, respectively.
3. Upon clicking the "Calculate Tip" button, the application should validate the inputs and display an alert if they are invalid.
4. If the inputs are valid, the application should calculate the tip amount and total amount based on the selected tip percentage.
5. The calculated tip amount and total amount should be displayed below the form.

**Formula**

The **calculateTip()** function retrieves the bill amount and tip percentage from the HTML inputs, validates them, and calculates the tip amount using **(billAmount \* tipPercentage) / 100**. The total amount is then computed by adding the bill amount and tip amount. Finally, the calculated tip amount and total amount are formatted to two decimal places.

## Jquery

### Exercise 1: Advanced jQuery Techniques

In this session, learn how to use jQuery to create dynamic interactions and animations.

**Question:**

Create an HTML page with several sections. Your task is to use jQuery to apply the following interactions:

1. The page should have an **H1** tag with the heading as "**Advanced jQuery Techniques**".
2. On clicking a button, hide all sections with a sliding effect using **slideUp()**. **Button id**: **btnHideSections**. Button text should be "**Hide Sections**".
3. Implement a feature where, when the mouse enters a section, the background **color** of the section changes to **lightblue** (rgb(173, 216, 230)). When the mouse leaves the section, revert the background color to its original state.
4. The initial background color of the sections should be white.
5. The sections must contain the following text:

* Section 1: "jQuery makes it easy to create rich, dynamic web pages with just a few lines of code."
* Section 2: "By using jQuery's extensive library of plugins, you can add advanced features like animations and AJAX with ease."
* Section 3: "Explore the world of jQuery and enhance your web development skills with interactive examples and tutorials."
* There must be 3 sections in total.

Write the jQuery code to achieve these interactions. You can assume that jQuery is already included in the HTML page.

Ensure that the jQuery code is appropriately written, and the interactions work seamlessly on the HTML page

**How to run the project:**

Open the terminal and give the below command to run the project

* **npm install** -> To install node modules
* **node server.js**-> To start the server

And click the **port** **8081**to view your webpage.

Click on run testcase to pass all the testcases.

**NOTE:**The testcase will be passed only if the port 8081 is running.

### Exercise 2: Traversing Ajax

In this session, learn about traversing and Ajax in jQuery.

**Question:**

1.   Create an HTML page with a button that has the ID "**btnLoadData**."

2.   Implement an Ajax call using jQuery's **load()** method to fetch data from a file named "**data.txt**." Once the data is successfully loaded, display an alert box with the message "**Data loaded successfully!**"

3.   Create an unordered list (UL) with some fruit names as list items (LI). For example:

* Apple
* Banana
* Orange
* Mango
* Kiwi

Write jQuery code to perform the following:

* On clicking a button with the ID "**btnFadeOddFruits**," fade out all the odd-indexed fruits (**Apple, Orange, Kiwi**) from the list.

Ensure that the Ajax call and traversal functionalities are appropriately implemented, and the desired results are achieved on the HTML page.

**How to run the project:**

Open the terminal and give the below command to run the project

* **npm install** -> To install node modules
* **node server.js**-> To start the server

And click the **port** **8081** to view your webpage.

Click on run testcase to pass all the testcases.

**NOTE:** The testcase will be passed only if the port 8081 is running.

ID's to be used:

Load button id = **btnLoadData**

Data container id = **dataContainer**

FruitList id = **fruitList**

Fade Button id = **btnFadeOddFruits**