**Asynchronous Methods Exercises**

**Exercise 1:** setTimeout

1. Create a function displayMessage that logs "Hello, after 3 seconds!" to the console.

2. Use setTimeout to call displayMessage after 3 seconds.

3. Modify the function to accept a message as a parameter. Call displayMessage("Hello, World!") with setTimeout after a 2-second delay.

**Exercise 2:** setInterval

1. Write a function startCountdown that logs a countdown from 5 to 1, decreasing each second.

2. Use setInterval to call startCountdown and log the countdown in the console.

3. Stop the countdown using clearInterval when it reaches 0.

**Exercise 3: Promises**

1. Write a function loadData that returns a new Promise that resolves with "Data loaded successfully" after 2 seconds.

2. Chain .then() to loadData to log the resolved message.

3. Add a .catch() block to handle any errors, logging "Data load failed" if an error occurs.

**Exercise 4:** Async/Await

1. Rewrite loadData from Exercise 3 using async/await.

2. Create an async function fetchAndLog that awaits the loadData function and logs the message.

3. Use a try...catch block to catch and log any errors.

**Exercise 5:** Promise.all

1. Create three promises:

• promise1 resolves after 1 second with "First promise resolved".

• promise2 resolves after 2 seconds with "Second promise resolved".

• promise3 resolves after 3 seconds with "Third promise resolved".

2. Use Promise.all to wait for all three promises to resolve, then log the results as an array.

**Exercise 6:** Promise.race

1. Use the three promises (promise1, promise2, and promise3) from Exercise 5.

2. Use Promise.race to find the first promise that resolves and log its value.

**DOM Selection Exercises**

**Exercise 7: Selecting Elements by ID and Class**

1. Create an HTML page with:

• An <h1> element with the id "header".

• A <div> element with the class "container".

• A <p> element inside the <div> with the class "text".

2. In JavaScript:

• Select the <h1> element by id and log it.

• Select the <div> element by class name and log it.

• Select the <p> element inside the container using querySelector and log it.

**Exercise 8: Selecting Elements with Query Selectors**

1. Add three <li> elements with the class "list-item" inside an <ul> element on your HTML page.

2. Select all elements with the class "list-item" using querySelectorAll and log the returned NodeList.

3. Iterate through the NodeList and log the text content of each <li> element.

**DOM Event Handling Exercises**

**Exercise 9: Handling Click Events**

1. Create a button with the text "Click me" in your HTML.

2. In JavaScript:

• Select the button element.

• Attach a click event listener to the button that logs "Button clicked!".

3. Create another button that logs "Another button clicked!" when pressed.

**Exercise 10: Preventing Default Form Behavior**

1. Create a simple form in your HTML with a text input and a submit button.

2. In JavaScript:

• Add a submit event listener to the form.

• Use event.preventDefault() in the event handler to prevent the form from submitting.

3. Log a message like "Form submission prevented" when the submit button is clicked.

**Exercise 11: Adding Keyboard Event Handlers**

1. Create an input field in your HTML.

2. In JavaScript:

• Select the input field and add a keydown event listener.

• Log the key that was pressed (use event.key) each time a key is pressed.

**Exercise 12: Mouseover Event and Changing Styles**

1. Create a <div> element with some text in your HTML.

2. In JavaScript:

• Add a mouseover event listener to the <div> that changes its background color when the mouse hovers over it.

• Add a mouseout event to change the color back when the mouse leaves the <div>.

These exercises will give you practical experience with JavaScript’s asynchronous features, DOM manipulation, and event handling