ASSESSMENT BRIEF

COURSE DETAILS	
COURSE NAME	WEB PROGRAMMING (100458-11001)
ASSESSMENT DETAILS	
TITLE/NAME	ASSIGNMENT: ADVANCED JAVASCRIPT CONCEPTS
WEIGHT	15%
DATE/DEADLINE	**WEEK 11 (14/05/2025) BEFORE 3:00PM**
DELIVERABLES	Your submission must be a zipped file (containing the HTML & JavaScript files) and uploaded to e-class.

Task 1: Scope and Closures - Online Store Discount

Objective: To understand variable scope (global, function, block) and closures by creating a discount system for an online store.

Steps:

1. Setup:
    - Create a new folder named discount_store.
    - Inside the folder, create two files: index.html and script.js.
2. HTML (index.html): Create the basic HTML structure including:
    - A heading "Online Store".
    - A paragraph displaying the "Current Price" with an element (e.g., span) to hold the price value, giving it an ID price. Start the current price at $100.
    - A button to "Apply Discount" with an ID applyDiscount.
    - A paragraph to display the "Discounted Price" with an element to hold the discounted price value, giving it an ID discountedPrice.
    - Link the external JavaScript file (script.js).
3. JavaScript (script.js):
    - Declare a global variable to represent the base price.
    - Create a function createDiscount(discountPercentage) that:
        - Accepts discountPercentage as an argument.
        - Returns another (inner) function.
        - This inner function should calculate the discounted price based on the discountPercentage and the base price.
        - The inner function should update the HTML to display the calculated discounted price.
    - Add an event listener to the "Apply Discount" button that, inside a block (e.g. an if statement), calls the createDiscount function with a pre-defined discount percentage (e.g. 20%). The result of createDiscount is a function that is used as the event handler.
4. Run: Open index.html in your browser. You should be able to click the "Apply Discount" button to see the discounted price.

Task 2: Asynchronous JavaScript - Fetching User Data

Objective: To practice asynchronous JavaScript using async/await to fetch user data from a mock API.

Steps:

1. Setup:
    - Create a new folder named user_data.
    - Inside the folder, create two files: index.html and script.js.
2. HTML (index.html): Create the basic HTML structure including:
    - A heading "User Data".
    - A button with the id "fetchUser" to trigger the data fetching.
    - A div with the id "userData" where user data will be displayed.
    - Link the external JavaScript file (script.js).
3. JavaScript (script.js):
    - Create an async function named fetchUserData(). Inside this function:
        - Use fetch to retrieve user data from a mock API endpoint (e.g., https://jsonplaceholder.typicode.com/users/1).
        - Check the response status. If the response is not "ok" (status code is not 200), throw an error.
        - Parse the JSON response to get the user data.
        - Call a displayUserData() function (defined in the next step), passing the retrieved user data to it.
        - Handle any errors that occur during the fetch process using a try...catch block, displaying an error message in the userData div.
    - Create a function displayUserData(user) that accepts a user object and dynamically generates HTML content inside the userData div to display the user's name, email, and city (accessed from the address.city property).
    - Add an event listener to the "fetchUser" button to trigger the fetchUserData() function when clicked.
4. Run: Open index.html in the browser and click the "Fetch User" button to display the user data.

Task 3: Object-Oriented Programming - Shapes

Objective: To implement OOP concepts (classes, inheritance, polymorphism, and encapsulation) by creating a system for different shapes.

Steps:

1. Setup:
    - Create a new folder named shapes.
    - Inside the folder, create two files: index.html and script.js.
2. HTML (index.html): Create the basic HTML structure including:
    - A heading "Shapes".
    - A div element with the ID "output" where the shape descriptions and calculated areas will be displayed.
    - Link the external JavaScript file (script.js).
3. JavaScript (script.js):
    - Create a base class called Shape with:
        - A private property #color.
        - A constructor to initialize the color.
        - A getter method getColor() to retrieve the color.
        - A method calculateArea() that *returns* 0 (since a generic shape has no defined area).
        - A toString() method to *return* a string that describes the shape and its color.
    - Create a class Circle that extends Shape. It should have:
        - A property radius.
        - A constructor that initializes the color (using super()) and the radius.
        - Override the calculateArea() method to return the circle's area.
        - Override the toString() method to return a string that describes the circle including its color and radius.
    - Create a class Square that extends Shape. It should have:
        - A property side.

        - A constructor that initializes the color (using super()) and the side length.
        - Override the calculateArea() method to return the square's area.
        - Override the toString() method to return a string that describes the square including its color and side length.
    - Instantiate a Circle and a Square with example values.
    - Using document.getElementById("output"), set the innerHTML of the output div to dynamically created HTML that displays the toString() result AND the calculateArea() result (formatted to two decimal places) for each shape in separate paragraphs.
4. Run: Open index.html in the browser. You should see the shape descriptions and areas displayed on the webpage.

END OF ASSIGNMENT QUESTIONS