JavaScript Exercises

Project Theme: "Local Community Event Portal"

Users can view upcoming events, register, filter events by category or location, and interact dynamically with the portal.

1. JavaScript Basics & Setup

Scenario: Set up your community portal to use JavaScript.
Objective: Configure environment and test basic script functionality.
Task:

- Use *<script src="main.js"></script>* in HTML
- Log "Welcome to the Community Portal" using console.log()
- Use an alert to notify when the page is fully loaded

2. Syntax, Data Types, and Operators

Scenario: Store event details like name, date, and available seats.
Objective: Use proper data types and operations.
Task:

- Use *const* for event name and date, let for seats
- Concatenate event info using template literals
- Use *++* or -- to manage seat count on registration

3. Conditionals, Loops, and Error Handling

Scenario: Only show valid events and limit registrations.
Objective: Apply conditions and handle invalid data.
User Story: As a user, I want only upcoming events with seats to be displayed.
Task:

- Use *if-else* to hide past or full events
- Loop through the event list and display using *forEach()*
- Wrap registration logic in *try-catch* to handle errors

4. Functions, Scope, Closures, Higher-Order Functions

Scenario: Create reusable functions for event operations.
Objective: Encapsulate logic and use closures.
Task:

- Create *addEvent(), registerUser(), filterEventsByCategory()*
- Use closure to track total registrations for a category
- Pass callbacks to filter functions for dynamic search

5. Objects and Prototypes

Scenario: Each event is an object with properties and methods.
Objective: Model real-world entities using objects.

Task:

- Define Event constructor or class
- Add *checkAvailability()* to prototype
- List object keys and values using *Object.entries()*

6. Arrays and Methods

Scenario: Manage an array of all community events.
Objective: Use array methods for CRUD operations.
Task:

- Add new events *using .push()*
- *Use .filter()* to show only music events
- *Use .map()* to format display cards (e.g., "Workshop on Baking")

7. DOM Manipulation

Scenario: Display all events dynamically on the webpage.
Objective: Render events using JS.
Task:

- Access DOM elements using *querySelector()*
- Create and append event cards using *createElement()*
- Update UI when user registers or cancels

8. Event Handling

Scenario: Add interactive elements like buttons and filters.
Objective: Respond to user actions.
Task:

- Use *onclick* for "Register" buttons
- Use *onchange* to filter events by category
- Use *keydown* to allow quick search by name

9. Async JS, Promises, Async/Await

Scenario: Fetch event data from a mock API.
Objective: Use asynchronous logic for remote operations.
Task:

- Fetch events from a mock JSON endpoint
- Use *.then()* and *.catch()* to handle results
- Rewrite using *async/await* and show loading spinner

10. Modern JavaScript Features

Scenario: Refactor code to be concise and maintainable.
Objective: Use ES6+ features.
Task:

- Use *let, const*, default parameters in functions
- Use destructuring to extract event details

- Use *spread* operator to clone event list before filtering

11. Working with Forms

Scenario: Create a registration form for event sign-up.
Objective: Connect form inputs to JavaScript.
Task:

- Capture name, email, and selected event using *form.elements*
- Prevent default form behavior using *event.preventDefault()*
- Validate inputs and show errors inline

12. AJAX & Fetch API

Scenario: Send user registration to the server.
Objective: Simulate backend communication.
Task:

- Use *fetch()* to POST user data to a mock API
- Show success/failure message after submission
- Use *setTimeout()* to simulate a delayed response

13. Debugging and Testing

Scenario: Registration is failing silently. You need to debug.
Objective: Use browser tools to inspect and fix issues.
Task:

- Use *Chrome Dev Tools* Console and *Network* tab
- Add breakpoints and inspect variables
- Log form submission steps and check fetch request payload

14. jQuery and JS Frameworks

Scenario: Use jQuery to simplify DOM tasks.
Objective: Understand and use jQuery.
Task:

- Use *$('#registerBtn').click(...)* to handle click events
- Use *.fadeIn()* and *.fadeOut()* for event cards
- Mention one benefit of moving to frameworks like React or Vue