HTML5 Exercises

Project Theme: Local Community Event Portal

A local city council wants a lightweight, browser-based portal to help residents register for events, check locations, and access basic services.

1. Create the HTML5 Base Template

Scenario: You're setting up the base document that every page on the portal will use.

Objective: Ensure semantic structure and compatibility across browsers.

Task:

- Use *<!DOCTYPE html>, <html lang="en">, <meta charset="UTF-8">*
- Add comments to label sections like "Navigation", "Main", "Footer"
- Save as *index.html* and open it in Chrome
- Inspect the document structure in Chrome Dev Tools

2. Navigation and Linking

Scenario: Users should navigate between "Home", "Events", and "Contact" sections.

Objective: Provide intuitive navigation and section-based references.

Task:

- Use *<nav>* with anchor tags *<a href="#events">Events</a>*
- Define matching IDs for each section like *<section id="events">*
- Add a link to an external help document using *<a href="help.html" target="_blank">*

3. Welcome Message with Styling and ID/Class

Scenario: Display a welcome banner styled uniquely for a logged-in user.

Objective: Practice block/inline tags and differentiate *id* and *class*

Task:

- Use *<div id="welcomeBanner">* and apply a blue background via internal CSS
- Use inline styles for a special offer *<span>* (e.g., color red, bold)
- Apply the *.highlight* class to certain elements for visual emphasis

4. **Image Gallery for Community Events**

   Scenario: Show images from past events in a table layout.

   Objective: Work with `<img>`, tables, and formatting tags.

   Task:

   - Use a *<table>* with 2 rows and 3 columns of *<img>* tags
   - Include *alt*, *title*, and style each image with borders using a class
   - Add a caption to describe each event

5. **Event Registration Form**

   Scenario: Residents need to register for events.

   Objective: Practice input types, validation, placeholder, autofocus, and output

   Task:

   - Include fields: name (text), email (email), date (date), event type (select), message (textarea)
   - Add *placeholder*, *required*, and *autofocus*
   - Display a confirmation message using *<output>* when the form is submitted
   - Style the form using CSS

6. **Event Feedback with Events Handling**

   Scenario: Collect real-time feedback and interactions from the user.

   Objective: Handle blur, change, click, double-click, and keyboard events.

   Task:

   - Use *onblur* to validate a phone number field
   - Use *onchange* on a dropdown to display the selected event fee
   - *onclick* on a submit button to show a confirmation
   - *ondblclick* on an image to enlarge it
   - Capture key events in the feedback textarea and count characters

7. **Video Invite with Media Events**

   Scenario: Show a short event promo video.

   Objective: Work with *<video>* and *oncanplay* event

Task:

- Insert a *<video>* element with source and controls
- Use *oncanplay* to display a message like "Video ready to play"
- Use *onbeforeunload* to warn users if they try to leave the form page unfinished

8. Saving User Preferences

Scenario: Store preferred event type for returning users.

Objective: Work with *localStorage*, *sessionStorage*, and deletion

Task:

- Save selected event type in *localStorage*
- On reload, retrieve and pre-select it
- Add a "Clear Preferences" button that clears both *localStorage* and *sessionStorage*

9. Geolocation for Event Mapping

Scenario: Locate the nearest event to the user.

Objective: Practice *geolocation.getCurrentPosition*, error handling, and options

Task:

- Create a button “Find Nearby Events”
- On click, use *getCurrentPosition* to get and display coordinates
- Handle permission denial and timeouts
- Use high accuracy options

10. Debugging with Chrome DevTools

Scenario: A few users report layout issues and script errors.

Objective: Use Chrome DevTools and VS Code features to debug.

Task:

- Use “Inspect Element” to modify styles and experiment live
- Use the Console tab to view logs from your *<script>*
- Add breakpoints in JS and reload the page to watch variable values