+91 96622 16697
jay@frameboxxers.com
F-201, Shilp Square-B, Nr. Shreeji Tower,
Drive-in Road, Vastrapur, Ahmedabad - 380015

Candidates should work their way through these levels. If they get stuck, they can ask for hints, but the goal is to complete as much as possible independently.

Level 1: Basic DOM Manipulation & Event Handling (Essential)

Goal: Allow users to add new tasks to the list.

Steps:

1. **Add Task Functionality:**
    - When the "Add Task" button (#addTaskBtn) is clicked, or the user presses Enter in the task input field (#taskInput):
        - Get the value from the #taskInput.
        - If the input is not empty, create a new <li> element.
        - The <li> should contain:
            - A with the task text.
            - A "Complete" button (e.g., with class complete-btn).
            - A "Delete" button (e.g., with class delete-btn).
        - Append the new <li> to the #taskList.
        - Clear the #taskInput field.
2. **Mark Task as Complete/Incomplete:**
    - When the "Complete" button of a task is clicked:
        - Toggle the completed class on the parent <li> element. This class should apply a strikethrough effect (already defined in style.css).
3. **Delete Task:**
    - When the "Delete" button of a task is clicked:
        - Remove the parent <li> element from the DOM.

Hints for Level 1:

- Use $("#selector").val() to get input value.
- Use $("#selector").append() or $("#selector").prepend() to add elements.
- Use $(this) inside event handlers to refer to the clicked element.
- Use .parent() to get the parent element.
- Use .remove() to delete an element.
- Use .toggleClass() to toggle a class.

+91 96622 16697
jay@frameboxxers.com
F-201, Shilp Square-B, Nr. Shreeji Tower,
Drive-in Road, Vastrapur, Ahmedabad - 380015

Level 2: Filtering & Clearing Completed Tasks (Intermediate)

Goal: Implement filters to show all, active, or completed tasks, and allow clearing completed tasks.

Steps:

1. **Filtering Tasks:**
    - When a filter button (#showAll, #showActive, #showCompleted) is clicked:
        - Remove the active class from all filter buttons and add it to the clicked button.
        - **"All" Filter:** Show all tasks.
        - **"Active" Filter:** Show only tasks that do *not* have the completed class. Hide tasks that have the completed class.
        - **"Completed" Filter:** Show only tasks that *do* have the completed class. Hide tasks that do *not* have the completed class.
2. **Clear Completed Tasks:**
    - When the "Clear Completed" button (#clearCompleted) is clicked:
        - Remove all <li> elements that have the completed class.

Hints for Level 2:

- Use $("#selector").show() and $("#selector").hide() to control visibility.
- Use .hasClass() to check for a class.
- Use a combination of .each() and conditional logic to iterate through tasks for filtering.

+91 96622 16697
jay@frameboxxers.com
F-201, Shilp Square-B, Nr. Shreeji Tower,
Drive-in Road, Vastrapur, Ahmedabad - 380015

Level 3: Persistence with Local Storage (Advanced)

Goal: Make the To-Do List persist even after the browser is closed and reopened.

Steps:

1. **Save Tasks:**
    - Whenever a task is added, marked complete/incomplete, or deleted, save the current state of the tasks to localStorage.
    - Store tasks as an array of objects, where each object has properties like text and completed (boolean).
2. **Load Tasks:**
    - When the page loads, check if there are saved tasks in localStorage.
    - If tasks exist, load them and dynamically add them to the #taskList using the same structure as in Level 1. Ensure their completed status is correctly applied.

Hints for Level 3:

- localStorage.setItem('key', value) to save data. Remember to JSON.stringify() your array of objects before saving.
- localStorage.getItem('key') to retrieve data. Remember to JSON.parse() the retrieved string.
- Perform the loading logic inside $(document).ready().

+91 96622 16697
jay@frameboxxers.com
F-201, Shilp Square-B, Nr. Shreeji Tower,
Drive-in Road, Vastrapur, Ahmedabad - 380015

Level 4: Enhancements and Refinements (Bonus/Excellent)

Goal: Add extra features for a more polished user experience.

Steps:

1. **Edit Task:**
    - Allow users to edit existing tasks by clicking on the task text.
    - When clicked, the should turn into an editable <input type="text"> pre-filled with the current task text.
    - When the user presses Enter or clicks outside the input field, update the task text and revert to .
2. **Drag and Drop Reordering (More Complex - Optional):**
    - Implement drag-and-drop functionality to allow users to reorder tasks within the list. (Requires jQuery UI Sortable or a custom implementation).
3. **Confirmation Dialog for Deletion:**
    - Before deleting a task, show a simple confirmation dialog (e.g., using confirm() or a custom modal) to prevent accidental deletions.

Hints for Level 4:

- For editing, you can replace the with an <input> element dynamically.
- Use blur event for input element to detect when it loses focus.
- For drag and drop, jQuery UI Sortable is the easiest path. If not using jQuery UI, it involves listening to dragstart, dragover, drop events and managing dataTransfer.

+91 96622 16697
jay@frameboxxers.com
F-201, Shilp Square-B, Nr. Shreeji Tower,
Drive-in Road, Vastrapur, Ahmedabad - 380015

Evaluation Criteria

Candidates will be evaluated on:

- **Correctness:** Does the functionality work as expected?
- **Code Quality:** Is the code clean, readable, and well-organized? Are variable names descriptive?
- **jQuery Best Practices:** Proper use of selectors, event delegation, and DOM manipulation.
- **Error Handling (Basic):** Does the application handle empty input for tasks gracefully?
- **Logical Flow:** Is the code structured logically?
- **Problem-Solving:** How do they approach and solve the different levels of complexity?
- **Commit History (Optional but Recommended):** If using Git, meaningful commit messages showing progress.
