**DOCUMENTATION: Mastering DOM Manipulation with a To-Do List in JavaScript**

**Author:** Ismail Munyentwari

**🧱 1. What Is DOM Manipulation?**

✅ **Definition:**  
DOM (Document Object Model) manipulation is the process of using JavaScript to **dynamically access, modify, and update elements** on a webpage. It allows your web page to respond to user actions, like clicks, typing, or checking boxes, in real-time.

**⚙️ 2. Why DOM Manipulation Matters in Real-World Apps**

* Modern web apps (like Trello, Todoist, Gmail, or Spotify) rely heavily on dynamic updates without refreshing the page.
* DOM manipulation allows **interactive features**: adding, editing, completing, and deleting tasks.
* It teaches **how to bridge user actions and UI updates** programmatically.

**🧠 3. Core Elements of a To-Do List App**

1. **Input Field**: Allows users to enter new tasks.
2. **Save Button**: Triggers adding the task to the list.
3. **Task List Container**: Where tasks appear dynamically.
4. **List Item Components**: Each task contains:
   * Checkbox → marks as done
   * Task Text → the content
   * Edit Button → allows editing the task
   * Delete Button → removes the task if completed

**Practice Tip:** Start by creating the HTML skeleton, then dynamically add tasks via JavaScript.

**📜 4. Step-by-Step: Adding a Task**

**Step 1: Select HTML Elements**



* container → where tasks will appear.
* btn → triggers task addition.
* inputValue → captures what the user types.

**Step 2: Listen for Click Event**



* Validates input before adding.
* Ensures only non-empty tasks are added.

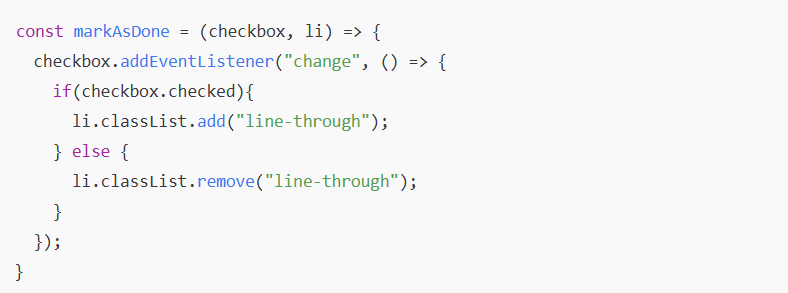
**Step 3: Create Task Elements Dynamically**

* document.createElement() → create <ul>, <li>, <input>, <span>
* Assign classes for styling: element.classList.add()
* Append elements: parent.appendChild(child)

**⚙️ 5. Step-by-Step: Marking Tasks Done**

**Goal:** Strike through tasks when checkbox is checked.

**Implementation:**

****

* checkbox.checked tracks the state.
* Updates li style in real-time.

**Practice Tip:** Experiment with different styles for completed tasks.

**🧩 6. Step-by-Step: Editing Tasks**

**Goal:** Update task text dynamically.

**Implementation:**

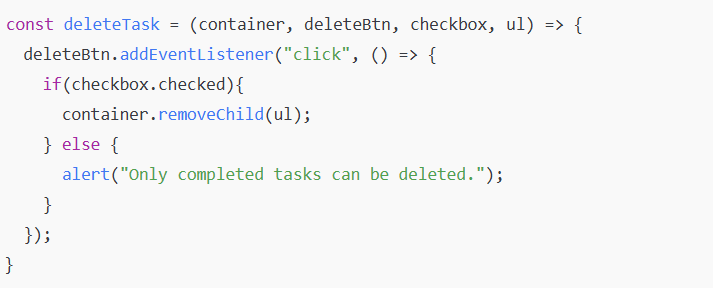


* Temporarily replaces <li> with <input> for editing.
* Press Enter to save changes and restore the list item.

**🗑 7. Step-by-Step: Deleting Tasks**

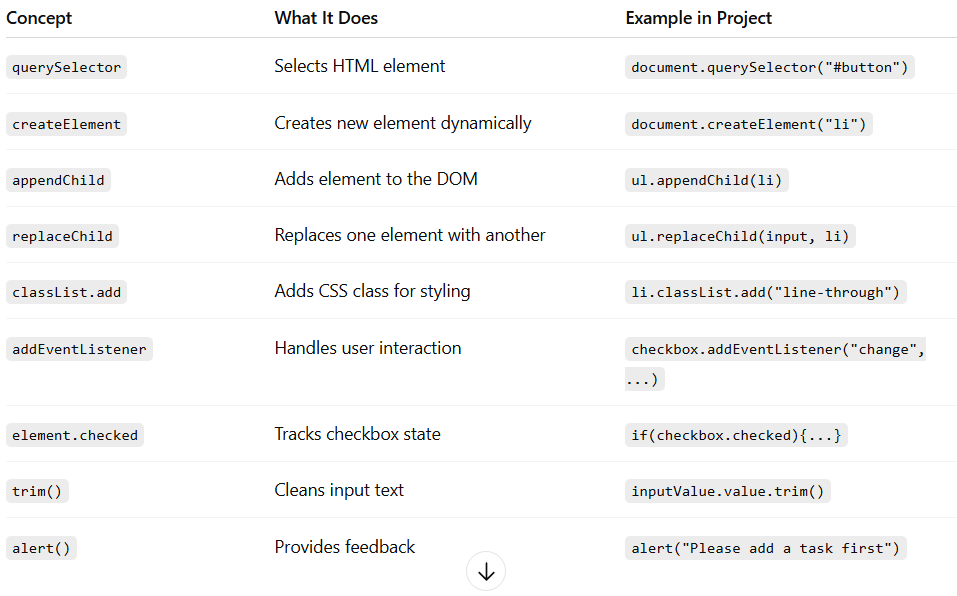
**Goal:** Remove tasks only if they are marked as done.

**Implementation:**



* Ensures users don’t accidentally delete unfinished tasks.
* Reinforces logic and conditional event handling.

**🧮 8. Summary of Core DOM Concepts Used**

****

**🧠 9. Behind the Scenes – What Happens When You Add a Task**

1. User types a task → input value is captured.
2. *Clicks Save* → validation checks if input is empty.
3. JavaScript creates elements*: list item*, *checkbox*, *edit/delete buttons*.
4. Elements are styled and appended to the container dynamically.
5. Event listeners are attached to:
   * *Checkbox* → marks task done
   * *Edit button* → edits task
   * *Delete button* → deletes task if done