

# JavaScript To-Do List Logic Breakdown

## STEP 1: Connecting JS to HTML

```
const todoInput = document.getElementById('todo-input');
```

What: Selects the input box.

Why: So JS can read what the user types.

```
const addTaskButton = document.getElementById('add-task-button');
```

Why: So JS can detect when user clicks 'Add'.

```
const todoList = document.getElementById('todo-list');
```

Why: This is where tasks will be displayed.

## STEP 2: Loading saved data (localStorage)

```
let tasks = JSON.parse(localStorage.getItem('tasks')) || [];
```

Break it down:

- localStorage.getItem('tasks') -> gets saved data (string)
- JSON.parse(...) -> converts string -> array
- || [] -> if nothing exists, start empty

Why: So tasks don't disappear after refresh.

## STEP 3: Rendering saved tasks on page load

```
tasks.forEach((task) => renderTasks(task));
```

What: Loop through each task object.

Why: Each task must be shown as a list item.

IMPORTANT IDEA: Data -> UI (Array -> visible list)

## STEP 4: Click to add a task

```
addTaskButton.addEventListener('click', () => {
```

Why: Nothing should happen until the user clicks.

```
const task1 = todoInput.value.trim();
```

What: Get text user typed.

Why: Tasks come from user input.

```
if (!task1) return;
```

Why: Prevent empty tasks.

## STEP 5: Create task object

```
const newTask = {  
  id: Date.now(),  
  text: task1,
```

# JavaScript To-Do List Logic Breakdown

```
    completed: false,  
};
```

Why object? Because each task has identity, text, and completion state.

## STEP 6: Save & show new task

```
tasks.push(newTask);  
saveTasks();  
renderTasks(newTask);
```

WHY THIS ORDER MATTERS:

1. Save data
2. Show UI

This keeps data and UI in sync.

## STEP 7: renderTasks (MOST IMPORTANT FUNCTION)

```
function renderTasks(task) {
```

This function means: 'Take ONE task object and show it on screen'

```
const li = document.createElement('li');
```

Why: We need an <li> for each task.

```
li.innerHTML = `  
  <span>${task.text}</span>  
  <button>delete</button>  
`;
```

Why span? To style or target task text separately.

```
li.addEventListener('click', ...)
```

Why: Clicking task toggles completed state.

```
task.completed = !task.completed;
```

Why: Change data.

```
li.classList.toggle('Completed');
```

Why: Change UI.

-> This is data <-> UI synchronization

## STEP 8: Delete logic

```
tasks = tasks.filter((t) => t.id !== task.id);
```

Why: Remove task from array.

```
li.remove();  
saveTasks();
```

Why: Remove from screen + save change.

# JavaScript To-Do List Logic Breakdown

## BIG PICTURE (THIS IS GOLD)

Your todo app is doing only 3 things:

1. Store data
2. Show data
3. Sync data with UI

Everything else is just tools.