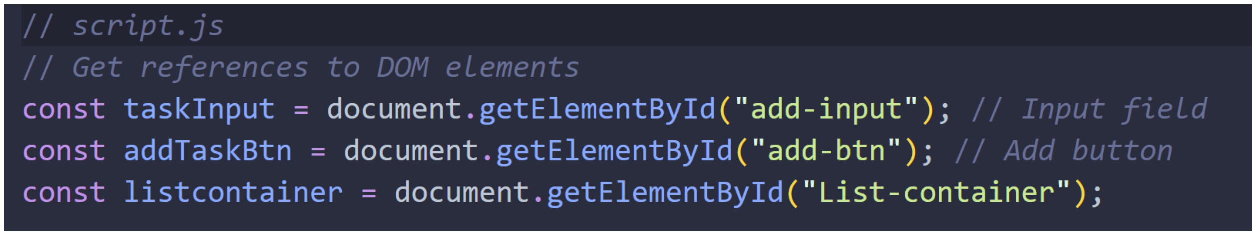
This document explains how the JavaScript code in the To-Do app manipulates the DOM and handles events to create an interactive task management application.

DOM Manipulation

The code interacts with the DOM in several ways:

1. **Element Selection**:

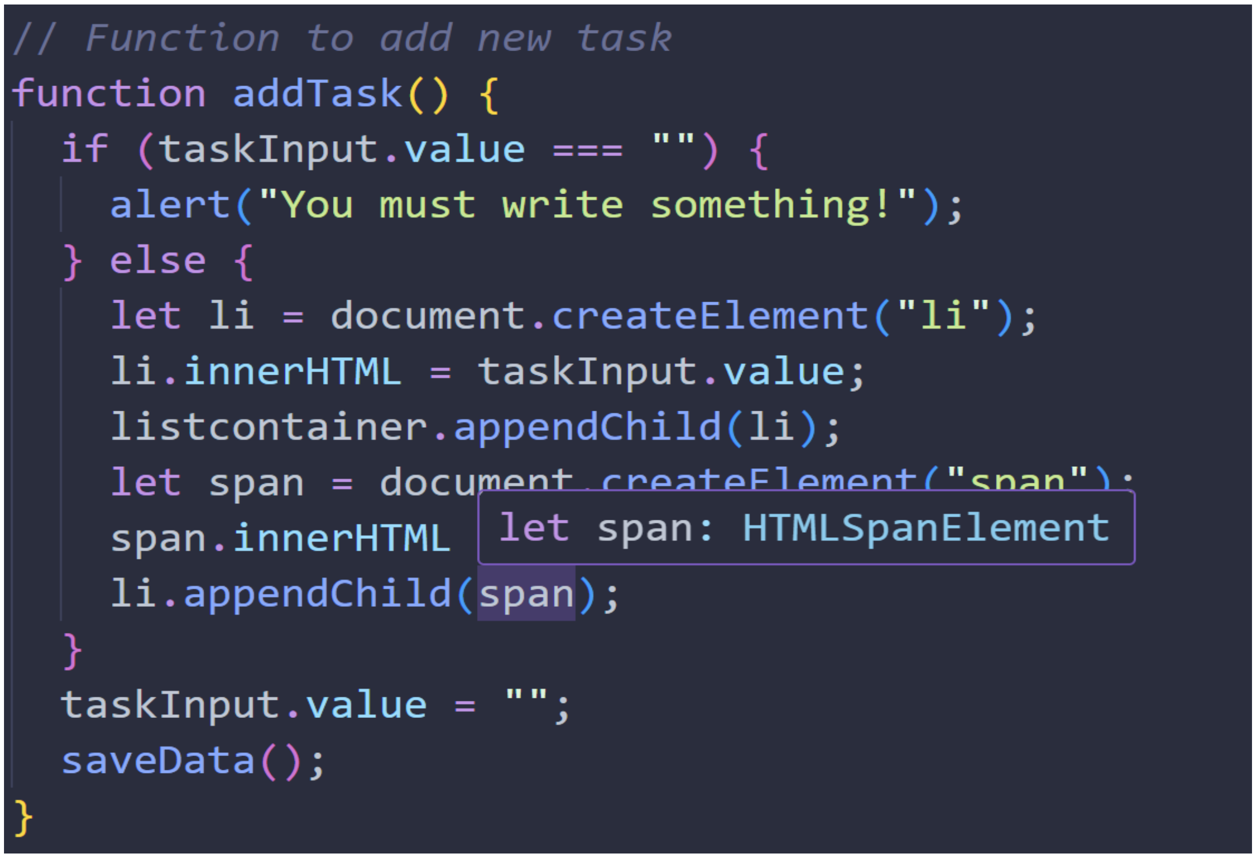
Javascript



```javascript
// script.js
// Get references to DOM elements
const taskInput = document.getElementById("add-input"); // Input field
const addTaskBtn = document.getElementById("add-btn"); // Add button
const listcontainer = document.getElementById("List-container");
```

2. **Creating New Elements**:
   In the addTask() function:

javascript

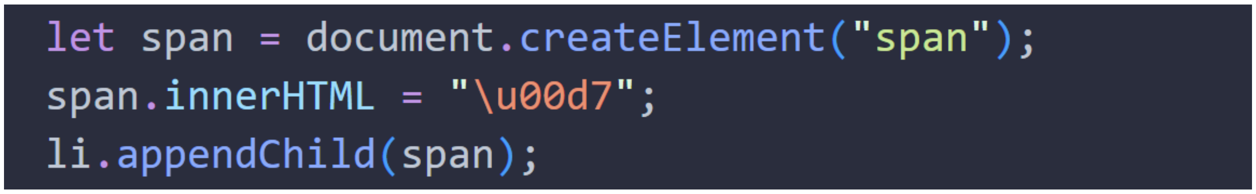


```javascript
// Function to add new task
function addTask() {
  if (taskInput.value === "") {
    alert("You must write something!");
  } else {
    let li = document.createElement("li");
    li.innerHTML = taskInput.value;
    listcontainer.appendChild(li);
    let span = document.createElement("span");
    span.innerHTML
    li.appendChild(span);
  }
  taskInput.value = "";
  saveData();
}
```

This creates a new list item (<li>) for each task and adds it to the list container.

3. **Adding Delete Button**:

javascript



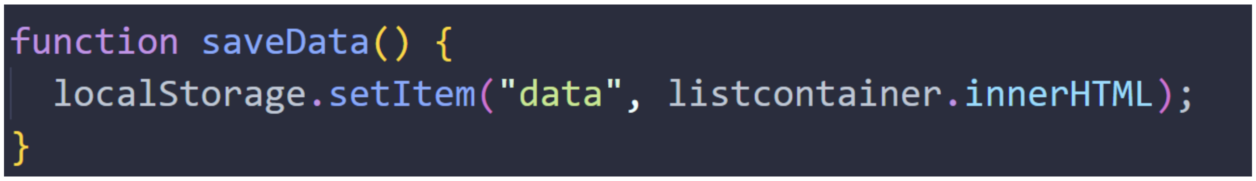
```javascript
let span = document.createElement("span");
span.innerHTML = "\u00d7";
li.appendChild(span);
```

Each task gets a delete button (represented by the × symbol) appended to it.

4. **Toggling Task Completion**:
   The code toggles the checked class on list items, which changes their appearance (adding strikethrough and changing the checkbox icon) through CSS.

5. **Persisting Data**:

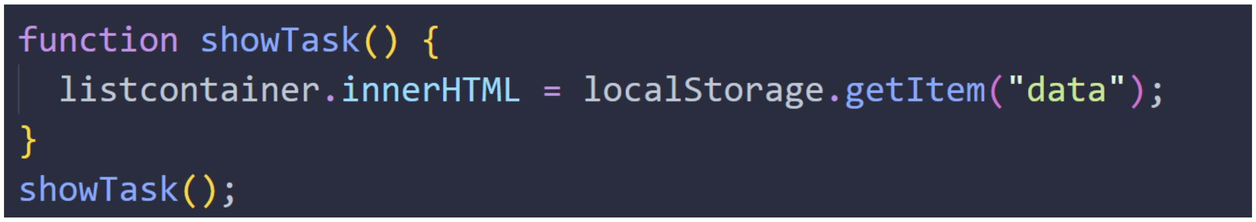
javascript



```javascript
function saveData() {
  localStorage.setItem("data", listcontainer.innerHTML);
}
```

The entire list HTML is saved to localStorage, and retrieved when the page loads:

javascript



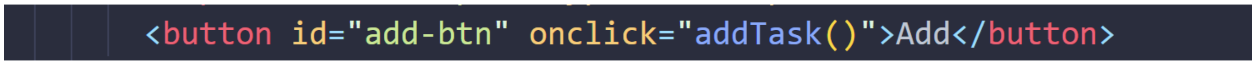
```javascript
function showTask() {
  listcontainer.innerHTML = localStorage.getItem("data");
}
showTask();
```

Event Handling

The code handles several user interactions:

1. **Add Button Click**:

html



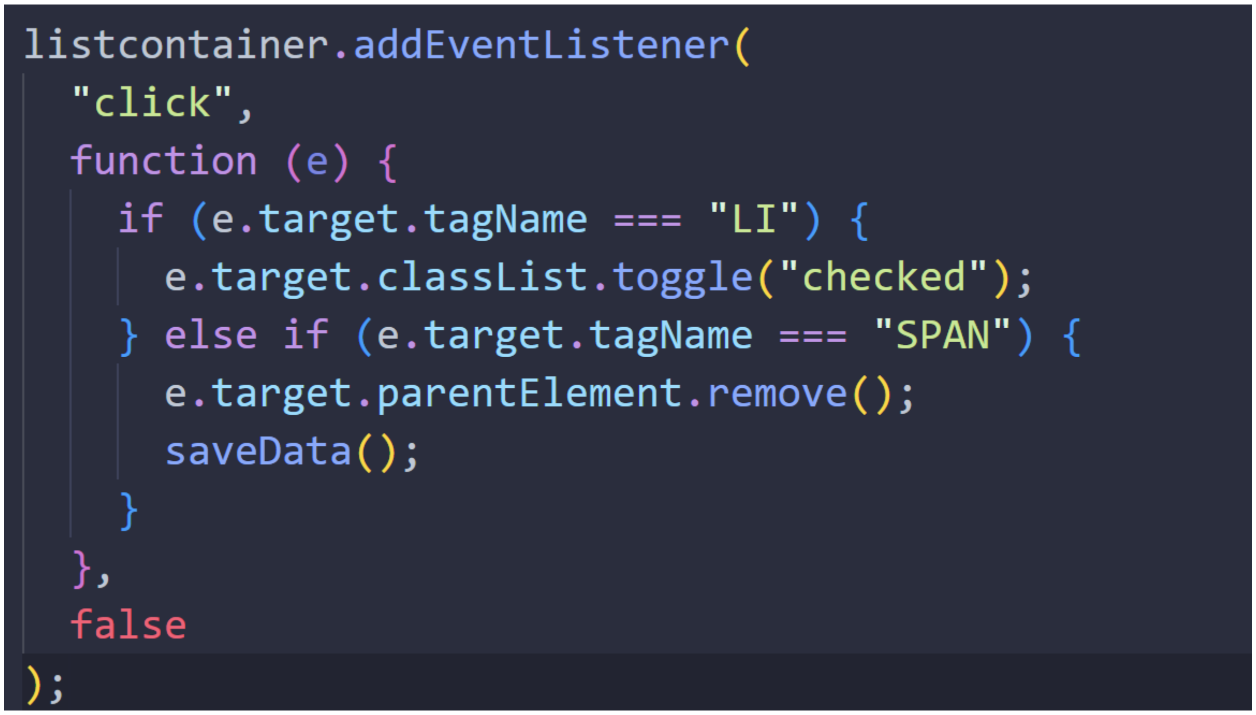
```html
            <button id="add-btn" onclick="addTask()">Add</button>
```

Run HTML

The onclick attribute directly calls the addTask() function when the button is clicked.

2. **Task List Interactions**:

javascript



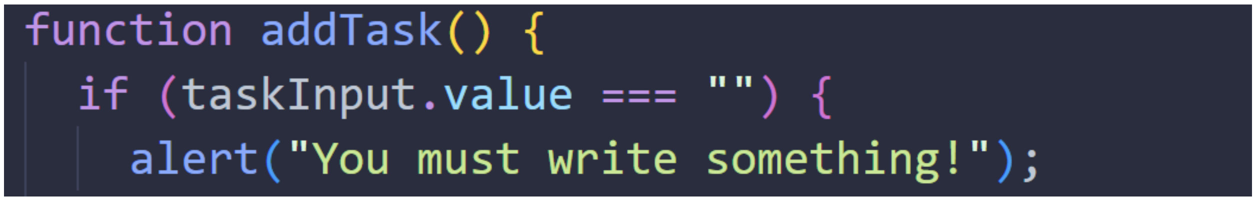
```javascript
listcontainer.addEventListener(
  "click",
  function (e) {
    if (e.target.tagName === "LI") {
      e.target.classList.toggle("checked");
    } else if (e.target.tagName === "SPAN") {
      e.target.parentElement.remove();
      saveData();
    }
  },
  false
);
```

This event listener handles two types of clicks within the task list:

- Clicking on a task (LI element) toggles its completed state
- Clicking on the delete button (SPAN element) removes the task

3. **Input Validation**:

javascript



```javascript
function addTask() {
  if (taskInput.value === "") {
    alert("You must write something!");
```

Prevents adding empty tasks.

Visual Feedback

The CSS provides visual feedback for user interactions:

- Hover effects on the Add button

- Different styles for completed tasks (strikethrough text and different checkbox icon)
- Hover effect on delete buttons

The combination of these DOM manipulations and event handlers creates a fully functional To-Do application that persists data between sessions