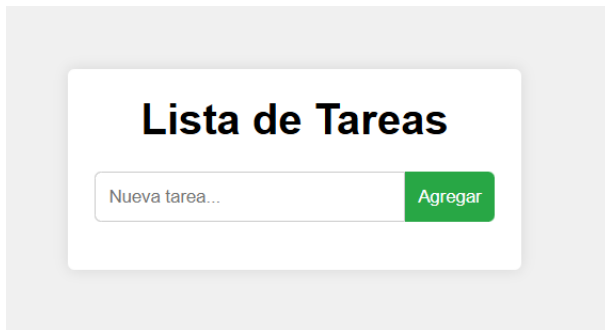


## LISTA DE TAREAS



Lista de Tareas

Nueva tarea... Agregar

Index.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Lista de Tareas</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <h1>Lista de Tareas</h1>
    <form id="task-form">
      <input type="text" id="task-input" placeholder="Nueva tarea...">
      <button type="submit">Agregar</button>
    </form>
    <ul id="task-list"></ul>
  </div>
  <script src="script2.js"></script>
</body>
</html>
```

## Styles.css

```
/* styles.css */
body {
  font-family: Arial, sans-serif;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  margin: 0;
  background-color: #f0f0f0;
}

.container {
  background: white;
  padding: 20px;
  border-radius: 5px;
  box-shadow: 0 0 10px rgba(0,0,0,0.1);
  width: 300px;
}

h1 {
  margin-top: 0;
  text-align: center;
}

form {
  display: flex;
}

#task-input {
  flex: 1;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px 0 0 5px;
}

button {
  padding: 10px;
  border: none;
  background-color: #28a745;
  color: white;
  border-radius: 0 5px 5px 0;
  cursor: pointer;
}
```

```

}

button:hover {
  background-color: #218838;
}

ul {
  list-style: none;
  padding: 0;
}

li {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 10px;
  border-bottom: 1px solid #ccc;
}

li.completed {
  text-decoration: line-through;
  color: #aaa;
}

.delete-btn {
  background: none;
  border: none;
  color: red;
  cursor: pointer;
}

```

Script.js

```

document.addEventListener('DOMContentLoaded', () => {
  const taskForm = document.getElementById('task-form');
  const taskInput = document.getElementById('task-input');
  const taskList = document.getElementById('task-list');

```

```
// Array para almacenar las tareas
let tasks = [];

// Escuchar el evento submit del formulario
taskForm.addEventListener('submit', (e) => {
  e.preventDefault();
  const taskText = taskInput.value.trim();
  if (taskText) {
    addTask(taskText);
    taskInput.value = '';
  }
});

// Escuchar eventos de clic en la lista de tareas
taskList.addEventListener('click', (e) => {
  const taskId = e.target.dataset.id;
  if (e.target.classList.contains('delete-btn')) {
    removeTask(taskId);
  } else if (e.target.tagName === 'LI') {
    toggleComplete(taskId);
  }
});

// Función para agregar una tarea
function addTask(taskText) {
  const newTask = {
    id: Date.now().toString(),
    text: taskText,
    completed: false
  };
  tasks.push(newTask);
  renderTasks();
}

// Función para eliminar una tarea
function removeTask(id) {
  tasks = tasks.filter(task => task.id !== id);
  renderTasks();
}

// Función para alternar el estado completado de una tarea
function toggleComplete(id) {
  tasks = tasks.map(task =>
    task.id === id ? { ...task, completed: !task.completed } : task
  );
}
```

```

        renderTasks();
    }

    // Función para renderizar la lista de tareas
    function renderTasks() {
        taskList.innerHTML = '';
        tasks.forEach(task => {
            const li = document.createElement('li');
            li.textContent = task.text;
            li.dataset.id = task.id;
            if (task.completed) {
                li.classList.add('completed');
            }
            const deleteBtn = document.createElement('button');
            deleteBtn.textContent = 'Eliminar';
            deleteBtn.classList.add('delete-btn');
            deleteBtn.dataset.id = task.id;
            li.appendChild(deleteBtn);
            taskList.appendChild(li);
        });
    }
});

```

- **DOMContentLoaded:** Este evento se dispara cuando el documento HTML ha sido completamente cargado y parseado, sin esperar a que se carguen hojas de estilo, imágenes y subtramas. Esto asegura que el código JavaScript se ejecute después de que el DOM esté completamente disponible.
- **taskForm:** Selecciona el formulario de tareas.
- **taskInput:** Selecciona el campo de entrada de texto para las tareas.
- **taskList:** Selecciona el elemento <ul> donde se mostrarán las tareas.
- **submit:** Se escucha el evento submit del formulario.
- **e.preventDefault():** Previene la acción predeterminada del formulario (recargar la página).
- **taskInput.value.trim():** Obtiene y limpia el texto de entrada.
- **addTask(taskText):** Llama a la función addTask si el texto de la tarea no está vacío.
- **taskInput.value = '':** Limpia el campo de entrada después de agregar la tarea.
- **click:** Se escucha el evento click en el elemento <ul>.
- **e.target.dataset.id:** Obtiene el id de la tarea desde el elemento clicado.
- **e.target.classList.contains('delete-btn'):** Verifica si el elemento clicado es el botón de eliminar.
- **removeTask(taskId):** Llama a la función removeTask si se clicó en el botón de eliminar.
- **e.target.tagName === 'LI':** Verifica si el elemento clicado es una lista (<li>).
- **toggleComplete(taskId):** Llama a la función toggleComplete si se clicó en un elemento de la lista.

- click: Se escucha el evento click en el elemento <ul>.
- e.target.dataset.id: Obtiene el id de la tarea desde el elemento clicado.
- e.target.classList.contains('delete-btn'): Verifica si el elemento clicado es el botón de eliminar.
- removeTask(taskId): Llama a la función removeTask si se clicó en el botón de eliminar.
- e.target.tagName === 'LI': Verifica si el elemento clicado es una lista (<li>).
- toggleComplete(taskId): Llama a la función toggleComplete si se clicó en un elemento de la lista.
- removeTask(id): Función para eliminar una tarea.
- tasks.filter(task => task.id !== id): Filtra el array tasks para eliminar la tarea con el id proporcionado.
- renderTasks(): Llama a renderTasks para actualizar la lista de tareas en el DOM.
- toggleComplete(id): Función para alternar el estado completed de una tarea.
- tasks.map(task => ...): Mapea el array tasks para crear un nuevo array con el estado completed alternado para la tarea con el id proporcionado.
- task.id === id ? { ...task, completed: !task.completed } : task: Utiliza el operador ternario para alternar el estado completed de la tarea correspondiente.
- renderTasks(): Llama a renderTasks para actualizar la lista de tareas en el DOM.
- renderTasks(): Función para renderizar (mostrar) las tareas en el DOM.
- taskList.innerHTML = '': Limpia el contenido actual de la lista de tareas.
- tasks.forEach(task => {...}): Itera sobre el array tasks y crea elementos de lista (<li>) para cada tarea.
- document.createElement('li'): Crea un nuevo elemento <li>.
- li.textContent = task.text: Establece el texto del elemento <li> como el texto de la tarea.
- li.dataset.id = task.id: Asigna el id de la tarea al atributo data-id del elemento <li>.
- li.classList.add('completed'): Si la tarea está completada, agrega la clase completed al elemento <li>.
- document.createElement('button'): Crea un nuevo botón.
- deleteBtn.textContent = 'Eliminar': Establece el texto del botón como "Eliminar".
- deleteBtn.classList.add('delete-btn'): Agrega la clase delete-btn al botón.
- deleteBtn.dataset.id = task.id: Asigna el id de la tarea al atributo data-id del botón.
- li.appendChild(deleteBtn): Añade el botón al elemento <li>.
- taskList.appendChild(li): Añade el elemento <li> a la lista de tareas (<ul>).
-