What I have added

HTML:

```
<div class="inbox">

    <div class="add-task">
      <input type="text" id="newTask" placeholder="Add a new task" />
      <button onclick="addTask()">Add Task</button>
    </div>

    <!-- Existing task items -->
    <div class="item">
      <input type="checkbox">
      <p>Task 1</p>

      <span class="delete-task"
onclick="deleteTask(this.parentNode)">Delete</span>
    </div>

    <div class="item">
      <input type="checkbox">
      <p>Task 2</p>
      <span class="delete-task"
onclick="deleteTask(this.parentNode)">Delete</span>
    </div>

    <div class="item">
      <input type="checkbox">
      <p>Task 3</p>
      <span class="delete-task"
onclick="deleteTask(this.parentNode)">Delete</span>
    </div>

  </div>
```

CSS:

```
.add-task {
      margin: 20px;
    border-bottom: 2px solid blue;
    padding: 1rem;
    }

    .delete-task {
      margin: 20px;
      cursor: pointer;
```

```
      color: red;
    }
```

Script:

```
let lastChecked;
const checkboxes = document.querySelectorAll('.inbox input[type="checkbox"]');

function handleCheck(e) {
  let inBetween = false;
  const checkboxes = document.querySelectorAll('.inbox
input[type="checkbox"]');

  if (e.shiftKey && this.checked) {
    checkboxes.forEach((checkbox) => {
      if (checkbox === this || checkbox === lastChecked) {
        inBetween = !inBetween;
        console.log("These are checked in between");
      }

      if (inBetween) {
        checkbox.checked = true;
      }
    });
  }

  lastChecked = this;
}

function bindCheckboxListeners() {
  const checkboxes = document.querySelectorAll('.inbox
input[type="checkbox"]');
  checkboxes.forEach(checkbox => checkbox.removeEventListener('click',
handleCheck));
  checkboxes.forEach(checkbox => checkbox.addEventListener('click',
handleCheck));
}

const addTaskInput = document.getElementById('newTask');

function addTask() {
  const newTaskText = addTaskInput.value.trim();
  if (newTaskText !== '') {
    const newItem = document.createElement('div');
    newItem.classList.add('item');

    const checkbox = document.createElement('input');
    checkbox.type = 'checkbox';
```

```
    const paragraph = document.createElement('p');
    paragraph.textContent = newTaskText;

    const deleteButton = document.createElement('span');
    deleteButton.classList.add('delete-task');
    deleteButton.textContent = 'Delete';
    deleteButton.onclick = function () {
      deleteTask(newItem);
      bindCheckboxListeners(); // Update event listeners after deleting a task
    };

    newItem.appendChild(checkbox);
    newItem.appendChild(paragraph);
    newItem.appendChild(deleteButton);

    document.querySelector('.inbox').appendChild(newItem);

    addTaskInput.value = ''; // Clear the input field after adding a task
    bindCheckboxListeners(); // Update event listeners after adding a task
  }
}

function deleteTask(taskItem) {
  taskItem.parentNode.removeChild(taskItem);
  bindCheckboxListeners(); // Call it after deleting a task
}

bindCheckboxListeners();
```

As an extra feature, I have changed the HTML structure and added new functionalities to the page. Notably, a task addition section has been introduced at the top of the .inbox div. This section includes an input field and a button, enabling users to add new tasks dynamically. Each new task is represented by a div containing a checkbox, a paragraph for the task description, and a delete button, enhancing the interactivity and usability of the application.

In the CSS, adjustments were made to accommodate these new elements. A styling class .add-task was added, giving a distinct style to the task addition section, including margins and a bottom border. The .delete-task class was also introduced, giving the delete buttons a margin, a pointer cursor on hover, and a red color for visual emphasis, thereby improving the user interface.

On the JavaScript side, two main functions, addTask and deleteTask, were incorporated. The addTask function creates and appends new task elements to the .inbox div based on the user's input. It creates a new div with a checkbox, a paragraph for the task description, and a delete button, mirroring the structure of existing tasks. The deleteTask function, on the other hand, provides functionality to remove tasks from the list. It is triggered by the delete button next to each task, allowing users to efficiently manage their task list.

These enhancements significantly augment the functionality of the webpage, transforming it from a simple checkbox interaction demo to a more complex and interactive task management tool. The ability to add and remove tasks dynamically offers a practical utility to the user, while maintaining the original checkbox interaction feature.