functional component in React:

Sure! Let's create a detailed example of a functional component in React. We'll build a simple to-do list application where users can add new tasks, mark tasks as completed, and delete tasks.

```jsx
import React, { useState } from 'react';

const TodoList = () => {
  // State for managing tasks
  const [tasks, setTasks] = useState([]);
  // State for managing input field
  const [inputValue, setInputValue] = useState('');

  // Function to handle adding a new task
  const addTask = () => {
    if (inputValue.trim() !== '') {
      setTasks([...tasks, { id: Date.now(), text: inputValue, completed: false }]);
      setInputValue('');
    }
  };

  // Function to handle toggling the completion status of a task
  const toggleTaskCompletion = (taskId) => {
    const updatedTasks = tasks.map(task =>
      task.id === taskId ? { ...task, completed: !task.completed } : task
    );
    setTasks(updatedTasks);
  };

  // Function to handle deleting a task
  const deleteTask = (taskId) => {
    const updatedTasks = tasks.filter(task => task.id !== taskId);
    setTasks(updatedTasks);
  };

  return (
    <div>
      <h2>Todo List</h2>
      {/* Input field for adding new tasks */}
      <div>
        <input
          type="text"
```

```
          placeholder="Enter task"
          value={inputValue}
          onChange={(e) => setInputValue(e.target.value)}
        />
        <button onClick={addTask}>Add Task</button>
      </div>
      {/* Displaying tasks */}
      <ul>
        {tasks.map(task => (
          <li key={task.id}>
            <span
              style={{ textDecoration: task.completed ? 'line-through' : 'none' }}
              onClick={() => toggleTaskCompletion(task.id)}
            >
              {task.text}
            </span>
            <button onClick={() => deleteTask(task.id)}>Delete</button>
          </li>
        ))}
      </ul>
    </div>
  );
};

export default TodoList;
```

In this example:

- We use the `useState` hook to manage state. `tasks` state stores an array of tasks, and `inputValue` state stores the value of the input field for adding new tasks.
- `addTask` function is responsible for adding a new task to the `tasks` state array. It first checks if the input value is not empty and then adds the task to the array.
- `toggleTaskCompletion` function toggles the completion status of a task. It creates a new array of tasks with the completion status of the clicked task toggled.
- `deleteTask` function deletes a task from the array of tasks.
- The component renders an input field for adding new tasks, a list of tasks with checkboxes for marking them as completed, and delete buttons for removing them.
- `map` function is used to iterate over the `tasks` array and render each task as a list item (`<li>`).

This example demonstrates how to build a simple to-do list application using functional components in React.