React Fundamentals & Todo List Example

1. Components

Components are the building blocks of a React application. They're reusable pieces of code that represent UI elements.

- Functional Components: Simple, stateless components that take props and return JSX.
- Class Components: More complex components that can have state and lifecycle methods.

```
// Functional component
function Button() {
  return <button>Click me!</button>;
}

// Class component
class Button extends React.Component {
  render() {
    return <button>Click me!</button>;
  }
}
```

2. Props

Props are short for "properties" and are used to pass data from a parent component to a child component.

```
function Button(props) {
  return <button>{props.label}</button>;
}

function App() {
  return <Button label="Click me!" />;
}
```

3. State

State is used to store data that changes over time. It's like a variable that can be updated dynamically.

```
import { useState } from 'react';
```

```
function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}
```

4. Events

Events are used to handle user interactions, such as clicks or form submissions.

```
function Button() {
  const handleClick = () => {
    alert('Button clicked!');
  };

  return <button onClick={handleClick}>Click me!</button>;
}
```

5. Conditional Rendering

Conditional rendering is used to show or hide elements based on certain conditions.

```
function DataFetcher() {
  const [loading, setLoading] = useState(true);

  if (loading) {
    return <p>Loading...</p>;
  }

  return <p>Data fetched!</p>;
}
```

6. Lists & Keys

Lists are used to render arrays of data. Keys are used to help React keep track of the elements in the list.

React Fundamentals & Todo List Example

```
function ItemList() {
  const items = ['Item 1', 'Item 2', 'Item 3'];

  return (
    <ul>
      {items.map((item, index) => (
        <li key={index}>{item}</li>
      ))}
    </ul>
  );
}
```

Important Files in a React Project

- index.js: The entry point of the application.

- App.js: The main component of the application.

- components/: A folder for storing reusable components.

- index.html: The HTML file that renders the React application.

Least Important Files

- reportWebVitals.js: A file that measures the performance of the application.

- setupTests.js: A file that sets up testing for the application.

Todo List App Example

Let's try to create a simple React application that uses these concepts. We'll create a to-do list application that allows users to add, remove, and mark tasks as completed.

```
import { useState } from 'react';

function TodoList() {
  const [tasks, setTasks] = useState([]);
  const [newTask, setNewTask] = useState('');

  const handleAddTask = () => {
    setTasks([...tasks, { text: newTask, completed: false }]);
    setNewTask('');
  };

  const handleRemoveTask = (index) => {
    setTasks(tasks.filter((task, i) => i !== index));
  };

  const handleToggleCompleted = (index) => {
    setTasks(
      tasks.map((task, i) =>
        i === index ? { ...task, completed: !task.completed } : task
      )
    );
  };

  return (
    <div>
      <input
        type="text"
        value={newTask}
        onChange={(e) => setNewTask(e.target.value)}
      />
      <button onClick={handleAddTask}>Add Task</button>
      <ul>
        {tasks.map((task, index) => (
          <li key={index}>
            <span style={{ textDecoration: task.completed ? 'line-through' : 'none' }}>
              {task.text}
            </span>
            <button onClick={() => handleRemoveTask(index)}>Remove</button>
            <button onClick={() => handleToggleCompleted(index)}>
              {task.completed ? 'Unmark' : 'Mark as Completed'}
            </button>
          </li>
        ))}
      </ul>
    </div>
  );
}

export default TodoList;
```