

Lesson 05 Demo 02

Create a React Application Using the Principles of Redux

Objective: To develop a react application that demonstrates the principle of Redux

Tools Required: Node Terminal, React app, and Visual Studio Code

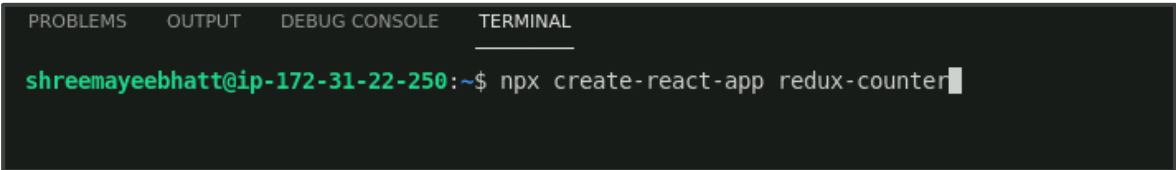
Prerequisites: Knowledge of creating a React app and understanding of the folder structure

Steps to be followed:

1. Create a new **React** app
2. Create a new file as **store.js**
3. Open the existing file called **App.js**
4. Wrap the **App** component with the **Provider** component
5. Run the application

Step 1: Create a new React app

- 1.1 Open the terminal and run the command **npx create-react-app redux-counter**



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
shreemayeebhatt@ip-172-31-22-250:~$ npx create-react-app redux-counter
```

Note: This command will create a new React app with the name **redux-counter**

- 1.2 Move to the **redux-counter** directory by running the **cd redux-counter** command

```
Happy hacking!  
shreemayeebhatt@ip-172-31-22-250:~$ cd redux-counter/
```

- 1.3 Install the necessary dependencies by running the **npm install redux react-redux** command

```
shreemayeebhatt@ip-172-31-22-250:~/redux-counter$ npm install redux react-redux
```

Step 2: Create a new file called store.js

- 2.1 create the **store.js** file in the **src** directory

- 2.2 Import the **configureStore** function from the **redux** package

```
1 import { createStore } from 'redux';  
2
```

- 2.3 Define the **initial state** of the store

- 2.4 Create a **reducer** function that handles **state** updates based on different action types

- 2.5 Use the **createStore** function to create the **Redux store** with the **reducer** function and **initial state**

2.6 Export the default store

```
1  import { createStore } from 'redux';
2
3  const initialState = {
4    count: 0,
5  };
6
7  function reducer(state = initialState, action) {
8    switch (action.type) {
9      case 'INCREMENT':
10     return { count: state.count + 1 };
11     case 'DECREMENT':
12     return { count: state.count - 1 };
13     default:
14     return state;
15   }
16 }
17
18 const store = createStore(reducer);
19
20 export default store;
```

```
import { createStore } from 'redux';
const initialState = {
  count: 0,
};
```

```
function reducer(state = initialState, action) {
  switch (action.type) {
    case 'INCREMENT':
      return { count: state.count + 1 };
    case 'DECREMENT':
      return { count: state.count - 1 };
    default:
      return state;
  }
}
```

```
const store = createStore(reducer);
```

```
export default store;
```

2.7 Inside the folder, create a new file called **Counter.js**

Note: This will be our presentational component that will display the current count and the buttons to increment and decrement it

2.8 Create a new file called **Counter.js** in the directory **src/components**

2.9 Create a functional component named **Counter** that receives props for **count**, **increment**, and **decrement**

2.10 Render the **count** value, along with buttons for incrementing and decrementing the **count**

2.11 Export the **Counter** component

```
import React from 'react';
function Counter({ count, increment, decrement }) {
  return (
    <div>
      <h1>Count: {count}</h1>
      <button onClick={increment}>Increment</button>
      <button onClick={decrement}>Decrement</button>
    </div>
  );
}
export default Counter;
```

```
components / Counter.js
import React from 'react';

function Counter({ count, increment, decrement }) {
  return (
    <div>
      <h1>Count: {count}</h1>
      <button onClick={increment}>Increment</button>
      <button onClick={decrement}>Decrement</button>
    </div>
  );
}

export default Counter;
```

Step 3: Open the existing file called App.js

Note: The container component will establish the connection between the **Counter** component and the **Redux store**

3.1 In the **src** directory, open the existing file named **App.js**

3.2 Import **React**, the **useSelector** and **useDispatch** Hooks from **react-redux**, then **Counter** component

```
import { useSelector, useDispatch } from 'react-redux';
```

3.3 Create the **App** functional component

3.4 Use the **useSelector** Hook to select the **count** value from the **Redux store**

3.5 Utilize the **useDispatch** hook for accessing the **dispatch** function

3.6 Define the **increment** and **decrement** functions that dispatches the corresponding actions to the **Redux store**

3.7 Render the **Counter** component, passing the **count**, **increment**, and **decrement** as props

3.8 Export the **App** component

```
function App() {  
  const count = useSelector(state => state.count);  
  const dispatch = useDispatch();  
  
  const increment = () => {  
    dispatch({ type: 'INCREMENT' });  
  };  
  
  const decrement = () => {  
    dispatch({ type: 'DECREMENT' });  
  };  
  
  return <Counter count={count} increment={increment} decrement={decrement} />;  
}  
  
export default App;
```

```
import React from 'react';
import { useSelector, useDispatch } from 'react-redux';
import Counter from './components/Counter';

function App() {
  const count = useSelector(state => state.count);
  const dispatch = useDispatch();

  const increment = () => {
    dispatch({ type: 'INCREMENT' });
  };

  const decrement = () => {
    dispatch({ type: 'DECREMENT' });
  };

  return <Counter count={count} increment={increment}
    decrement={decrement} />;
}

export default App;
```

```
import logo from './logo.svg';
import React from 'react';
import { useSelector, useDispatch } from 'react-redux';
import Counter from './components/Counter';

function App() {
  const count = useSelector(state => state.count);
  const dispatch = useDispatch();

  const increment = () => {
    dispatch({ type: 'INCREMENT' });
  };

  const decrement = () => {
    dispatch({ type: 'DECREMENT' });
  };

  return <Counter count={count} increment={increment} decrement={decrement} />;
}

export default App;
```

Step 4: Wrap the App component with the Provider component

- 4.1 In the **src** directory, open the existing **index.js** file
- 4.2 Import **React**, **ReactDOM**, the **Provider** component from **react-redux**, the **store** from the **store.js** file, and the **App** component
- 4.3 Wrap the **App** component with the **Provider** component, passing the **store** as a property
- 4.4 Use the **ReactDOM.render** function to render the wrapped **App** component to the DOM

```
import React from 'react';
import ReactDOM from 'react-dom';
import { Provider } from 'react-redux';
```

```
import store from './store';
import App from './App';
```

```
ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById('root') );
```

```
import './index.css';
import reportWebVitals from './reportWebVitals';
import React from 'react';
import ReactDOM from 'react-dom';
import { Provider } from 'react-redux';
import store from './store';
import App from './App';

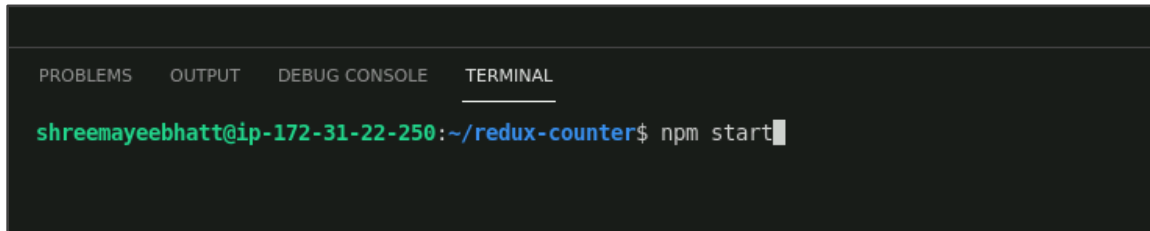
ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById('root')
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

Step 5: Run the application

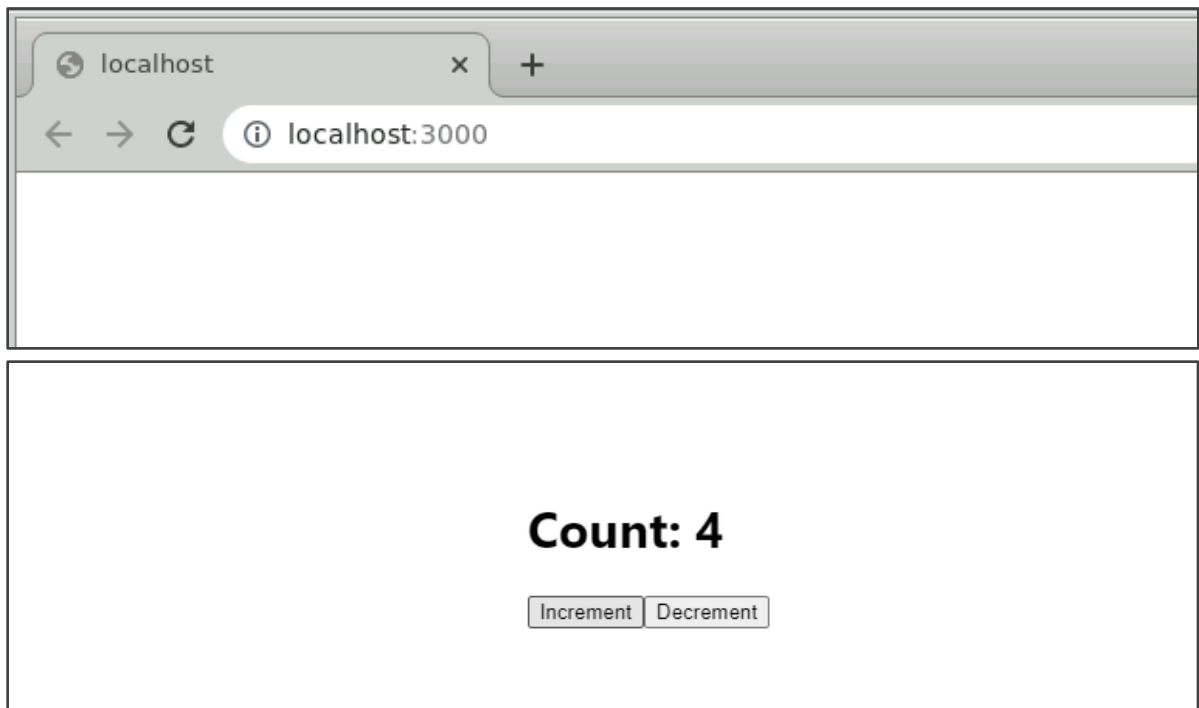
5.1 In the terminal, navigate to the project directory

5.2 Run the **npm start** command to start the development server



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
shreemayeebhatt@ip-172-31-22-250:~/redux-counter$ npm start
```

5.3 Open your browser and navigate to <http://localhost:3000>



You should now be able to see the **counter** app in your browser. Clicking on the buttons should increment or decrement the count, and the value should be stored and managed by the **Redux store**