

State Management in React with Redux



Efficient Global State Management for React Applications



Introduction to Redux. What is Redux?



- A predictable state container for JavaScript applications.
- Manages global state in a unidirectional data flow.
- Integrates seamlessly with React.

Key Concepts



- Actions
 - Plain JavaScript objects that represent events.
 - Must have a type property and can have a payload.
- Reducers
 - Pure functions that determine how the state changes in response to an action.
 - Combine smaller reducers to handle specific parts of the state.
- Store
 - The single source of truth for state.
 - Manages the application state and allows access and updates to it.

Installation and Setup



Install Redux and React-Redux

```
npm install redux react-redux
```

// Import the necessary libraries

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

```
import { Provider } from 'react-redux';
```

Creating Actions



```
// actionTypes.js
```

```
export const INCREMENT = 'INCREMENT';
```

```
export const DECREMENT = 'DECREMENT';
```

```
// actions.js
```

```
import { INCREMENT, DECREMENT } from './actionTypes';
```

```
export const increment = () => ({  
  type: INCREMENT  
});
```

```
export const decrement = () => ({  
  type: DECREMENT  
});
```

Creating Reducers



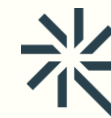
```
import { INCREMENT, DECREMENT } from './actionTypes';
```

```
const initialState = {  
  count: 0  
};
```

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

```
export default counterReducer;
```

Setting Up the Store



```
import { createStore } from 'redux';
import { Provider } from 'react-redux';
import counterReducer from './reducers';

const store = createStore(counterReducer);

function App() {
  return (
    <Provider store={store}>
      <Counter />
    </Provider>
  );
}

export default App;
```


Connecting React Components



```
import React from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { increment, decrement } from './actions';

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

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

export default Counter;
```

Redux DevTools




Enabling Redux DevTools:

- Useful for inspecting state changes and action dispatch.

```
import { createStore } from 'redux';  
import { composeWithDevTools } from 'redux-devtools-extension';  
import rootReducer from './reducers';  
  
const store = createStore(rootReducer, composeWithDevTools());
```

Benefits of Redux DevTools:

- Track state changes over time.
- Revert/Replay actions.
- Inspect actions and state tree.



The shift begins with

you
Trainer name

Trainer

Assistant name

Assistant

trainer@mail.com
assistant@mail.com