**Persisting State in Your React App with Redux-Persist**

[[](https://medium.com/@xbstrxct?source=post_page-----1e7dd877c58a--------------------------------)](https://medium.com/@xbstrxct?source=post_page-----1e7dd877c58a--------------------------------)

[Roman Andrieiev](https://medium.com/@xbstrxct?source=post_page-----1e7dd877c58a--------------------------------)

·

Follow

3 min read

·

Sep 18, 2023

1.2K

In the world of front-end development, React has emerged as a dominant player. Its component-based architecture and virtual DOM make it a powerful tool for building user interfaces. When it comes to managing the state of your React applications, Redux is a popular choice due to its predictability and ease of debugging. However, what happens when you need to persist your Redux state across page refreshes or even app restarts? This is where “redux-persist” comes into play.

Redux-persist is a library that seamlessly integrates with Redux to enable the persistence of your application’s state. In this article, we will explore how to use redux-persist in a React app with practical code snippets.

**Why Persist State?**

Before diving into redux-persist, it’s essential to understand the need for state persistence in your React applications. In most cases, your application’s state is stored in memory, and it gets reset when the page is refreshed or the app is closed and reopened. For some applications, this behavior might be acceptable. Still, many require the ability to save and load user data, settings, or session information even after the user leaves or refreshes the page.

Redux-persist offers a straightforward solution to this problem. It allows you to save the Redux store’s state to a persistent storage medium, such as local storage or AsyncStorage (for React Native applications), and then rehydrate the store with the saved state when the app loads.

Let’s see how to implement this in your React application.

**Getting Started**

First, make sure you have Redux set up in your React project. If you haven’t already, you can install Redux and create a store for your application. For this demonstration, we’ll assume you have the Redux store ready.

Next, you’ll need to install the redux-persist library. You can do this using npm or yarn:

npm install redux-persist  
# or  
yarn add redux-persist

Now that you have redux-persist installed, you can configure it to work with your Redux store.

**Configuration**

In your Redux store configuration file, import persistReducer from redux-persist and configure it to persist your desired parts of the state. You can choose which reducers and which properties of those reducers to persist.

// store.js  
import { createStore } from 'redux';  
import { persistStore, persistReducer } from 'redux-persist';  
import storage from 'redux-persist/lib/storage'; // Choose your storage engine  
import rootReducer from './reducers'; // Import your root reducer  
const persistConfig = {  
 key: 'root',  
 storage,  
 // Specify the reducers you want to persist  
 whitelist: ['user'], // In this example, we persist the 'user' reducer  
};  
const persistedReducer = persistReducer(persistConfig, rootReducer);  
export const store = createStore(persistedReducer);  
export const persistor = persistStore(store);

In this example, we configure redux-persist to persist the 'user' reducer, but you can customize this to fit your application's needs.

**Wrapping Your App**

Now that you have your Redux store configured with redux-persist, you need to wrap your application with a PersistGate component from redux-persist. This component ensures that your app waits for the persisted state to be retrieved before rendering.

// index.js  
import React from 'react';  
import ReactDOM from 'react-dom';  
import { Provider } from 'react-redux';  
import { PersistGate } from 'redux-persist/integration/react'; // Import PersistGate  
import { store, persistor } from './store';  
import App from './App';  
ReactDOM.render(  
 <Provider store={store}>  
 <PersistGate loading={null} persistor={persistor}>  
 <App />  
 </PersistGate>  
 </Provider>,  
 document.getElementById('root')  
);

**Usage**

With redux-persist set up, you can now use Redux in your React components as you normally would. Any state you specify in the persistConfig will be automatically persisted and rehydrated.

// userReducer.js  
const initialState = {  
 username: '',  
 email: '',  
 // ... other user-related properties  
};  
const userReducer = (state = initialState, action) => {  
 switch (action.type) {  
 case 'SET\_USERNAME':  
 return {  
 ...state,  
 username: action.payload,  
 };  
 case 'SET\_EMAIL':  
 return {  
 ...state,  
 email: action.payload,  
 };  
 // ... other reducer cases  
 default:  
 return state;  
 }  
};  
export default userReducer;

In the example above, any changes made to the ‘user’ reducer will be automatically persisted and reloaded when the app restarts or the page is refreshed.

**Conclusion**

Redux-persist is a powerful library that enhances your React applications by providing state persistence. This can be incredibly useful for applications that require users to log in or want to save user preferences and session data.

By following the steps outlined in this article, you can easily integrate redux-persist into your Redux-powered React app. This will allow your users to enjoy a seamless experience with data persistence across sessions and page reloads, ultimately leading to a more robust and user-friendly application.

[Redux Persist](https://medium.com/tag/redux-persist?source=post_page-----1e7dd877c58a---------------redux_persist-----------------)

[React](https://medium.com/tag/react?source=post_page-----1e7dd877c58a---------------react-----------------)

[React Native](https://medium.com/tag/react-native?source=post_page-----1e7dd877c58a---------------react_native-----------------)

[Redux](https://medium.com/tag/redux?source=post_page-----1e7dd877c58a---------------redux-----------------)

[Local State](https://medium.com/tag/local-state?source=post_page-----1e7dd877c58a---------------local_state-----------------)