**How to Use the React Redux Toolkit**

[[](https://medium.com/@taraparakj75?source=post_page-----b8dbd5e6bb81--------------------------------)](https://medium.com/@taraparakj75?source=post_page-----b8dbd5e6bb81--------------------------------)

[Kuldeep Tarapara](https://medium.com/@taraparakj75?source=post_page-----b8dbd5e6bb81--------------------------------)

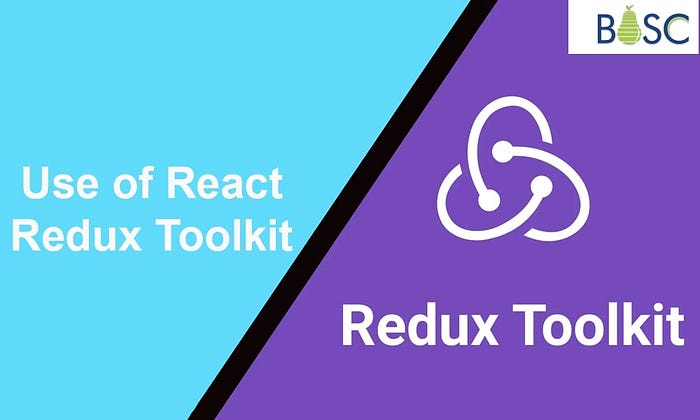
·

Follow

5 min read

·

Jul 3



A strong library called React Redux Toolkit makes it easier to create Redux apps. It offers a collection of standards and utilities that streamline and speed up Redux development which is easy and simple for [**React developers**](https://bosctechlabs.com/hire-reactjs-developers/). In this article, we’ll look at the fundamentals of the React Redux Toolkit and how it can make your Redux process more efficient.

**What is React Redux Toolkit?**

A description of the React Redux Toolkit and how it makes Redux development simpler.  
A description of the main components of the React Redux Toolkit, including the configureStore function, createSlice for reducer logic, and createAsyncThunk for asynchronous action processing.

**What Makes Redux Toolkit Useful?**

The preferred method for creating Redux logic is to use the Redux Toolkit package.  
Redux Toolkit makes it simpler and quicker to write Redux. We must declare the state, reducer, and action individually in common Redux. In Redux Toolkit, however, we describe it collectively inside createSlice. It is simpler and easier to understand. Redux Toolkit is faster than regular redux since less [**React boilerplate**](https://bosctechlabs.com/react-boilerplates-in-2023/) code has to be written.

**Using the React Redux Toolkit to Create a Redux Project**

A step-by-step tutorial for installing the React Redux Toolkit on a fresh React project.

**Instructions for installing the required dependencies**

Modifying the setup options and configuring the Redux store using the configureStore function.

**Creating Slices with createSlice**

The idea of slices, which include reducer logic and action producers, is introduced.  
An explanation of how to use createSlice to create a slice and define its initial state and reducer functions.

An explanation of how createSlice uses the specified reducer functions to construct action creators and action types automatically.

**Dispatching Actions and Accessing State**

Demonstration of using the useDispatch hook to dispatch actions from React components.  
using the useSelector hook to retrieve and use state from the Redux store in components.

**Handling Asynchronous Actions with createAsyncThunk**

Overview of createAsyncThunk’s asynchronous action handling capabilities.  
Using createAsyncThunk, an example of defining and sending an asynchronous operation.

**Managing Entities with Redux Toolkit**

The React Redux Toolkit introduces the createEntityAdapter function for handling normalized data.  
An example of how to create entity adapters and interact with entities using the CRUD method.

**Advanced Redux Toolkit Features**

Discussion of additional features offered by React Redux Toolkit, such as middleware setup, immutability with the immer library, and testing utilities.

**How to setup Create-React-App With Redux**

Let’s start by setting up a fresh React application with CRA for this Redux tutorial:

npm install -g create-react-app

create-react-app redux-tutorial

cd redux-tutorial

**We’ll add redux after that with:**

npm install — save react-redux @reduxjs/toolkit

**Firstly configure store. Create file src/store/index.js containing:**

import { configureStore } from ‘@reduxjs/toolkit’

import { combineReducers } from ‘redux’

const reducer = combineReducers({

// here we will be including reducers

})

const store = configureStore({

reducer,

})

export default store;

Instead of taking several function parameters, configureStore only takes one object. It’s because some Redux middleware has been included by default and the store has been set up to allow utilizing the Redux DevTools Extension.

Next, we must link our website with the React application. Import it into index.js like this:

import { Provider } from ‘react-redux’

import store from ‘./store’

ReactDOM.render(

<Provider store={store}>

<App />

</Provider>,

document.getElementById(‘root’)

)



**How To Structure Your Redux?**

Let’s now construct the simple login form and logout button displayed after authentication using Redux authentication. The organisation of the folders and files in your application is irrelevant to Redux itself. However, it is typically simpler to maintain that code when the logic for a given feature is co-located in a single location. Instead of dividing logic across several folders by “kind” of code, Redux.org advises that most applications should arrange files using the “feature folder” technique (all files for a feature in the same folder) or the “ducks” pattern (all Redux logic for a feature in a single file) (reducers, actions, etc).

Let’s add src/store/user.js store slice:

import { createSlice } from ‘@reduxjs/toolkit’

// Slice

const slice = createSlice({

name: ‘user’,

initialState: {

user: **null**,

},

reducers: {

loginSuccess: (state, action) => {

state.user = action.payload;

},

logoutSuccess: (state, action) => {

state.user = **null**;

},

},

});

export default slice.reducer

// Actions

const { loginSuccess, logoutSuccess } = slice.actions

export const login = ({ username, password }) => async dispatch => {

**try** {

// const res = await api.post(‘/api/auth/login/’, { username, password })

dispatch(loginSuccess({username}));

} **catch** (e) {

**return** console.error(e.message);

}

}

export const logout = () => async dispatch => {

**try** {

// const res = await api.post(‘/api/auth/logout/’)

**return** dispatch(logoutSuccess())

} **catch** (e) {

**return** console.error(e.message);

}

}

**Conclusion**

Redux development is made easier with the help of the React Redux Toolkit, which makes it more effective and accessible. Utilizing its features, like createSlice, createAsyncThunk, and configureStore, allows you to write less boilerplate code and concentrate more on developing the functionality of your application. Additionally, some of the complexity involved in implementing Redux in our React application is reduced by using the Redux Toolkit. Repetitive code is not necessary. Discover how to React Redux Toolkit enhances your Redux workflow and boosts productivity by taking the time to explore and experiment with it.

If you need more support and assistance with the React Redux toolkit, you can contact [**Bosc Tech Labs**](https://bosctechlabs.com/).

**Frequently Asked Questions (FAQs)**

**1. What does React’s Redux Toolkit do?**

The preferred method for creating Redux logic is to use the Redux Toolkit package. It was first developed to assist in addressing three frequent worries regarding Redux: “Setting up a Redux store is too difficult.”

**2. What distinguishes Redux from the Redux Toolkit?**

A JavaScript package called Redux is used to control the application state. Although it works nicely with React, it may also be utilised with other JavaScript frameworks. The Redux Toolkit is a collection of tools that makes Redux development easier.

**3. Can I use the Redux toolkit with React query?**

While Redux Toolkit Query and React Query have certain similarities, Redux Toolkit Query has one major benefit over React Query: when combined, they provide a comprehensive data fetching and state management solution for React apps.

Website : <https://bosctechlabs.com/how-to-use-the-react-redux-toolkit/>