

React.js

Regards: Hassan Bilal

TABLE OF CONTENTS

01 > Global State Management System

01

Global State Management System

Redux Toolkit – Installation

- The recommended way to start new apps with React and Redux is
- By using the official Redux+JS template for Create React App.
- Which takes advantage of Redux Toolkit.
- And React Redux's integration with React components.

Redux Toolkit – Installation – New React app

```
# Redux + Plain JS template  
npx create-react-app my-app --template redux
```

```
# Redux + TypeScript template  
npx create-react-app my-app --template redux-typescript
```

Redux Toolkit – Installation – New React app

- After installing react redux

```
# NPM  
npm install @reduxjs/toolkit
```

```
# Yarn  
yarn add @reduxjs/toolkit
```

Redux Toolkit – Create store.js file

- **Create a Redux store with `configureStore`**
 - `configureStore` accepts a `reducer` function as a named argument
 - `configureStore` automatically sets up the store with good default settings

app/store.js

```
import { configureStore } from '@reduxjs/toolkit'

export const store = configureStore({
  reducer: {},
})
```

Provide the Redux store to the React application components

- Put a React-Redux `<Provider>` component around your `<App />`
- Pass the Redux store as `<Provider store={store}>`

index.js

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

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



Create a Redux "slice" reducer with `createSlice`



- Call `createSlice` with a string name, an initial state, and named reducer functions
- Reducer functions may "mutate" the state using Immer

```
import { createSlice } from '@reduxjs/toolkit'
```

```
const initialState = {  
  value: 0,  
}
```



```
export const counterSlice = createSlice({  
  name: 'counter',  
  initialState,  
  reducers: {  
    increment: (state) => {  
      state.value += 1  
    },  
    decrement: (state) => {  
      state.value -= 1  
    },  
    incrementByAmount: (state, action) => {  
      state.value += action.payload  
    },  
  },  
})
```




Actions




Slice

```
export const { increment, decrement, incrementByAmount } = counterSlice.actions
```



```
export default counterSlice.reducer
```



Redux Toolkit – Add Slice Reducers to the Store

app/store.js

```
import { configureStore } from '@reduxjs/toolkit'
import counterReducer from '../features/counter/counterSlice'

export const store = configureStore({
  reducer: {
    counter: counterReducer,
  },
})
```

Use the React-Redux `useSelector/useDispatch` hooks in React components

- Read data from the store with the `useSelector` hook
- Get the `dispatch` function with the `useDispatch` hook, and dispatch actions as needed

```
import { useSelector, useDispatch } from 'react-redux' ←
import { decrement, increment } from './counterSlice' ←

export function Counter() {
  const count = useSelector((state) => state.counter.value) ←
  const dispatch = useDispatch() ←

  return (
    <div>
      <div>
        <button
          aria-label="Increment value"
          onClick={() => dispatch(increment())} ←
        >
          Increment
        </button>
        <span>{count}</span>
        <button
          aria-label="Decrement value"
          onClick={() => dispatch(decrement())} ←
        >
          Decrement
        </button>
      </div>
    </div>
  )
}
```

Redux Toolkit – Summary – 1

- **Create a Redux store with `configureStore`**
 - `configureStore` accepts a `reducer` function as a named argument
 - `configureStore` automatically sets up the store with good default settings

Redux Toolkit – Summary – 2

Provide the Redux store to the React application components

- Put a React-Redux `<Provider>` component around your `<App />`
- Pass the Redux store as `<Provider store={store}>`

Redux Toolkit – Summary – 3

Create a Redux "slice" reducer with `createSlice`

- Call `createSlice` with a string name, an initial state, and named reducer functions
- Reducer functions may "mutate" the state using Immer

Redux Toolkit – Reducers to the Store – 4

app/store.js

```
import { configureStore } from '@reduxjs/toolkit'
import counterReducer from '../features/counter/counterSlice'

export const store = configureStore({
  reducer: {
    counter: counterReducer,
  },
})
```

Redux Toolkit – Summary – 5

Use the React-Redux `useSelector/useDispatch` hooks in React components

- Read data from the store with the `useSelector` hook
- Get the `dispatch` function with the `useDispatch` hook, and dispatch actions as needed

<QnA>

>

Thanks!

<

Regards: Hassan Bilal