Redux

Redux follows a unidirectional data flow

There are 3 core components in redux.

1. Action
2. Reducer
3. Store

Redux Middleware

Redux middleware is a function or a piece of code that sits between action and reducer and can interact with the dispatched action before reaching to the reducer function.

Redux middleware provides a way to extend the functionality of redux by allowing us to apply additional logic or actions before or after the state is updated. It is commonly used for tasks such as logging actions, making asynchronous api calls, or handling side effects.

Redux saga

Redux saga is a middleware library for reduxjs that helps manage side effects in our application, such as asynchronous operations like data fetching and handling complex asynchronous flow like chaining multiple requests or cancelling requests.

Saga uses generator function to handle asynchronous operation. These generator function allow us to write asynchronous code in synchronous style, making it easier to manage complex asynchronous flow.

Saga, we define our side effects declaratively using special functions called “effects” like “call”, “put”,”take”,”select”,etc.

Redux thunk

React context and React redux

|  |  |
| --- | --- |
| Context API | Redux |
| Built in tool that ships with React | Additional installation required, driving up the final bundle size |
| Requires minimal setup | Require extensive setup to integrate it with a React application |
| Specifically designed for static data, that is not often refreshed or updated | It effectively manages both static and dynamic data, providing seamless control and updates throughout the application |
| Adding new contexts requires creation from scratch | Redux js allow for easy expansion by making it simple to add new data or action after the initial setup |
| Debugging can be hard in highly nested react component structure even with dev tool | Incredibly powerful redux dev tools to ease debugging |
| We can change state in it | The state is read only we cannot change them directly |
| It rerenders all components whenever there is any updates in the provider value prop | It only rerender the updated components |
| It is better to use with small application | It is perfect for larger application |
| It is easy to understand and requires less code | It is quite complex to understand |

React context API is more suitable when

The state updates are infrequent or simple

The application is small or medium-sized

The state management requirements are simple and straightforward

The state is only needed in a few components, and it is not critical to have centralized state management.

Redux is more suitable when

The state updates are frequent or complex

The application is larger or complex

The state management requirements are complex and require a centralized approach.

The state is needed in many component, and it is critical to have a predictable and reliable way of managing it.