What is react redux

Redux is a**popular state management library for Javascript**. We can use redux in React for creating stores and it is often combined with React via the React-Redux library. **React Redux is a state management tool, when our application grows, we need a state management tool like Redux to manage the global state, it can access across many parts of our React application.**

The state in react is often referred to as a **single source of truth** because it is never mutated or modified but **always recreated,**whenever there is a change in state.

So whenever we want to change the state**we dispatch an action to the Store object** via reducer function, based only on **action type and existing state** we can update part of the state or whole state. The state is a Javascript object that represents the entire state of a redux application. It can be a single value or a more complex object.

**We inject store objects into our application using the Provider component**from the react-redux library. Now our components inside our application can consume the state or part of the state object. The state is managed through the unidirectional data flow.

**Store:**  
The state is accessible through the entire application that manages by a centralized container l known as Store. The Store object has a state, and some methods like dispatch, getState, and more methods. The only way to access the store is by dispatching an action via reducer.

With redux, we need to centralize all of our logic into reducers and action as much as possible  
and instead, our react component is only responsible for showing data they need, and not  
responsible for fetching data at all.

## Why use React-Redux?

The Redux is a state management library and it makes complex applications easier to write and manage. In most cases we don’t need Redux, special on small applications where data isn’t changed often [context API](https://edupala.com/when-and-how-to-use-react-context-api/)seems to be a better choice. But applications like e-commerce and other complex application where state changes often need state management like Redux.

For example, useDispatch lets you dispatch an action to update the state in the store. Say you have an application to build question sets for quizzes and you want to add a question:

1. Redux State management library makes creating complex applications easier.
2. Now introduce or useSelect, useDispatch, and useStore in React-redux 7.1, making less code and easier to learn.
3. Server-side rendering is possible with Redux.
4. It is kept up-to-date with any changes to the state object to ensure that our React components behave as expected.
5. Commonly used a lot of problems have been resolved by the community.
6. It implements many performance optimizations internally, which allows to components rerender only when it actually needs.
7. With @reduxjs/toolkit, we don’t need to install thunk separately it is included in reduxjs/toolkit and we can perform the async operation with**createAsyncThunk function.**

### What’s the difference between Context API and React Redux?

|  |  |
| --- | --- |
| **Context API** | **Redux** |
| Is part of React library and no need to install any third-party library. Is built specifically for React. | The redux is predictable state management for Javascript. We need to install redux, which adds an additional bundle to our code. |
| Easy to share data between parents and nested children but difficult to build store components. | Excellent documentation. Redux is much more powerful and provides a large number of features that the Context API doesn’t provide. |
| Specifically designed for static data, that is not often refreshed or updated | Can use with both static and dynamic data. |
| Don’t have any specific dev tools and debugging can be hard in the highly nested components. | We can use the powerful Redux Dev Tools browser extension for ease of debugging. |
| It **re-renders all component**s whenever there is any update in the provider’s value prop. | React Redux implements several optimizations to ensure your actual component only re-renders when actually necessary. |

React redux vs context API

## React Redux Architecture

Diagram

Description automatically generated

In the above diagram, we have a different part of React Redux architecture.

**Store object:** A Store is a Javascript object that holds the application state, there should only single store in the Redux application. Store object has some method like.

1. dispatch(action) : dispatch an action.
2. getState() : return current state.
3. subscribe(listener) Adds a change listener.
4. replaceReducer(nextReducer): Repalces the reducer.

**Action creators** are just simple functions that return an action. Action is a plain Javascript  
object. Action must always have a type defined and they can optionally have a payload. Here is what an action creator function looks like.

const addTodo = argumentTask => {

return {

type: 'AddTodod',

payload: argumentTask

}

}

**Dispatch:** Redux store has a method called dispatch. The only way to update the state is to call the dispatch method with the action object as an argument.

store.dispatch(addTodo);

**Reducers**are responsible for changing the application state over time. So whenever we dispatch an action object, it sends it to all of the different reducers in our application. Each reducer has the option to return a different piece of state than the usual based on the type of action that was received. **Whenever there is a change in the State object in the Redux store, it will notify all consumer components of the state, to update the View**.