An in-depth examination of the React-Redux development framework

Harshala Ashok Doiphode
Software Engineering
SJSU ID : 015222480

Abstract :

As developers, making cross-platform apps is a difficult undertaking that requires familiarity with two or more native systems' capabilities. Furthermore, certain hybrid application frameworks on the market were unable to provide a consistent experience across all native platforms for the same user. As a solution to this problem, in this paper I will be exploring advantages of react-redux framework to create hybrid applications which could be used on windows or iso machines and provides the same user experience to both users.

Introduction:

Creating a web application which is compatible with all platforms seems a tedious task for developers. The common issue persists when developers have to go through the specifics of a particular platform in order to make the application compatible with that platform. For example, in case of IOS development knowledge of swift is needed and in case of Android , react native and Kotlin is being used. There are various applications in the market which tend to address this problem and help developers to create hybrid applications. However, these applications are not very successful in providing sophisticated experience for developers of multiple platforms. Addressing this need, in this research paper I will be exploring the React -Redux framework which is capable of providing sophisticated ways to develop cross-platform applications.

Objective :

- To compare React-Redux with available frameworks for hybrid web application development.
- Explore how React-Redux works.
- Explore the advantages of React-Redux Framework over other frameworks.
- Provide proof of concept (POC).

History:

React Redux was developed by Dan Abramov and Andrew Clark in 2015. React-Redux was actually developed while creating a POC for Flux. Abramov remarks, "I was trying to make a

proof of concept of Flux where I could change the logic. And it would let me time travel. And it would let me reapply future actions on the code change".
This idea leads to the discussion of what if Flux store is not a store but the reducer function. This chain of thought fueled the development of React-Redux. The popular userReducer was introduced in React-Redux in 2019.

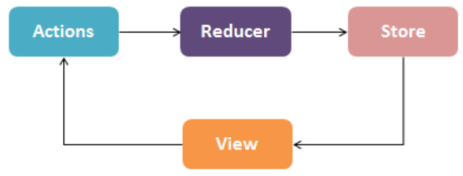
Introduction to React-Redux working :

Redux is a state container that helps you develop JavaScript apps that are easy to test and act reliably across client, server, and native platforms. Although it is basically a state management tool with React, one can use Redux with any application with a javascript framework whether it be react-native for android or any other framework.
Redux is pretty much lightweight, around 2KB (including all its dependencies). With redux , the state of the application is stored in the store and any other child component can access that state and perform logical operations. Although this sounds simple, this feature simplifies the React development and lets developers develop many more features by manipulating the application's state.

Working of Redux :

The complete state of the program is stored in a central store. Without needing to transmit down props from one component to another, each component may access the saved state.

Redux Architecture :



Redux Architecture

Terminologies :

Action :
actions are things that happen. You can only communicate data from your application to your Redux store using these. User interactions, API calls, and event form submissions can all provide data.The store.dispatch() method is used to send actions. Actions are simple JavaScript objects with a type property that specifies the sort of action to be performed. They must also have a payload that contains the data that the action should function with. An action creator is used to create actions.
Example :
In my POC for this paper I have implemented below Action :

```
export const add_user = ({
    userID = "",
    emailID = "",
    type = ""

} = {}) => {

    return {
        type: 'add_user',
        user: {
            emailID,
            userID,
            type

        }
    }

}
```

Reducer :

Reducer takes an application's current state, executes an action, and then returns a new state. These states are objects that describe how an application's state changes in response to an action made to the store.Reducer is similar to the Reduce function in Javascript.

Example :

In my POC for this paper I have used below Reducer :

```
export default (state = user_global_state, action) =>
{
    switch (action.type) {
        case 'add_user':
            {

                return action.user
            }
        case 'remove_user':
```

```
                {
                    return action.user
                }


            default:
                {
                    return state;
                }
        }
    }
```

Store :

Sore hold the application state.

Example :

In my POC, I have created a store to store the user data

```
    import { createStore, combineReducers,
applyMiddleware, compose } from 'redux'
    import userReducer from '../reducer/user_reducer.js'


    export default () => {
        // Store creation
        const store = createStore(
            combineReducers({
                user: userReducer
            }),
            window.__REDUX_DEVTOOLS_EXTENSION__ &&
window.__REDUX_DEVTOOLS_EXTENSION__()
        )

        return store
    }
```

Advantages of Redux :

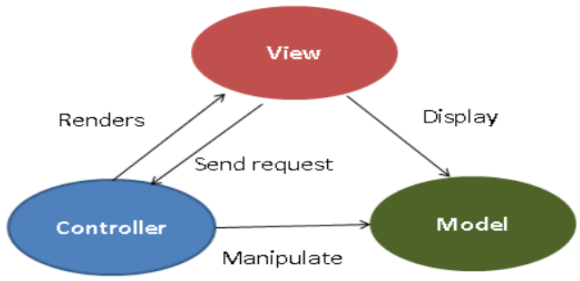
- With redux , state is stored in the delegated space, so there is no need to pass props between components and this makes application development a lot more sophisticated.
- State is always predictable in Redux, because the same result will be produced by the reducer when the same input is passed. Because reducers are pure function, but state is mutable. This powerful feature helps to develop features like undo/redo in web applications.
- Debugging is easy in redux, as state is stored at one place.
- Performance benefits of using redux are more than anticipated, with redux your application will render components only when it actually needs to.
- State persistence to local storage of the browser is easy with redux.

Comparison between MVC , Flux and Redux :

MVC :

It is popular architecture , which divides the application into three components

1. Model : Maintains the behaviour of application.
2. View : Displays model in UI.
3. Controller : Serves as interface between Model and View.

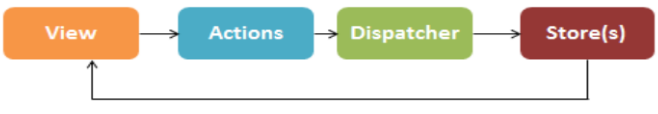


MVC Architecture

Flux :

Facebook developers developed Flux as an alternative to MVC.

1. Store : serves as a container for app state and logic
2. Action : Enables data passing to dispatcher.
3. View : Displays model in UI.
4. Dispatcher : Coordinates action and dispatch store.



Flux Architecture

Advantages of Redux over Flux and MVC :

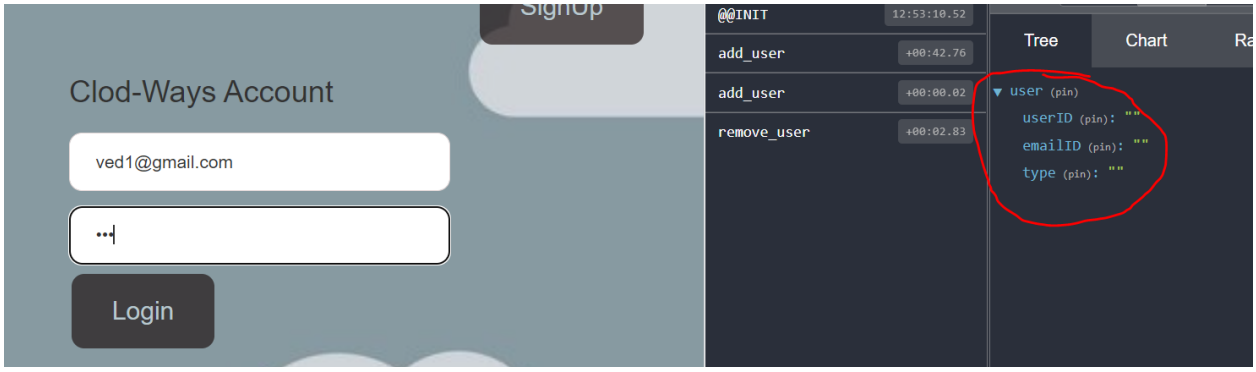
1. Follows unidirectional flow as compared to MVC which follows bidirectional flow. This feature simplifies state management.
2. Redux includes a single store as compared to flux which includes multiple stores.
3. Single store makes debugging lot easier

Proof of concept :

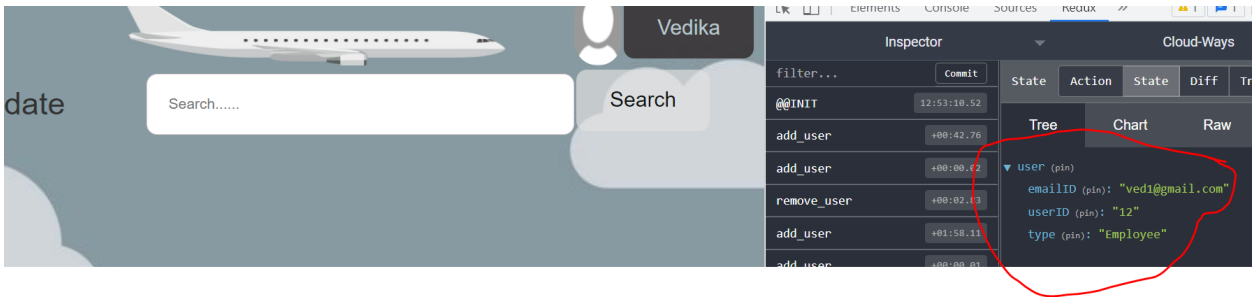
In order to implement the Redux, I have created a simple web application with login and signup functionality. In this web application I have created a Redux store to store the user data like “userID” , “emailid ” and “userType”. Upon login, the application fatched the information from the database and put the information in the redux store. I have used Redux chrome extension to view data in Redux store :

Below is snap upon login :

Before login (Redux store is empty):



After hitting login button :



Same thing happens after Signup as well.

As now userId is already maintained in the redux store, whenever there is need to implement anything based on userID, for example if the user is admin different dashboard will be displayed and if user is customer different dashboard will be displayed. Such kind of conditional functionalities will be easily implemented without getting back -forth with the backend server for the same data over and over again.This makes the application's performance better.

Conclusion :

Redux is a powerful framework when it comes to cross platform web application development. Redux makes applications faster. Redux optimizes the performance by reducing unnecessary rendering of components.

References :

https://blog.logrocket.com/why-use-redux-reasons-with-clear-examples-d21bffd5835/#what
https://www.ijcrt.org/papers/IJCRT2004607.pdf
https://dzone.com/articles/managing-state-in-angular-with-ngrxstore
https://www.clariontech.com/blog/mvc-vs-flux-vs-redux-the-real-differences