Dec 1, 2017 · 3 min read

MobX vs Redux with React: A noob's comparison and questions

State management in React has been a topic of discussion in the Javascript fraternity. I recently wanted to implement state management in one of my projects wanted to do little research before making the big decision.

Is it **MobX or Redux?**

In this article we will explore the benefits and tradeoffs of both. And how do you choose between any one of them? This article assumes that you have basic knowledge about React state management.

Both Redux and MobX works well with react.

Single store vs Multiple Store

In simple words a store is a place where you keep all your data.

Redux always have one large store in which all the states are stored. MobX typically has more than one store. As a result in MobX you can logically separate your stores.

Also, in Redux the data is usually normalised. In MobX, you can keep an denormalised data.

Plain data vs Observable data

Redux uses a normal Javascript object to store the data. On the other hand MobX uses an observable to store the data.

Why does that matter?

You can listen to an observable and automatically track changes that occur to the data. In redux all the updates have to be tracked manually.

Immutable vs mutable (Pure vs Impure)

Redux uses immutable states. That means the states are read-only, and you cannot directly overwrite them. In Redux the previous state is replaced by a new state. As a result Redux is pure, or it uses pure functions.

This can come in really handy when you have to revert back to a previous state. Eg—An undo action.

In MobX states can be overwritten. You can simply update a state with the new values. As a result MobX can be termed as impure.

Learning

MobX is much easier to learn and has a steady learning curve. Especially since most of the traditional Javascript developers are familiar with OOP, it is easy to get hold of MobX. There is lot of abstraction in MobX which makes it easier too. You need not bother about many things, you otherwise have to take care. (Eg: Subscribing to state)

Redux follows functional programming paradigm. For Javascript developers without experience in functional programming, it can be hard to get a complete grasp of Redux straight away. You will need to learn about middleware like Redux Thunk which will make the learning curve even steeper.

In MobX there is lot of built in abstraction, and this leads to less code. But when implementing redux you will end up writing lot of boilerplate

code.

Debugging

Since there is more abstraction, debugging becomes a lot harder. And the developer tools existing for MobX is also just average. The outcomes can become unpredictable at times.

On the other hand, Redux provides kickass developer tools including time travelling. And with pure functions and less abstraction, debugging in Redux will be a better experience compared to MobX. Following the flux paradigm makes Redux more predictable.

Scale and Maintenance

Because of the whole pure functions things and functional programming paradigm Redux is more maintainable. Things are more under control with Redux.

Developer Support

Redux hands down. The developer community of Redux is way ahead of the MobX community.

Ask yourself

1)Is the application small and simple?

Choose MobX

2)Prefer to build the app fast?

Choose MobX

3)Large team looking for more maintainable code?

Choose Redux

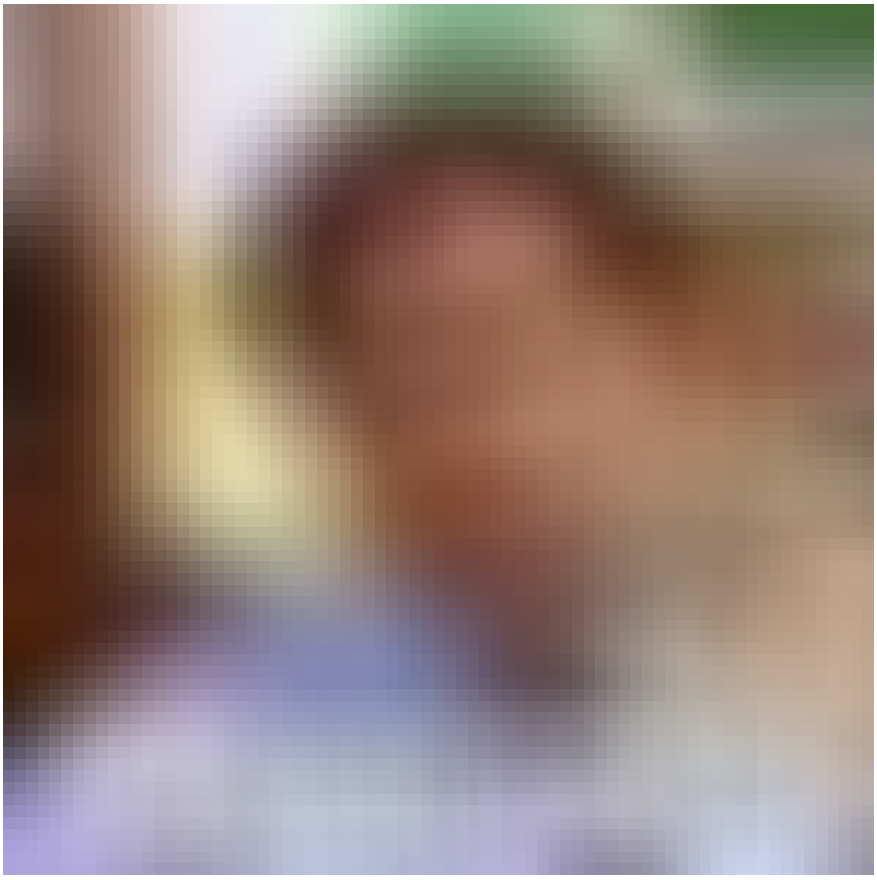
4)Complex app with scalable option?

Choose Redux

Final and the most important question—Which one do you like?

Choose that.

Because:



Thanks for reading

If you liked what you've read, subscribe here to stay up-to-date with my posts!