Recoil.js — High-Performance State Management for React Simplified

What makes Recoil so good and how can you take advantage of it?

Caelin Sutch Follow
Nov 17, 2020 · 4 min read ★

Introduction

State management for React is a huge field, encompassing giants like Redux and Mobx. Recoil.js is a relatively new kid on the block of state management libraries. Created and open-sourced by Facebook, it seems promising for simplifying global state management for React developers. It offers low-boilerplate and simple global state management that's implemented simpler and easier than other state management libraries.

Dave McCabe, a software engineer at Facebook who worked on Recoil, gave an interesting talk on Recoil in May 2020, citing a need for high performance and efficiency in his reason for building a new state management library.

Dave McCabe's talk on Recoil.js

First, lets address the need for global state management at all. With a component-based structure in React, we can manage separate state and logic for each component, and can even reuse the logic for them. But what happens if you want multiple components to access and update the state? This is where a global state management framework is needed.

Photo by Ferenc Almasi on Unsplash

Redux — The Global State Powerhouse

If you've spent much time in React, you've probably ran into Redux. With over 20k stars on Github, Redux is an established powerhouse in the global state management space.

Redux is composed of stores, actions, and reducers. Data flow is uni-directional, and data management is completely separated from the UI. Redux has quite a lot of boilerplate and is strict on how code should be organized. While optimized for larger projects, it has a large learning curve compared to recoil.

State for each redux store/action/reducer is immutable, meaning that the reducer returns a new state each time it's updated, which can cause excessive memory use for larger data. In recoil, it's easier to break data down into smaller chunks, creating multiple `Atoms` independent of each other to store state.

The React Context API

The React Context API is another method for managing global state. Context provides a great way to pass data down the component tree without manually passing data down at each level. Context is set up for states that have low-frequency updates, such as themes or locale.

Context API Diagram (source)

The downside of Context comes in performance. Each Provider can only store a single value and on each update, each component that consumes the value provided has to re-render. Yes, you can split up state into multiple provider, but this can lead to provider hell:

```
function App() {
  return (
    <AuthProvider>
      <UserProvider>
        <AnotherDataProvider>
          <NetProvider>
            <ThisIsGettingReallyBigProvider>
              <OMyGodTheresMoreProviders>
                <finallySomeRealComponents />
              </OMyGodTheresMoreProviders>
            </ThisIsGettingReallyBigProvider>
          </NetProvider>
        </AnotherDataProvider>
      </DataProvider>
    </AuthProvider>
  );
}
```

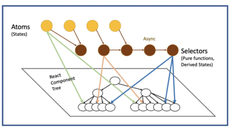
This is inefficient, hard to maintain, and not very performant. Creating context providers at the top of a tree causes the whole tree to **unmount** and **mount** again, taking a large performance hit, meaning global state isn't very dynamic.

Recoil addresses this by adding the ability dynamic global state with the `atomFamily` and `selectorFamily` .

The Context API also introduced **coupling** between the root of the tree and the leaves of the tree, making it harder to code split.

The Recoil Advantage

Unlike other state management libraries, Recoil defines a directed graph orthogonal, but it is also intrinsic and attached to the React component tree (see the Recoil data-flow graph at the top). In this data-flow graph, state changes flow from the roots of the graph (atoms) through pure functions for derived states (selectors) and into components. This means high-performance state management even for massive applications!



Recoil Orthogonal Graph

There's a couple of other main advantages for Recoil:

- A boilerplate-free API where shared state has the same simple get/set interface as React local state
- Concurrent Mode and other new React features have support
- State definition is incremental and distributed, not centralized, making it easier to maintain
- State can be replaced with derived data without modifying components that use it
- Derived data can move between being asynchronous and synchronous easily
- It's easy to persist the entire application state in a way that's backward-compatible
- Facebook backing (!!)
- Derived Data and queries to compute things based on state efficiently and robustly

Conclusion

Although a newish framework, Recoil has a variety of advantages over current global state management solutions. Recoil is a high performant solution that can handle fast granular updates anywhere in the tree, perfect for React applications that require simple and high-performance global state management.

Keep in Touch

There's a lot of content out there, I appreciate you reading mine. I'm a young entrepreneur and I write about software development and my experience running and growing companies. You can signup for my newsletter here

Feel free to reach out and connect with me on Linkedin or Twitter.

React State Management Recoiljs Web Development Software Development

WRITTEN BY
Caelin Sutch
Follow
Passionate about building cool shit. First-year undergrad student studying EECS and Business @ UC Berkeley MET Software developer at Carline.

Weekly Webtips
Follow
Explore the world of web technologies through a series of tutorials

More From Medium

An introduction to TypeScript for JavaScript developers
Sam Piggott in CodeSnap

Using Different Number Bases To Display Unique Permutations
Christopher Doornbos

Scaffolding a project using lerna and lerna-changelog
Jaya Krishna Namburu

Building reason-react with parcel-bundler
Sam Stotsky in manifoldco

React Native: Take images and upload it to a Node.js server
Louis Petrik in Teaching Tech

Using React Hooks and the Context API to build a Pokemon web app
TK in Frontend Digest

Automatically Start Node.js Server on System Restarts
Samuel Eh

React Component Communication with Redux
Ergün Kezer in Neyasis Technology