Motivation

As the requirements for JavaScript single-page applications have become increasingly complicated, **our code must manage more state than ever before**. This state can include server responses and cached data, as well as locally created data that has not yet been persisted to the server. UI state is also increasing in complexity, as we need to manage active routes, selected tabs, spinners, pagination controls, and so on.

Managing this ever-changing state is hard. If a model can update another model, then a view can update a model, which updates another model, and this, in turn, might cause another view to update. At some point, you no longer understand what happens in your app as you have **lost control over the when, why, and how of its state.** When a system is opaque and non-deterministic, it's hard to reproduce bugs or add new features.

As if this weren't bad enough, consider the **new requirements becoming common in front-end product development**. As developers, we are expected to handle optimistic updates, server-side rendering, fetching data before performing route transitions, and so on. We find ourselves trying to manage a complexity that we have never had to deal with before, and we inevitably ask the question: is it time to give up? The answer is *no*.

This complexity is difficult to handle as **we're mixing two concepts** that are very hard for the human mind to reason about: **mutation and asynchronicity.** I call them Mentos and Coke. Both can be great in separation, but together they create a mess. Libraries like React attempt to solve this problem in the view layer by removing both asynchrony and direct DOM manipulation. However, managing the state of your data is left up to you. This is where Redux enters.

Following in the steps of Flux, CQRS, and Event Sourcing, **Redux attempts to make state mutations predictable** by imposing certain restrictions on how and when updates can happen. These restrictions are reflected in the three principles of Redux.

Edit this page

*Last updated on **Jun 18, 2023***