SPA needs to manage more state, including:

* server responses
* cached data
* locally created data (that has not yet been persisted to the server)
* UI state
  + active routes,
  + selected tabs,
  + spinners,
  + pagination controls, and so on

When a system is opaque and non-deterministic, it's hard to reproduce bugs or add new features.

As developers, we are expected to handle

* optimistic updates,
* server-side rendering,
* fetching data before performing route transitions,
* and so on.

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.**

* Libraries like [React](http://facebook.github.io/react) attempt to solve this problem in the view layer by removing both asynchrony and direct DOM manipulation.
* managing the state of your data is left up to you. This is where Redux enters.

Below is basically the whole idea of Redux. The main idea is that you describe how your state is updated over time in response to action objects, and 90% of the code you write is just plain JavaScript, with no use of Redux itself, its APIs, or any magic.

* app’s state is described as a plain object.
* To change something in the state, you need to dispatch an action. An action is a plain JavaScript object that describes what happened.
* To tie state and actions together, we write a function called a reducer. it’s just a function that takes state and action as arguments, and returns the next state of the app.
  + It would be hard to write such a function for a big app, so we write smaller functions managing parts of the state.
  + And we write another reducer that manages the complete state of our app by calling those two reducers for the corresponding state keys:

**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](https://redux.js.org/introduction/threeprinciples) of Redux.

**Single source of truth**

**The** [**state**](https://redux.js.org/glossary#state) **of your whole application is stored in an object tree within a single** [**store**](https://redux.js.org/glossary#store)**.**

# State is read-only

**The only way to change the state is to emit an** [**action**](https://redux.js.org/glossary#action)**, an object describing what happened.**

# Changes are made with pure functions

**To specify how the state tree is transformed by actions, you write pure** [**reducers**](https://redux.js.org/glossary#reducer)**.**