A Short Introduction to Redux

Joe Parry
Motivation

State is very hard to deal with in all applications. In web apps, it feels even more difficult.
Problems

- State stored in JS and in DOM
- Async behaviour
- Synching of data browser <-> client
- JS code often becomes ‘spidery’ at scale
- Code is hard to organise
- Command ‘routing’ tends to get complex
What about React?

React
A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES

Get Started
Download React v15.0.2

JUST THE UI
Lots of people use React as the V in MVC. Since React makes no assumptions about the rest of your technology stack, it’s easy to try it out on a small feature in an existing project.

VIRTUAL DOM
React abstracts away the DOM from you, giving a simpler programming model and better performance. React can also render on the server using Node, and it can power native apps using React Native.

DATA FLOW
React implements one-way reactive data flow which reduces boilerplate and is easier to reason about than traditional data binding.
React only fully solves one problem

- State stored in JS and in DOM
- Async behaviour
- Synching of data browser <-> client
- JS code often becomes ‘spidery’ at scale
- Code is hard to organise
- Command ‘routing’ tends to get complex
• There are lots of these sorts of models. Which one to choose?
  – flux, reflux, alt.js, fluxxor, nuclearjs, ...
Redux

• A small functional flux-like library, written by Dan Abramov (now at FaceBook)
Redux Basic Terminology

- **Action**
  - An object with a type and a payload that is dispatched to the store
- **Dispatch**
  - The way actions are given to the store
- **Reducer**
  - A function that produces new state from actions
- **Selector**
  - A function that picks out parts of the store, or derives data from it, typically used to display things to the user
- **Store**
  - A simple wrapping around the state
Redux Code Organisation

• Components
  – ‘Dumb’ pieces of code that get told:
    • which properties to use
    • which functions to call when something happens

• Containers
  – Where components get hooked up to bits of the state & dispatchers

• State/Store
  – The actions, reducers, selectors
Redux helps with other problems

- State stored in JS and in DOM
- Async behaviour
- Synching of data browser <-> client
- JS code often becomes ‘spidery’ at scale
- Code is hard to organise
- Command ‘routing’ tends to get complex
Redux Extensions

• react-redux
  – Essential if using Redux with React:
    • connect function for your containers
    • <Provider/> component hooks in the store

• reselect
  – Essential: memoizes your selectors so they become super efficient

• redux-thunk, redux-sagas, redux-promise etc.
  – Some people claim these help with actions that are async. But you probably don’t need them.
More Reading

• Video: React.js Conf 2016 - Jared Forsyth
  • Puts Redux in a bit more context

• https://egghead.io/series/getting-started-with-redux
  • Excellent intro videos

• https://github.com/markerikson/react-redux-links
  • Curated list of links for more reading
Questions?
Thanks!

joe@cambridge-intelligence.com
@parry_joe