Lecture 9: Redux

Jordan Hayashi

Previous Lectures

- APIs
- Making Network Requests
- Promises, Async/Await
- Data Transformations
- Authentication
- HTTP Methods
- HTTP Response Codes
- Expo Components

Scaling Complexity

- Our apps have been relatively simple, but we’re already starting to see bugs related to app complexity
    - Forgetting to pass a prop
    - Directly managing deeply nested state
    - Duplicated information in state
    - Not updating all dependent props
    - Components with large number of props
    - Uncertainty where a piece of data is managed

Scaling Complexity: Facebook

- Facebook found the MVC architecture too complex for their scale
- The complexity manifested itself into bugs
- Facebook rearchitected into one-way data flow

https://youtu.be/nYkdrAPrdcw?t=10m22s

Flux

- "An application architecture for React utilizing a unidirectional data flow"
  - The views react to changes in some number of "stores"
  - The only thing that can update data in a store is a "dispatcher"
  - The only way to trigger the dispatcher is by invoking "actions"
  - Actions are triggered from the views
- Many implementations
  - https://github.com/facebook/flux
  - https://github.com/reactjs/redux
    - Whether redux is an implementation of Flux is an opinion that can be argued either way

Redux

- A data management library inspired by Flux
- Single source of truth for data
- State can only be updated by an action that triggers a recomputation
- Updates are made using pure functions
- Action → Reducer → Update Store

https://redux.js.org

simpleRedux/

Reducer

- Takes the previous state and an update and applies the update
- Should be a pure function
    - Result is deterministic and determined exclusively by arguments
    - No side effects
- Should be immutable
    - Return a new object
- What is responsible for invoking the reducer?

Store

- Responsible for maintaining state
- Exposes getter via getState()
- Can only be updated by using dispatch()
- Can add listeners that get invoked when state changes

Actions

- An action is a piece of data that contains the information required to make a state update
  - Usually objects with a type key
  - https://github.com/redux-utilities/flux-standard-action
- Functions that create actions are called action creators
- Actions must be dispatched in order to affect the state

`simpleRedux → redux`

- Our redux implementation has a very similar API
  - Missing a way to notify that state has updated
- How do we get the info from the store to our components?
  - `store.getState()`
- How do we update the store?
  - `store.dispatch()`
- How do we get the application to update when the store changes?

Review: HOCs

- Higher-Order Components take components as arguments or return components
- We could create a HOC that does the following:
  - Check for state updates and pass new props when that happens
  - Automatically bind our action creators with `store.dispatch()`
- We'd also need to subscribe to store updates
- https://github.com/reactjs/react-redux