React Redux Hooks

November 5th 2020

Toby Dussek

Virtual Machines

- Username: `administator`
- Password: `c0nygre`

Agenda

- Morning
  - Worked example of React Redux app using functional components
  - React functional Components and Redux
  - using hooks for props and events
  - when class-based components make sense
- Afternoon
  - very briefly: React Redux and the problems that need addressing
  - scope, scale and code simplicity
  - The React Redux toolkit
  - what problems it solves
  - a worked example using the toolkit

React State – let me count the ways

- You may end up having to work with multiple ways of writing Redux:
  - Flux
  - Reflux
  - Classical Redux
  - Recent Redux with connect()
  - New Redux with hooks like useSelector()
  - New-new Redux with Redux Toolkit

React Redux with Functional Components and Hooks

- Choose somewhere to work (e.g. create a folder)
  - Make sure NodeJS is installed
- Start a project
  - npx create-react-app redux-fn-app
- Let it do it’s stuff
- (Change directory into this folder)
- When done, install redux etc.
  - npm install redux react-redux redux-thunk
- Go and delete the default content inside App.js
- Change directory into the new app
  - cd redux-fn-app
- Start the development server
  - npm start

React Redux Toolkit with Functional Components and Hooks

- Choose somewhere to work (e.g. create a folder)
- Start a project
  - npx create-react-app redux-toolkit-app –template redux
- Let it do it’s stuff
- Go and delete the default content inside App.js
- Change directory into the new app
  - cd redux-toolkit-app
- Start the development server
  - npm start

React Redux

- Redux offers a ‘single-source-of-truth’ state for the entire app
- Hooks offer cleaner code in functional components

Doing Immutable Updates

- Get used to the idea of treating everything as immutable
- We **alter** immutable things by destroying them and creating new ones
- React ‘reacts’ by rendering the changes (efficiently)

React Convention: Controlled Inputs

- Handle form fields with as controlled inputs
    - Form field value is stored in the component's state

Hooks

- Hooks are functions
- They hook into state and lifecycle features in function components
  - new in React 16.8+
  - https://reactjs.org/docs/hooks-intro.html
- Hooks work well with Redux
  - https://react-redux.js.org/api/hooks
- Hooks don’t work in classes
  - Use them instead of writing classes

Why Hooks:
Reuse stateful logic between components

- React had no mechanism to attach reusable behaviour to a component
  - E.g. connecting a component to a store
- Patterns like ’render props’ and ’higher-order components’ tend to make code overly complex (see React Dev Tools! For wrapper-hell)
- Hooks extract stateful logic from a component
  - can be tested independently
  - Easier to reuse components
- Hooks reuse stateful logic without affecting the component hierarchy
- https://reactjs.org/docs/hooks-intro.html

Rules of Hooks

- Hooks are JavaScript functions
    - They impose two rules
- Only call Hooks at the top level
    - Never call Hooks inside loops conditions or nested functions
- Only call Hooks from React function components
    - Never call Hooks from regular JavaScript functions

- https://reactjs.org/docs/hooks-overview.html#rules-of-hooks

The Order and Quantity of Hooks

- We cannot change the order or numbers of the hooks in each render
  - It must be the same in every render
  - Not designed for conditional hooks
  - No hooks in loops