Lecture 11: Performance

Jordan Hayashi

Previous Lecture

- Async `simpleRedux/`
- Redux Middleware
- `redux-persist`
- Container vs Presentational Components
- ESLint
- Prettier

Performance

- How quickly and efficiently something works
- Performance optimization is the process of making something work as efficiently as possible
- Performance optimization is a very wide field
  - Today we'll discuss optimizing on the JavaScript side of things
  - Mostly high-level, with examples

Trade-Offs

- Performance optimization usually comes at a complexity cost
  - In most cases, optimization is not worth the cost in complexity and maintainability
- Don’t over-optimize until a bottleneck is found
- How do we measure for bottlenecks?

Measuring Performance

- Be mindful of the environment setting of your application
- React Native Perf Monitor
  - Shows you the refresh rate on both the UI and JS threads
  - Anything below 60 means frames are being dropped
- Chrome Performance Profiler
  - Shows you a flame chart of all of your components
  - Only available in development mode

Common Inefficiencies

- Rerendering too often
- Unnecessarily changing props
- Unnecessary logic in mount/update

Rerendering Too Often

- Components will automatically rerender when they receive new props
  - Sometimes, a prop that isn't needed for the UI will change and cause an unnecessary rerender
- If you use redux, only subscribe to the part of state that is necessary
- `keys` in arrays/lists
- `shouldComponentUpdate()` and `React.PureComponent`
  - A `PureComponent` has a predefined `shouldComponentUpdate()` that does a shallow diff of props

Unnecessarily Changing Props

- Unnecessarily changing a value that is passed to a child could cause a rerender of the entire subtree
- If you have any object (or array, function, etc.) literals in your `render()` method, a new object will be created at each render
  - Use constants, methods, or properties on the class instance

Unnecessary Logic in Mount/Update

- Adding properties to class instance instead of methods on the class
  - Properties are created at each mount whereas methods are one time ever

Reminder: Trade-Offs

- Performance optimization usually comes at a complexity cost
  - In most cases, optimization is not worth the cost in complexity and maintainability
- Don’t over-optimize until a bottleneck is found

Animations

- Let’s add a progress bar to our Project 1 timer
- Animations require both the JS and UI threads
  - Sending messages over the bridge 10s of times per second is expensive
  - Blocking either thread impacts the UX
- We could implement the animation in native
  - Requires knowing Obj-C/Swift and Java
- What if we could declare the animation in JS and have it execute on the native thread?

Animated

- Allows us to declare a computation in JS and compute it on the native thread
  - JS thread no longer needs to compute anything
  - JS thread can be blocked and the animation will still run
- Cannot use native driver for layout props

https://facebook.github.io/react-native/docs/animated.html