React Fundamentals

Workshop

Topics we'll cover

- Bare-bones rendering
- Components & JSX
- Responsive layouts
- Development tools
- General design principles & the Redux philosophy
- Working with remote data
- Routing
- Optimizations
- Higher-order components & library patterns

What is React for?

- user interfaces (view-controllers)
    - portability
    - efficient updating (dom updates are bottleneck)
    - enforces a state management pattern (framework) that is simple and scalable

Alternatives:

- Web
    - Angular
        - does not enforce state management pattern (at a serious cost)
        - offers solutions beyond view updating (does everything)
    - jQuery / vanilla
        - views must be updated manually, with no abstraction layer
        - more expensive rendering, have to code render logic, and gets messy

coding session #1

react-fun/1_bare-bones

Design principles

- Central state (single source of truth)
- Events cause the state to transition from one state to the next
- UI is updated with every state update, and each subcomponent acts independently, reading and reflecting some subset of the state.

- This is the pattern that a React component uses internally.
- The Redux framework extends this logic to the App as a whole.
  - Allows for functional app logic (independent and testable transitions and)

Development tools

- Compilation (Babel)
    - Allow import syntax
    - Transpile new ECMAScript syntax for backwards compat
    - Add static types (TypeScript / Flow)
- Module bundling (Webpack)
    - Generate 1 file containing all JS code (or a number of "chunks")
    - Generate CSS files as defined in styling language (LESS, SASS, JSS)
- File serving
    - Automatic reloading
- Testing
    - unit / integration / E2E
- Build optimizations
    - minify, uglify

guided tour

create-react-app

coding session #2

react-fun/2_simple-ui

coding session #3

react-fun/3_complete-app