React: The Big Picture

Why React

- A library, and less opinionated than 'frameworks'
- Versatility is improved because the 'renderer' is separate to React itself.
  - Incl server-side (Next.js, Gatsby, Phenomic) and thinkgs like render to PDF
- https://codesandbox.io
  - React sandbox
- Code mod
  - When breaking changes occur in React, Facebook provide a 'code mod'. This is a CLI tool you can point at your code to automatically make changes.
- Huge availability of mature and open source components
  - See the awesome React list for examples: https://github.com/enaqx/awesome-react
- Some key parts of the ecosystem
  - React Router
  - Redux
  - Mobx
  - Jest
  - GraphQL
  - Next.js
- React + ReactDom = 35K when gzipped and minified
  - Inferno and Preact are two smaller versions of React, if you need one.
- Testability
  - Reacts functional components are 'pure' (given the same input, you get the same output).
    - Jest is the most popular React testing solution

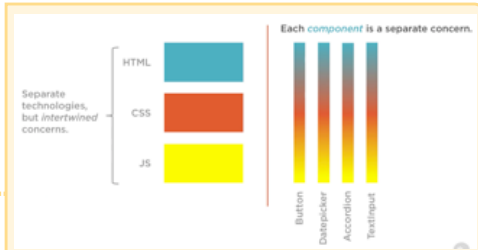
Tradeoffs

- Key tradeoffs:

	React	Angular
Components	✓	✓
Testing	Jest, Mocha	✓
HTTP library	Fetch, Axios	✓
Routing	React Router	✓
I18n	react-intl	✓
Animation	react-motion	✓
Form validation	react-forms	✓
CLI	create-react-app	angular-cli

Library (not framework)

- Explicit (not concise)
  - React uses one-way binding, so you have to handle changes in a handler (Angular and others prefer two-way binding)
- JavaScript centric (not template centric)
  - Whereas Angular and others put JS in HTML, React puts HTML in JS
- Separation of concerns - Single files for components (not separate templates)
  - React is not MVC



React mixes logic and markup - but thinks of separation of concerns as separating components. Each component is a separate concern.

- React is not the web component standard (which does exist)
  - The Web Component standard exists - but browser support is spotty and browser vendors don't seem interested in it
    - Templates
    - Custom elements
    - Shadow DOM
    - Imports
  - Everything you can do in Web Components can be done in React (or Angular etc.)
    - Template == JSX
    - Custom elements == components
    - Shadow DOM == CSS modules
    - Imports == one component per file
- Corporate backing means React is driven by Facebook's needs. But this has benefits as compared to purely 'community' driven projects

Why not React

- Concerns
  - HTML and JSX differ
    - Differences
      - htmlFor
      - className
      - Styles in JSON format
      - Comments in JavaScript format
    - There are conversion tools (including a NPM package)
  - Build step is required
    - The most popular transpires (Babel and Typescript) work well with React
    - There are also boilerplates (like create-react-app) which configure everything automatically
  - Version conflicts
    - You can't run two different versions of React together on the same page.
    - You also need to ensure your dependencies (like React router) are compatible with the version of React you're using
  - Out of date information online
    - Several features have been extracted from React core and exist separately now
    - When in doubt, check the React documentation
  - Decision fatigue
    - Key decisions for a team to make upfront:
      - What development environment
        - Most people use create-react-app
      - To use classes or functions
        - Most people prefer functions these days.
      - How to handle types
        - Three popular ways: prototypes, typescript or Flow
      - How to handle state
        - Popular ways: plain React, Flux, Redux or MobX
          - Redux is currently most popular
          - But remember: you often will not need a separate state management library. You can build powerful apps using plan React.
      - How to handle styling
        - Most React developers continue to use traditional styling approaches