Real-Life React

Tools & Techniques

Zürich ReactJS Meetup
April 27th, 2016
http://bit.ly/real-life-react

Agenda

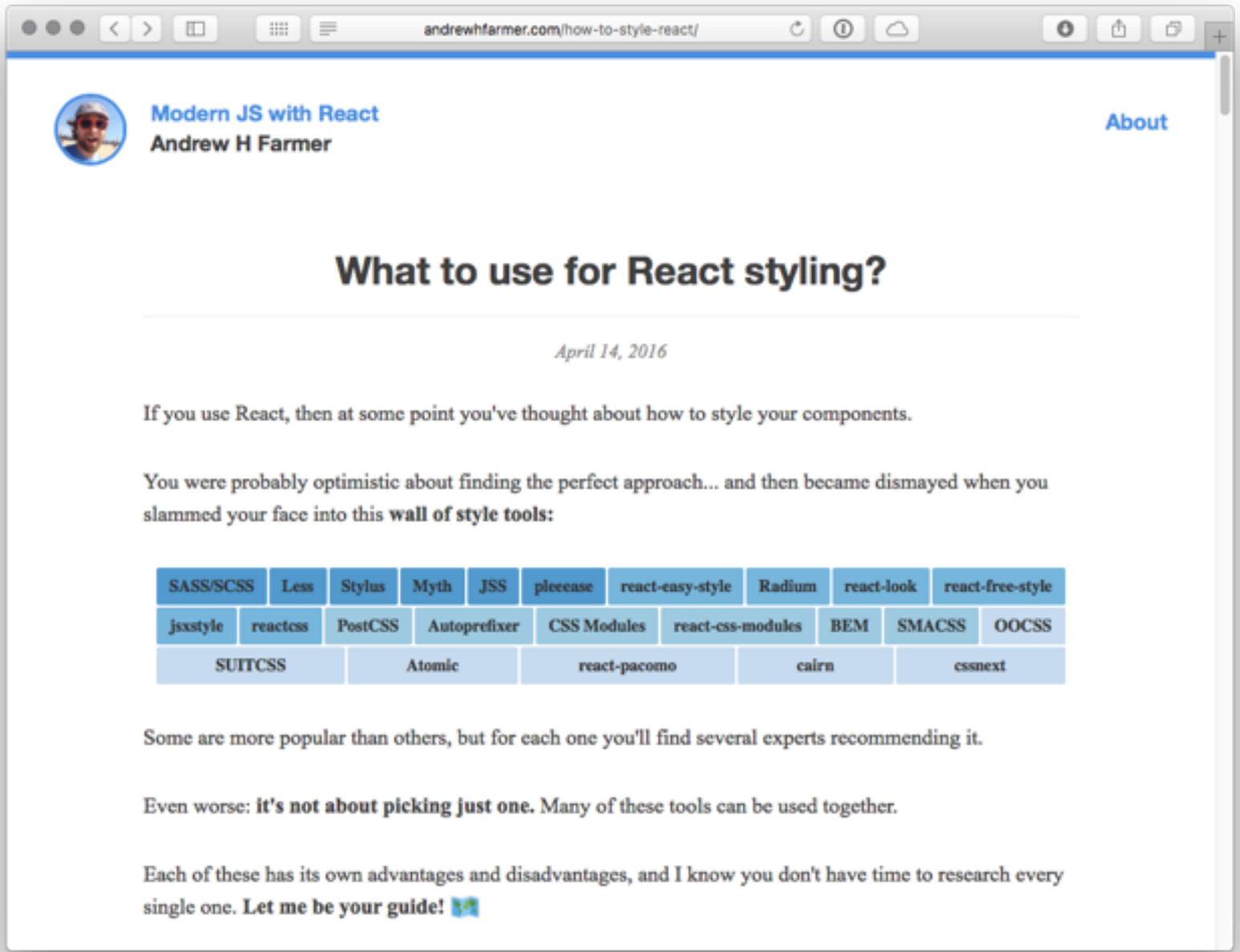
- **Styling** React components
  - Preprocessor: SCSS
  - Postprocessor: CSS Modules
  - Inline Style Helper: Radium
- **Documentation** for React components
  - Storybook
  - Catalog

The problem(s) with CSS

1. Global Namespace
2. Dependencies
3. Dead Code Elimination
4. Minification
5. Sharing Constants
6. Non-deterministic Resolution
7. Isolation

- https://speakerdeck.com/vjeux/react-css-in-js
- https://vimeo.com/116209150

Current solutions



Four Types of Tools

- Methodologies: disciplined use of CSS (e.g. BEM)
- Preprocessors
- Postprocessors
- Inline Style Helpers

Demo projects

```
dist/
  bundle.js
  index.html
src/
  index.jsx
.babelrc
package.json
webpack.config.js
```

Preprocessors

SCSS

- Like CSS, but with more features (Variables, Nesting, Functions…)
- Needs preprocessor (sass-loader for webpack), creates external stylesheets
- Makes CSS code easier to write and maintain, but used in components like CSS
- Not React-specific at all

Postprocessors

CSS Modules

- Stylesheet colocated with component
- Generates unique class names to avoid collisions
- Supports stylesheet composition
- Supported by css-loader for webpack
- ReactiveConf: The case for CSS modules https://www.youtube.com/watch?v=zR1lOuyQEt8

Inline Style Helpers

```
<div style={{color: "red"}}>
  Lorem ipsum dolor
</div>
```

- Joins form and function within component
- Explicit style attributes of DOM elements
- CSS can be handled like any other data
- Easy dynamic styling

Radium

- Adds features only available in CSS
  - Pseudo-selectors (e.g. `a:hover`)
  - Media queries (e.g. `@media print`)
- Automatic vendor prefixing
- Applied by wrapping component before instantiation (e.g. `Radium(App)`)