I. Introduce myself

- Self taught software developer with formal compsci education
- Worked on various large scale apps in React

II. Background

A. Just like JavaScript, there are many bad things about styles in React with the way they are handed to us. B. Go back to title -> fade out "Styles in React" C.Explain that just like JavaScript: when we first get started with React, we are given a lot of things that we should stray away from. D. But there are solutions and approaches to minimize these, and that's why I'm here today.

III. JavaScript Styles in React: The Bad Parts

A. Inline-Styles

1. Rational Basis for using them?
    - simple, modular, can be distributed without worry
      of alteration (inline styles take precedence over CSS)
      and can control
2. Problems
    - has own diffing tree in React
    - passing style as props to augment other style can
      easily create unwanted re-renders in large app;
      does not by default play well with immutability and
      a lot of care must be taken to avoid it
    - Many libraries depend on CSS, so we end up needing to
      go between both "className" and "style", as well as
      the mental overhead of tracking this.
    - Debugging : show screen;
      -> overall crazy
      -> gives no clues semantically as to what they are without
      otherwise manually assigning class/domIds

B. CSS

1. Rational Basis for using them?
    - They are fast (with caveats to talk about later)
    - Separate the appearance of a component from a component's
      layout/behavior
    - Well known; it's been this way forever
2. Problems
    - Lots of styles in the CSSOM at once slow down the browser
    - Workflow issues: need to switch syntax,

- Portability issues
- Page load speed decreases
- If using SASS/SCSS: page load speed is slower
- Not a dynamic style solution

IV. JavaScript Styles in React : Approaches to Traditional Methods

A. Inline-Styles

- Create style namespaces; S, SBase examples
- Use Radium for media queries
- Use fast-memoize for dynamic sheets that depend on variables
  B. CSS
- Use media queries to minimize render time
- Inline CSS files into HTML head (via server-side processing or simply putting it there), or minimize the external CSS stylesheets to just a few to prevent render-blocking content requests.
  C. These methods work, but there are several problems
- We need to switch between both to maximize efficiency
- They retain a lot of the issues listed earlier

V. A More Modern Approach: JSS

A. Takes the best of both worlds;

- Uses cache-ing to inject style tags directly into your browser's CSSOM opportunistically and creates rules only as needed; generates classNames, so its easily interoperable with existing CSS
- Briefly explain DOM render pipeline & how opportunistically attaching/detaching styles is good.
  B. Supports CSS pseudoselectors:
- [show example of hover and class nesting]
  C. Extensible:
- plugins : e.g. global classes
- can write your own plugins to do things such as name classNames generated to relate to your component and logic

B. Has functional values, so it can calculate dynamic styles

C. There are others such as Aphrodite, but of most "CSSinJS" libraries, it ranks as fastest in speed tests

[follow by screencap of test from CSSinJS repo]

D. Has aphrodite-like API anyway

IV. And Beyond!

A. React JSS

- When a component is rendered, it opportunistically attaches a stylesheet to the DOM with info about your component + [screenshot]
    - As a corollary, it detaches as well
    - Function values automatically retrieve props -- very powerful and simple interface
    - This is how I created these slides without going insane in a short amount of time

B. Examples

C. Get Started - react-redux-gulp /shameless plug

End. Questions?