<https://youtu.be/I3T17kupyv0>

## Styling: Stylesheet vs. Inline

Earlier you were introduced to React Native’s StyleSheet API for creating “stylesheets” out of JavaScript objects. At first this approach may seem a little strange, but there are some reasons behind it. Primarily those reasons are code quality and performance. Let’s take a look at some comparisons in regards to code quality.

<View style={{

borderRadius: 4,

borderWidth: 0.5,

borderColor: '#d6d7da',

}}>

<Text style={[

{fontSize: 19, fontWeight: 'bold'},

props.isActive && { color: 'red' }

]}>

Welcome

</Text>

</View>

Above we have some JSX for a pretty simple UI. Notice, that even though this UI is rather simple, the styling of it makes it rather messy. This is perhaps the biggest benefit to the StyleSheet API: by moving styles away from the render function, the code becomes easier to read and understand. Not only that, but naming the styles is a good way to make components a little more declarative. With the StyleSheet API, we can change the code above to now look like this:

var styles = StyleSheet.create({

container: {

borderRadius: 4,

borderWidth: 0.5,

borderColor: '#d6d7da',

},

title: {

fontSize: 19,

fontWeight: 'bold',

},

activeTitle: {

color: 'red',

},

});

<View style={styles.container}>

<Text style={[styles.title, props.isActive && styles.activeTitle]} />

</View>

On top of quality benefits, there are also performance benefits as well. Making a stylesheet from a style object makes it possible to refer to it by ID instead of creating a new style object every render.

## Media Queries

One thing you may have noticed is that React Native (and specifically the StyleSheet API) doesn’t support [media queries](https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries). The reason for this is because, for the most part, you can design responsive grids with flexbox which will bypass the need to use media queries. In the rare case where flexbox just won’t work for your specific needs, you can use the [Dimensions](https://facebook.github.io/react-native/docs/dimensions.html) API which we covered earlier to get similar results.

## CSS in JS Libraries

Styling in React is going through a renaissance period right now just as Flux did a few years ago (which eventually gave us Redux). There are many different styling libraries popping up and each has different tradeoffs.

Two of the most popular are [Glamorous](https://github.com/robinpowered/glamorous-native) and [Styled Components](https://github.com/styled-components/styled-components). The whole idea of both of these libraries is that styling is a primary concern of the component and because of that, should be coupled with the component itself.

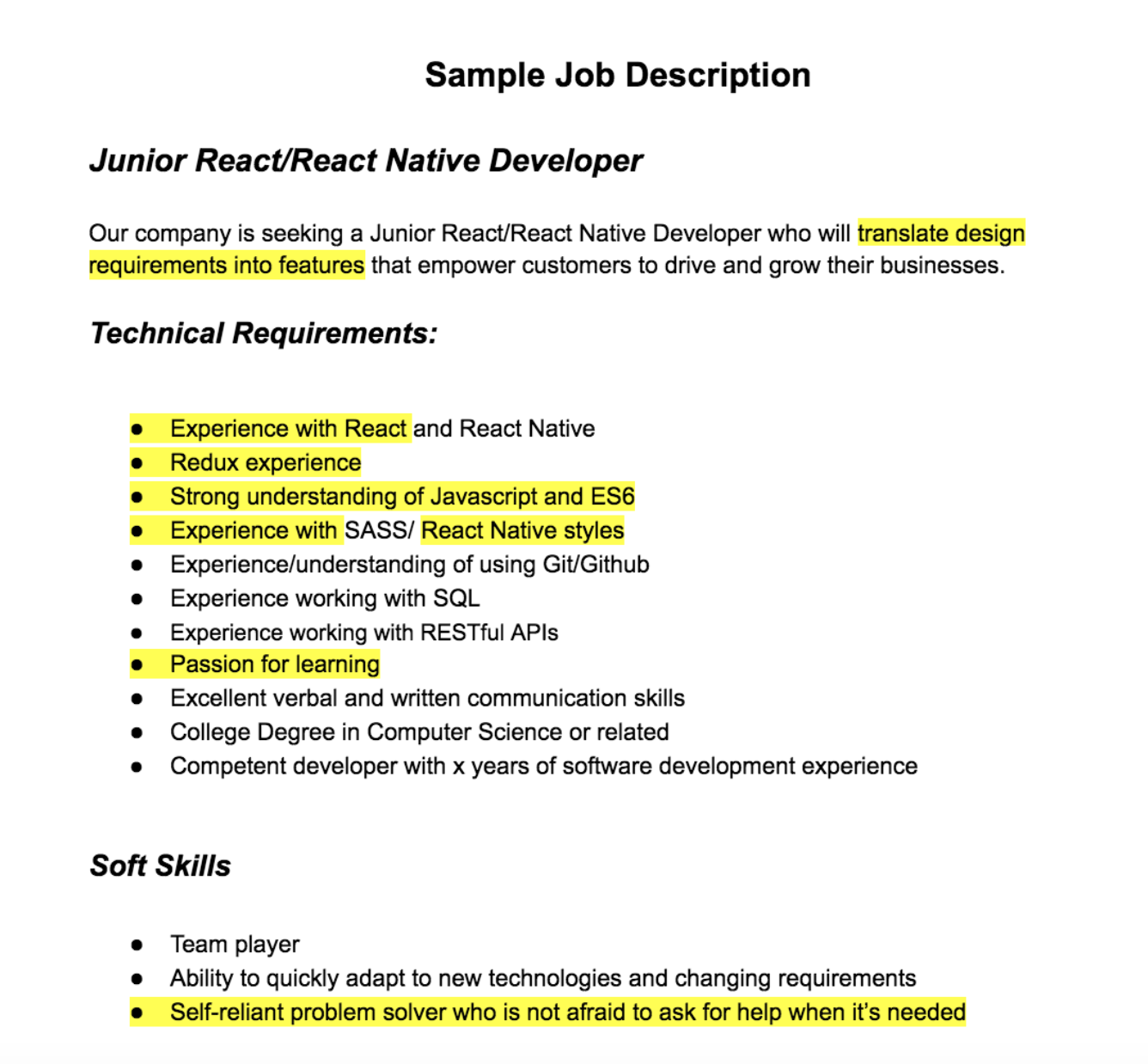
Let’s take a look at not only the Styled Components library, but also how you’d use it with React Native.

<https://youtu.be/XF_4MPpvRqs>

### Further Research

For further research on Styled Components, you can visit the [official documentation](https://www.styled-components.com/).

Here's how what you've accomplished in the program thus far maps to your career:

[[](https://classroom.udacity.com/nanodegrees/nd019/parts/197a09ba-b687-4f20-bc15-90a5fb6fad1a/modules/3c60017f-cf44-4904-b715-ffab69e84a92/lessons/aea80934-1f7d-41a7-b432-d4a1ce6bc3b1/concepts/f6b5262d-6265-4675-b3a2-13d67b63c1dc)](https://classroom.udacity.com/nanodegrees/nd019/parts/197a09ba-b687-4f20-bc15-90a5fb6fad1a/modules/3c60017f-cf44-4904-b715-ffab69e84a92/lessons/aea80934-1f7d-41a7-b432-d4a1ce6bc3b1/concepts/f6b5262d-6265-4675-b3a2-13d67b63c1dc)