**Component Calibration – React**

As outlined in the base styles documentation [link to base styles documentation], one can define all of the possible styles that can be applied variably to parts of a component, in order to give it its overall look and feel. However, to be extensible and flexible, components need to be able to modify their styling (e.g. box-shadows, geometry of borders, background colors) based on props and state, which means selectively applying different base styles according to the context presented at render-time.

In order to achieve this, one can define and implement a calibrateComponent() function which is responsible for interpreting which styles need to be applied based on props and state, and then creating stylechains that can be returned in a component’s render() method and applied to different parts of the JSX markup as appropriate.

The general execution path inside this function is as follows. At the top of the function, one can pull in any resources they might need from props and state. Then, they can compute the application of different styles based on this context by making determinations inline or by calling other helper functions when those determinations are more involved. Following this, stylechains can be calculated using [aphrodite](https://www.npmjs.com/package/aphrodite)’s css() function, minding the order in which styles are entered as arguments to the function since they are overridden from last to first (e.g. putting custom override styles provided from props last on the stylechains so that they override all preceding ones). Lastly, these stylechains are returned in an object payload to be applied to the JSX markup in the component’s render() method.

As an aside to the preceding discussion, it’s important to note that this component calibration function can be used to return anything else to the component that it might need at runtime other than just styles. For example, under certain conditions dictated by props and state, a component might be rendered as a button or as a non-interactable tag. This calculation and determination can be carried out by a helper function, and then the resulting component can be returned along with the calculated stylechains.

**Caveats**

At the time of writing this documentation, it appears that [aphrodite](https://www.npmjs.com/package/aphrodite)’s css() function does not accept styles that were generated as a result of multiple invocations of its StyleSheet.create() method. This means that any custom override styles that are passed in as props will need to be accounted for and processed during the single invocation of this stylesheet creation function that occurs in the component’s base style generation [link to base style generation documentation].

**Pseudocode Walkthrough**

These component calibration functions are typically stored in a separate file for helpers alongside the main component file that they support as part of a larger component directory structure. The pseudo code that one might follow would look like:

import {css} from ‘aphrodite’;

// -> Import other constants and resources you might need here.

// -> Pass in props, state, and calculated base styles.

const calibrateComponent = (props, state, styles) => {

// -> Here, an application’s theme is imported through a

// component’s props. More information on how this is

// accomplished, and theming in general, can be found

// in the theming documentation [link to theming

// documentation]

const {theme, raised, /\*other props you need \*/} = props;

const {/\*stuff you need from state\*/} = state;

// -> Dynamic style computations.

const computedBoxShadowStyle =

(raised) ? styles.componentRaised : null;

const computedGeometryStyle =

determineComponentCradleGeometry(props, state, styles);

// -> Construction of stylechains.

const componentStylechain = css(

styles.componentCradle,

computedBoxShadowStyle,

computedGeometryStyle,

styles.customCradleStyles,

);

// -> Returning stylechains and other resources back to calling

// component.

return {componentStylechain};

};

export default calibrateComponent;