# Style Guide -- Fin Print -- CMPS 116

*The following is the current -tentative - style guide for CMPS 116 Fin Print team. It will be updated as style develops. The style guide is not concrete as exceptions may need to be made with due diligence.*

*Tl;dr{ In short, the style guide can be condensed in its entirety to match with the core of our definition of done in it that the bottom line is that 'the client side site must not crash'.}*

----------------------------------------------------------------------

**File Structure**

Our stack is made with create-react-app which bootstraps the project and sets up a file structure that we follow pretty closely. In short all the code that we touch is in the *src/* directory. Components that reference pages go with in *src/components/pages*. Components that are constants can be outside the components dir but still within src. App.js is our home router, there will not be any editing on more conventional pages such as index.html/js.

**React Components, Inheritance, State-Management:**

Our code is heavily influenced by the book "The Road to React" by Robin Wieruch. Wieruch released a book on Firebase as well that is used heavily in the *SignUp* component and is sited in the component due to the amount of his code that we used.

The style that we followed is a system with the end goal of having components have as little repetitive state management code as possible by using higher order components to lift the state up and then pass that state to the components. Components are encouraged to be stateless whenever possible and state can be passed via the prop. Examples of these are our *App* component which is written as an arrow function as well as our *Header.* Stateful components are written in the standard render method and typically pass state to other components.

Another aspect of Wieruch's lightweight style is to 'import what you need'. For example Firebase is a huge library and we don't want components importing all of Firebase's methods to keep a component lightweight and to avoid unintentionally using a keyword. As a result we export specific functions from a Firebase directory and will give those functions to what needs it ( ie if a component needs Firebase.auth() methods they get passed the *auth* component but if they only need Firebase.database() they get the *db* component).

If an immutable type is recognized, *const* should be called.

**Javascript, JSX**

Use of JSX arrow function syntax should be used in lieu of .bind(this) as it is the new acceptable syntax.

Inline CSS should not be used unless it is really minor and there is good reason to do so. Inline HTML is allowed/supported in JSX. If of course a lot of js is interacting between small amounts of HTML, it may be best to extract the js so that both the js and HTML becomes more readable (ex *SignUpForm* ).

Import Statements use relative paths.

There currently do not exist any javascript indentation (et al) style rules in place.

**HTML, CSS**

We currently use SCSS which takes CSS and compresses it all on one line. As a result, unless there is good reason to do so, all CSS should go within the default SCSS file. ABSOLUTELY NO EDITING CSS WITHIN THE CSS FILE.

If generic tags are referenced with SCSS is <h> and <p>, then the entire class or component that the SCSS is stylizing should be wrapped in a div to prevent styles bleeding over to different pages.

Use JSX className= instead of alternative javascript ways to name divs for stylizing. This rule should almost never be broken on master.

**NODE**

All developers keep Node up to date. When there is an NPM install it must be added to the readme on our github so other developers know they need a dependency.