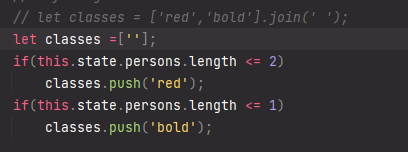
* **Problem 1**: how to change styles dynamically? Whatever we write inside the 2nd bracket is a javascript code. So we can simply select the style.background for example and change its value and put logic there.
* **Problem 2**: how to change class dynamically? We can assign a variable with class names inside the render method. We can put logic behind it as well. So everytime the component renders it will change classname according to the logic.



Note: in this example, the variable is an array. The className will expect a string. So we can join the array with a space.



**Radium library**

Using style inline has its advantage as the style will be applied to that component only. The downside is we can’t use pseudo selectors here.

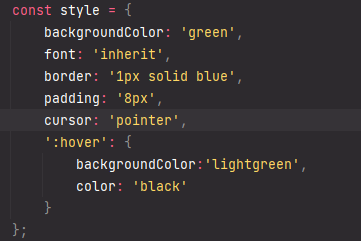
The alternative is using styling in CSS files, but that would affect all our components.

To solve this issue, we use a package called radium. Steps:

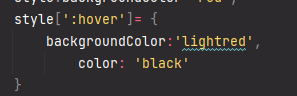
1. Install/import radium
2. Wrap the export with radium. This is called a higher order component.



1. After that we can use pseudo selectors.
2. The syntax is a bit weird though, we just need to add the pseudo selector as a string.



1. And when accessing them via style variable, we access them like this



**Media query using radium:** Here same issue again, we could do media query in css file. But in case we want to do inside component inline. To be able to achieve this inline, we need to do one extra thing, import StyleRoot. Then we need to wrap the app component div with this StyleRoot. Only then we can use the advance css features like media query



**Style Component Library: Check documentation for examples.**

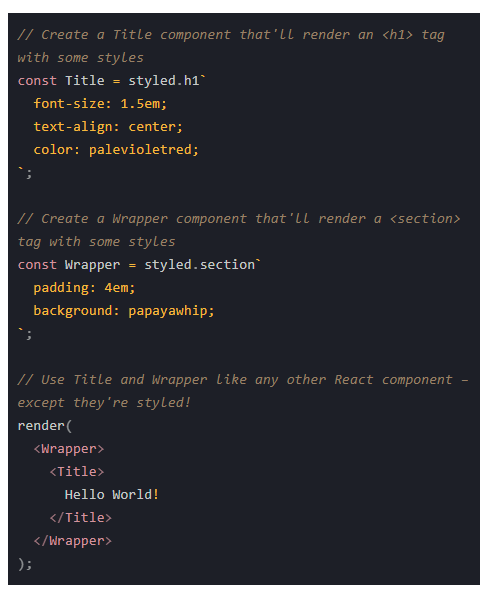
****

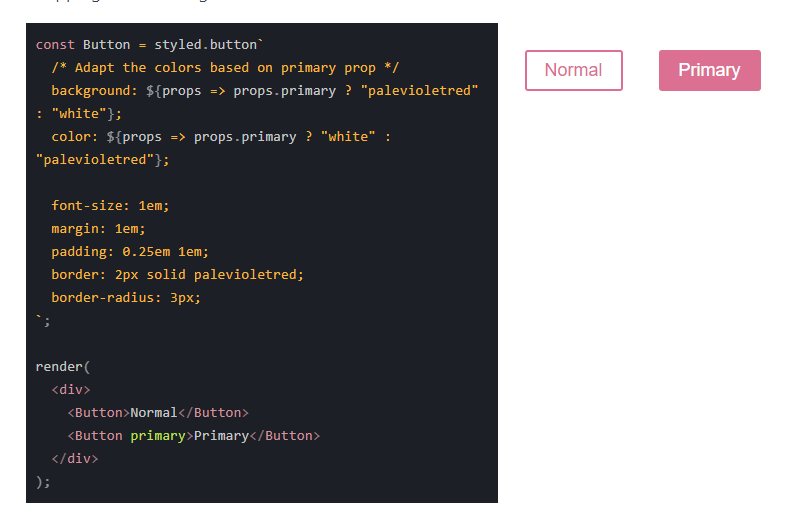
****

This styled object has a method for every html element. So, we can define styles inline inside the element. These methods return a valid react component. It assigns a random unique class name from the library.

**IMPORTANT** Note: 1) use capital letter constants 2) CSS syntax is regular not JSX

Some examples are:





Even though both libraries are good, they bloat up the js files with extra css code inside. There is another way of scoping css classes to a particular module. This concept is CSS modules. [at this point I find learning it unnecessary, so I’ll keep it for later]