React Fundamentals

Styling Components

Peter Kassenaar –
info@kassenaar.com

Styling components

Creating local or global styles for your application

CSS styling in React apps

- Components can receive styles in different ways:

1. From global imported styles
   - Like Bootstrap, Materialize, Tailwind CSS, etc.
   - Like we have done in the previous projects
2. From inline styles
   - Use CSS-in-JavaScript technique
3. From CSS modules
   - Create `.module.css` files for your styles

1. Global styles

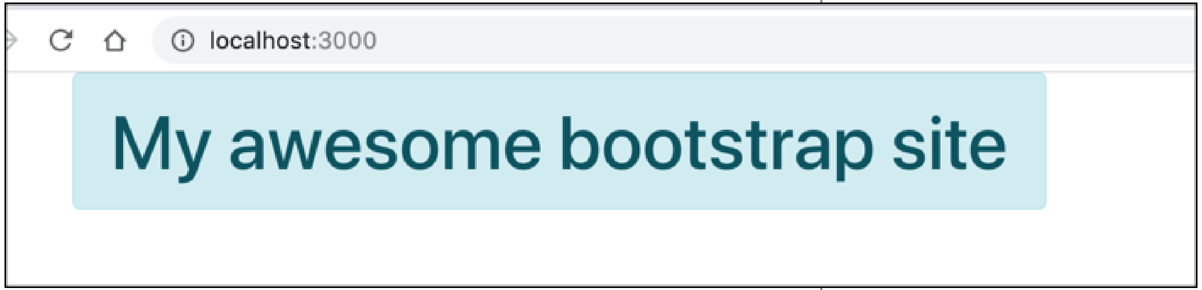
- Libraries: Just import the CSS in `../src/index.js`, like we have done before
  - Bootstrap, Tailwind CSS, Foundation, Framework7, etc.
- Custom global CSS
  - Place in `index.css`, import also
  - Can also be done in `App.js/App.css`
- Pro: styles are automatically available, everywhere
- Con: possible naming conflicts, too many styles in global scope

Example global styles – best practices

```
// index.js - import global styling

import 'bootstrap/dist/css/bootstrap.min.css'

import './index.css';
```

```
// App.js
function App() {
  return (
    <div className="container">
      <div className="row">
        <col-6>
          <h1 className="alert alert-info">
               My awesome bootstrap site

          </h1>
        </col-6>
      </div>
    </div>
  );
}
```

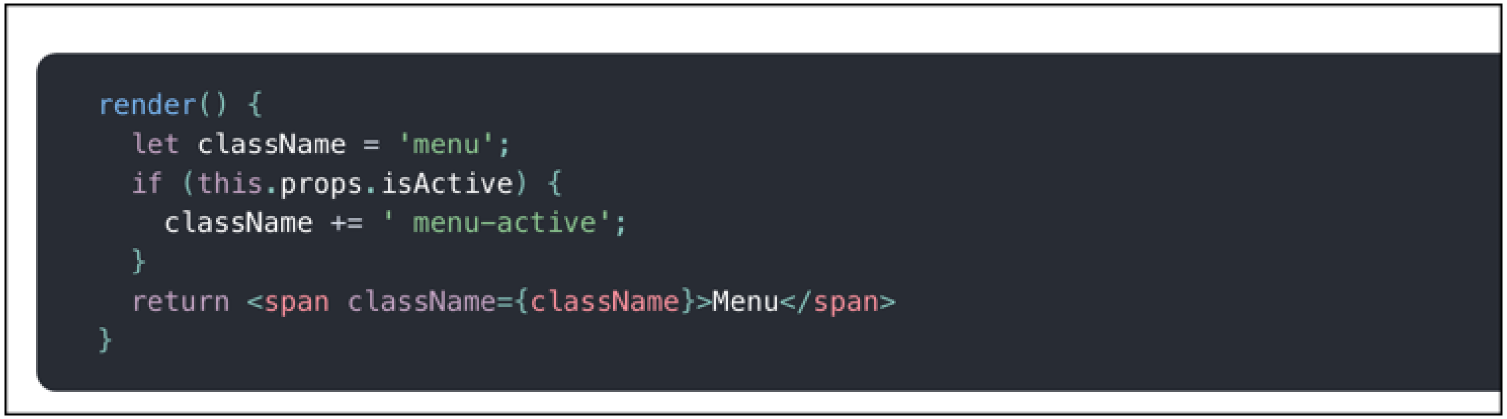
Global styles available in all components



My awesome bootstrap site

Composing class names

It is common for the `className` property to depend on the component `props` or `state`:



```
render() {
  let className = 'menu';
  if (this.props.isActive) {
    className += ' menu-active';
  }
  return <span className={className}>Menu</span>
}
```

This way you compose the exact styles of (global) class names that the component requires.

It responds automatically to changes in `props` or `state`.

2. Inline styles

"In React, inline styles are not specified as a string. Instead they are specified with an object whose key is the camelCased version of the style name, and whose value is the style's CSS value"

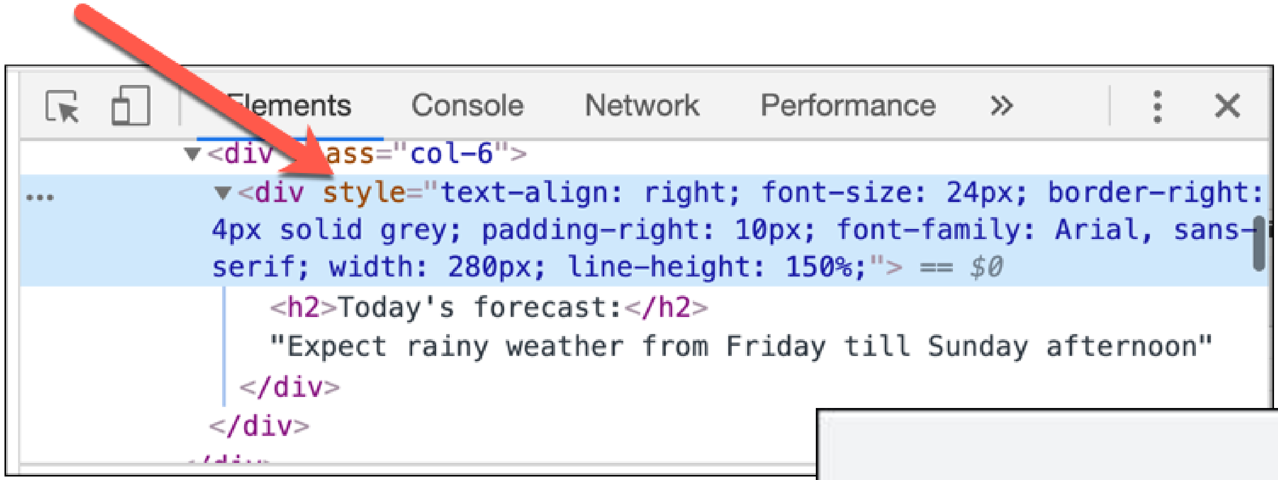
https://zhenyong.github.io/react/tips/inline-styles.html

Inline styles example

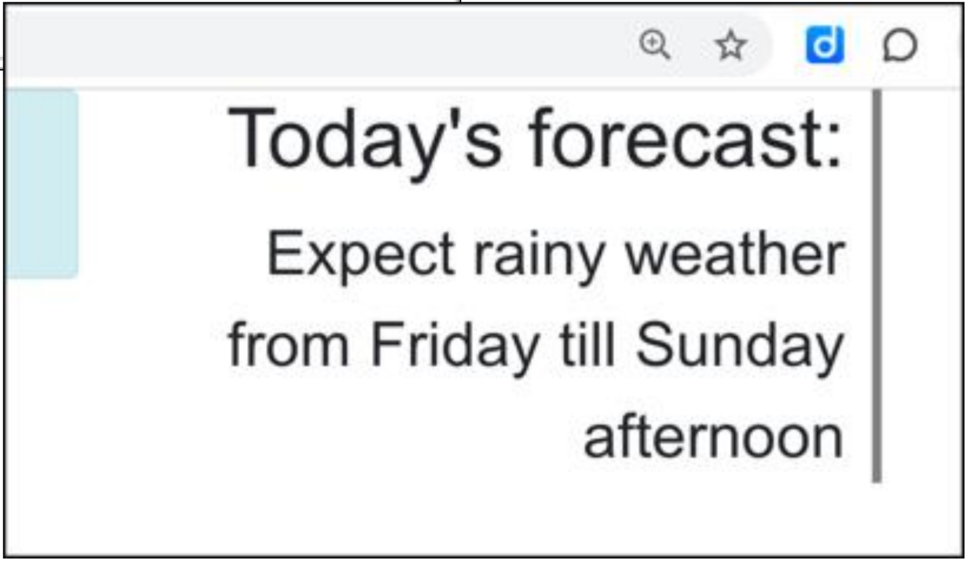
```
const headline = {
   textAlign: 'right',
   fontSize: '24px',
   borderRight: '4px solid grey',
   paddingRight:'10px',
   fontFamily: 'Arial, sans-serif',
   width: '280px',
   lineHeight: '150%'
};
```

```
<div style={headline}>
   <h2>Today's forecast:</h2>
   Expect rainy weather from Friday till Sunday afternoon
</div>
```

Transformed in the DOM



```
▼<div class="col-6">
  ▼<div style="text-align: right; font-size: 24px; border-right: 4px solid grey; padding-right: 10px; font-family: Arial, sans-serif; width: 280px; line-height: 150%;"> == $0
      <h2>Today's forecast:</h2>
      "Expect rainy weather from Friday till Sunday afternoon"
    </div>
  </div>
```



Today's forecast:

Expect rainy weather from Friday till Sunday afternoon

Defining inline, inline styles

The value of the `style` attribute has to be an object, so double curly braces:

```
<div style={headline}>
   <h2>Today's forecast:</h2>
   Expect <span style={{fontStyle: 'italic'}}>rainy weather</span>
   from Friday till Sunday afternoon
</div>
```

Verdict: try to avoid this. It's ugly.

No <style> block in components

```
<div style={headline}>
   <h2>Today's forecast:</h2>
   …
   <style>
      .boldText{
        font-variant: 'bold’, // invalid
        fontVariant: 'bold’    // also invalid
     }
   </style>
</div>
```

Invalid!

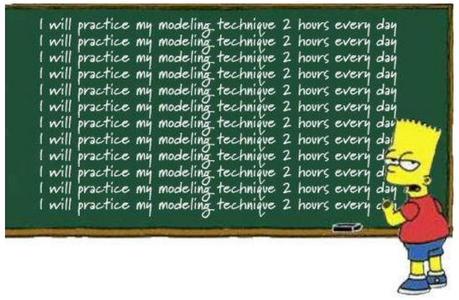
Will not compile

Verdict Inline Styles

- Pro
  - enclosed/scoped styles
  - No chance of naming conflicts
- Con
  - You might find yourself repeating styles in different components
  - Quickly leads to bloated code and less SoC
  - Less convenient `camelCasing` for style properties
  - May harm performance of rendering the DOM

Workshop

- Create a new component.
- Define some constants as inline styles in this component, for instance
  - `.warning, .info and .rejected.`
  - Give them some properties
- Use the classes in the component
- Example `../300-styling-components`
  - `../components/InlineStyles.js`


I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every

3. CSS Modules

- If you want to achieve the same goal as *scoped styles* in Vue or Angular, use CSS Modules
- The styles are only available in components that import them
- The compiler adds random hashes to the style names
- Styles are in separate `.css` files
- Styles have the `.module.css` extension

CSS Modules – structure

```
/* cssModules.module.css */
.warning {
    background-color: orangered;
    color: white;
    font-weight: bold;
    border-radius: 4px;
    padding: 6px;
}
…
```

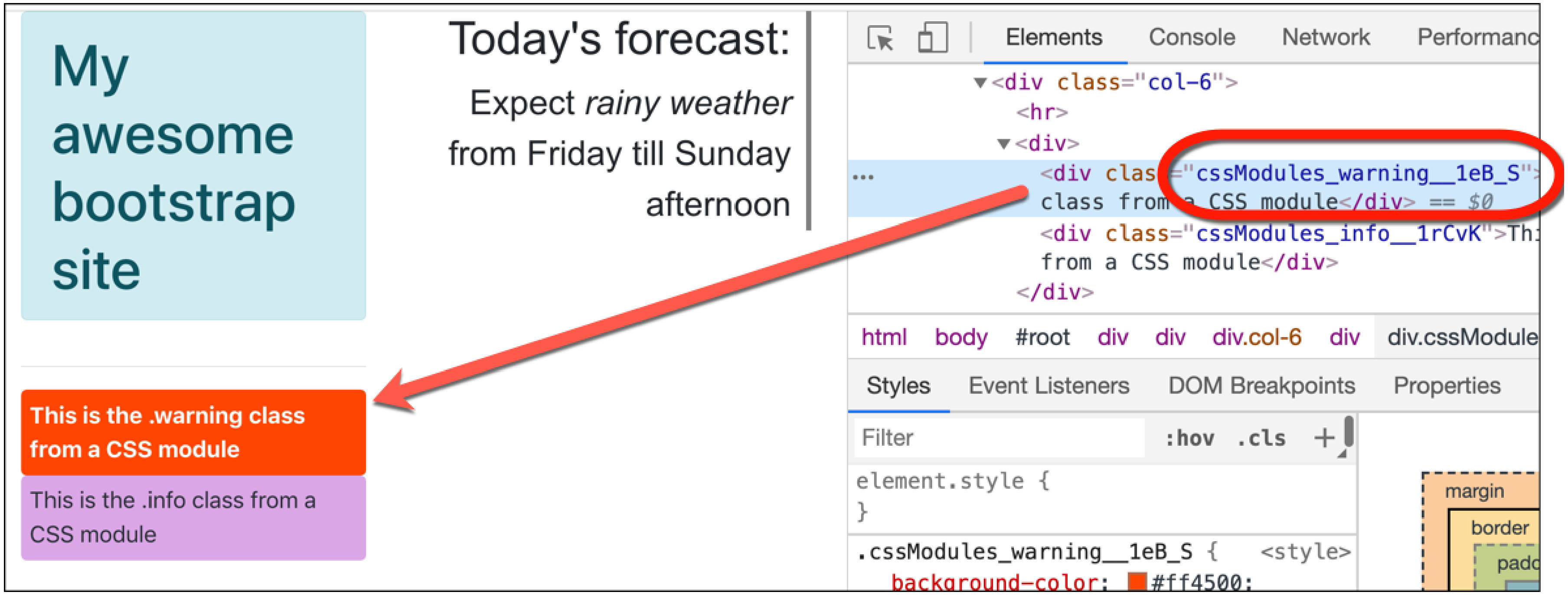
1. Separate `.module.css` file

```
import React, {Component} from 'react';
import styles from './cssModules.module.css'

class CssModules extends Component {
   render() {
      return (
         <div>
            <div className={styles.warning}>
               This is the .warning class from a CSS module
            </div>
            …
         </div>
      );
   }
}

export default CssModules;
```

2. Import in component

3 Use in component

CSS Modules in the UI/DOM



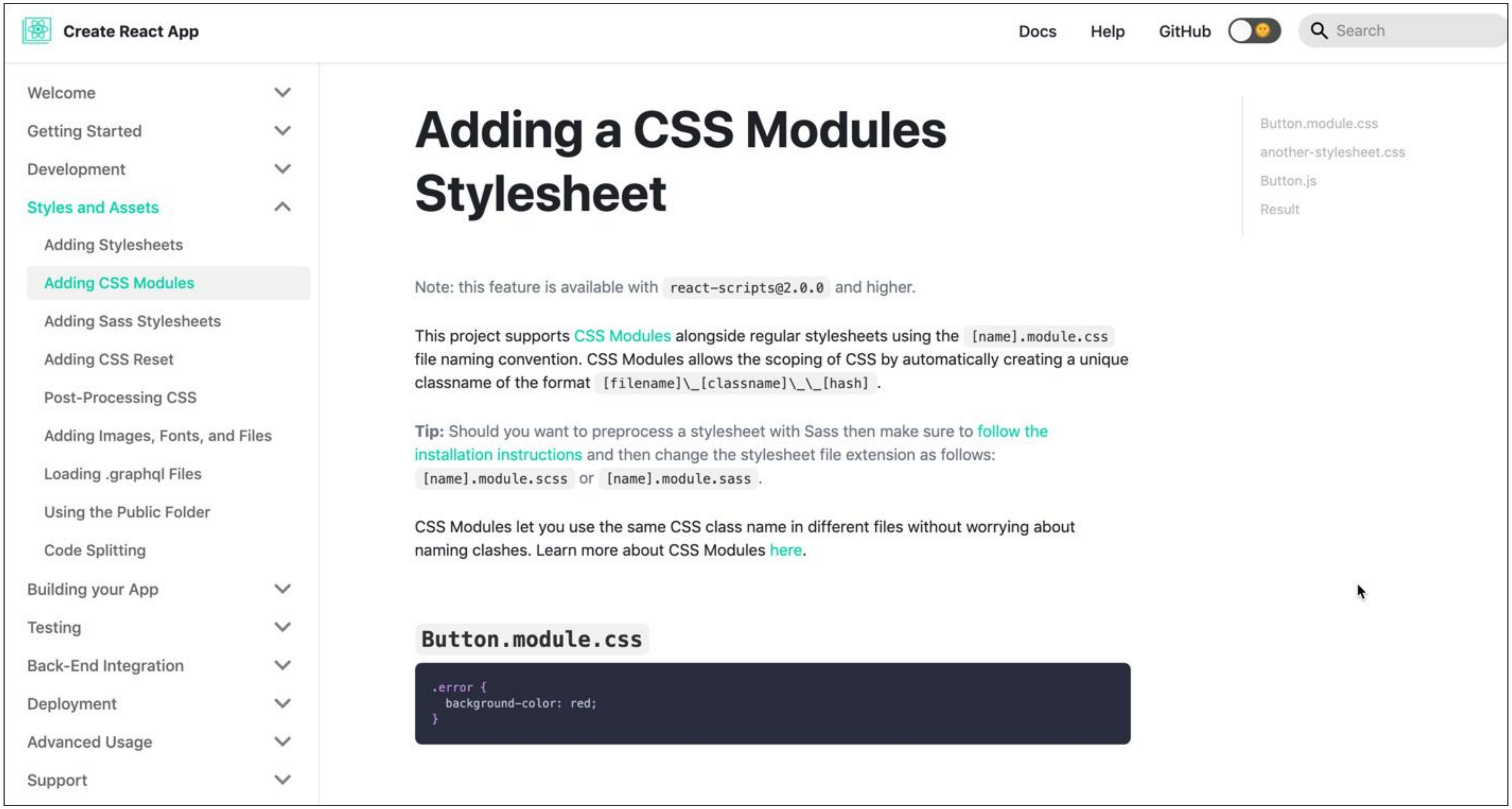
```
▼<div class="col-6">
   <hr>
  ▼<div>
     <div class="cssModules_warning__1eB_S">
     class from a CSS module</div> == $0
     <div class="cssModules_info__1rCvK">Th
     from a CSS module</div>
   </div>
```

```
element.style {
}
.cssModules_warning__1eB_S {    <style>
  background-color: ■#ff4500;
```

A random hash is added to the style name,

so no naming conflicts with other components

More info on CSS Modules



Adding a CSS Modules Stylesheet

Note: this feature is available with `react-scripts@2.0.0` and higher.

This project supports CSS Modules alongside regular stylesheets using the `[name].module.css` file naming convention. CSS Modules allows the scoping of CSS by automatically creating a unique classname of the format `[filename]\_[classname]\_\_[hash]`.

Tip: Should you want to preprocess a stylesheet with Sass then make sure to follow the installation instructions and then change the stylesheet file extension as follows: `[name].module.scss` or `[name].module.sass`.

CSS Modules let you use the same CSS class name in different files without worrying about naming clashes. Learn more about CSS Modules here.

`Button.module.css`

```
.error {
  background-color: red;
}
```

https://create-react-app.dev/docs/adding-a-css-modules-stylesheet/

Warning!

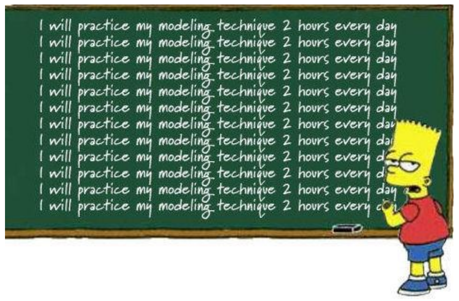
In CSS Modules, you can NOT have CSS classes with dashes in their name, like

```css
/*illegal! */
.info-important {
    background-color: #c1ab99;
}
```

(unless you `eject` your configuration and setup Webpack yourself. A hyphen is an illegal char in variable names, https://spectrum.chat/gatsby-js/general/css-modules-not-working-with-classnames-including-dashes~1a69e0d8-d8d8-4a03-b9aa-ad491e6c2093)

Workshop

- Create a new component.
- Create a new CSS module, again creating some classes in it
  - `.warning`, `.info` and `.rejected`.
- Import the module and use the classes in the component.
- Can you also import them in another component? If yes, how would you (re)structure your app?
- Example `../300-styling-components`
  - `../components/cssModules.js`



Using SASS

Creating `.sass` files and compiling/using them

Team React:

"Generally, we recommend that you don't reuse the same CSS classes across different components.

For example, instead of using a `.Button` CSS class in `<AcceptButton>` and `<RejectButton>` components, we recommend creating a `<Button>` component with its own `.Button` styles, that both `<AcceptButton>` and `<RejectButton>` can render"

"Following this rule often makes CSS preprocessors less useful, as features like mixins and nesting are replaced by component composition."

https://create-react-app.dev/docs/adding-a-sass-stylesheet

However, if you want/need to use Sass:

- Default available in CRA 2.0.0+
  - no more need to `eject` and configure `webpack-sass-loader` yourself
- First, install sass in your project:
  - `npm install sass [--save]`
- Rename your `.css` file to `.scss`.
- Use Sass variables, mixins, nesting, as usual
  - You can also `@import` other `.scss` files

Adding .scss files

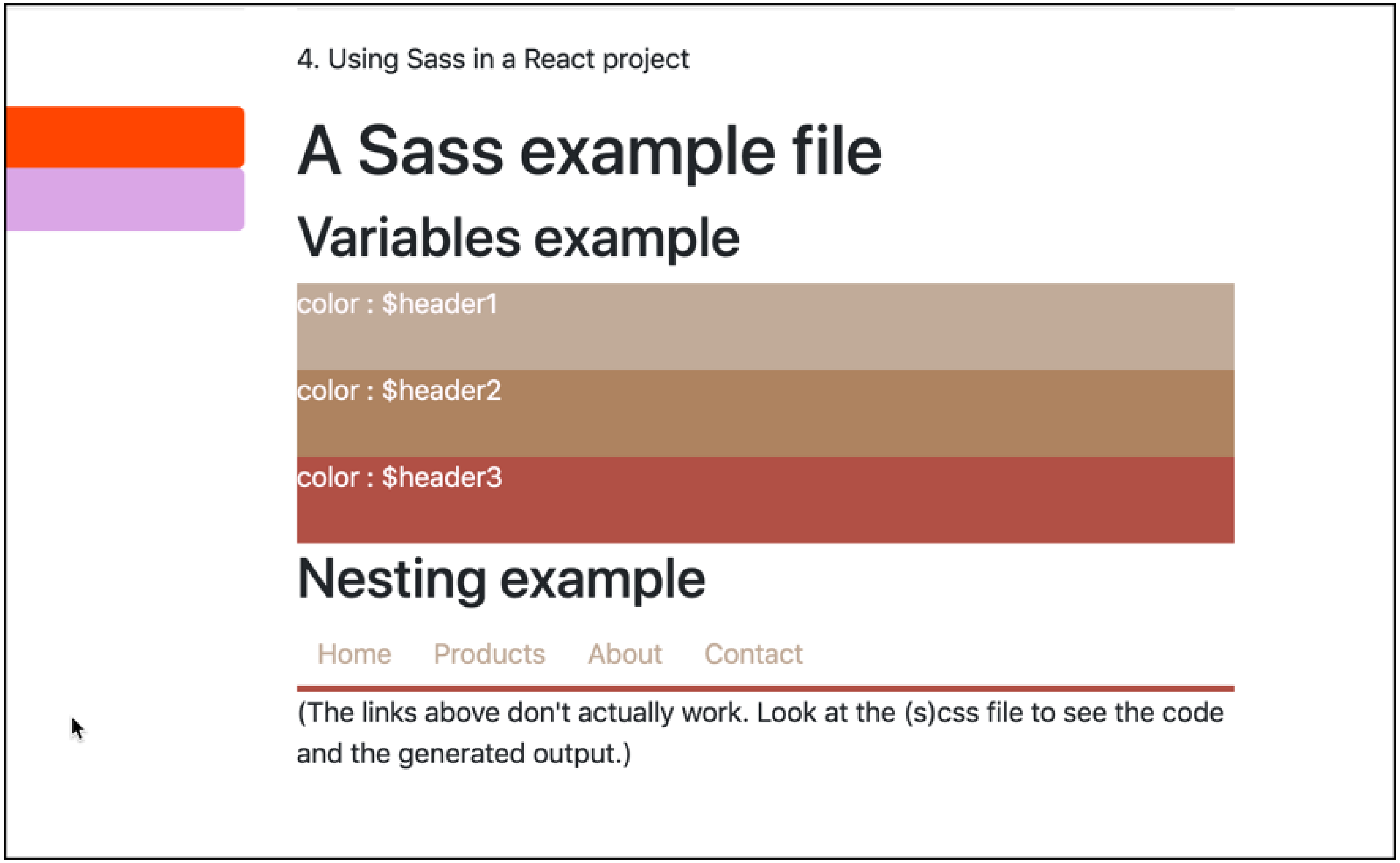
```
/* SassComponent.scss - a Sass file … */

// 1. Variables for the different header colors
$header1: #c1ab99;
$header2: #ae8360;
$header3: #b15145;
$forecolor: #fff9ff;

// Default styles
.header {
  width: 100%;
  height: 50px;
}
//Use the variables
.header1 {
  background-color: $he
  color: $forecolor
}

…
```

Import .scss file

```
import './SassComponent.scss'

class SassComponent extends Component {
   render() {
      return (
         <div>
            <p>4. Using Sass in a React pr
            <h1>A Sass example file</h1>
            {/*1. Header/variables example*/}
            <h2>Variables example</h2>
            <div className="header header1">color : $header1</div>
            <div className="header header2">color : $header2</div>
            <div className="header header3">color : $header3</div>
          )
…
```

Use className as usual

Result



4. Using Sass in a React project

A Sass example file

Variables example

color : $header1

color : $header2

color : $header3

Nesting example

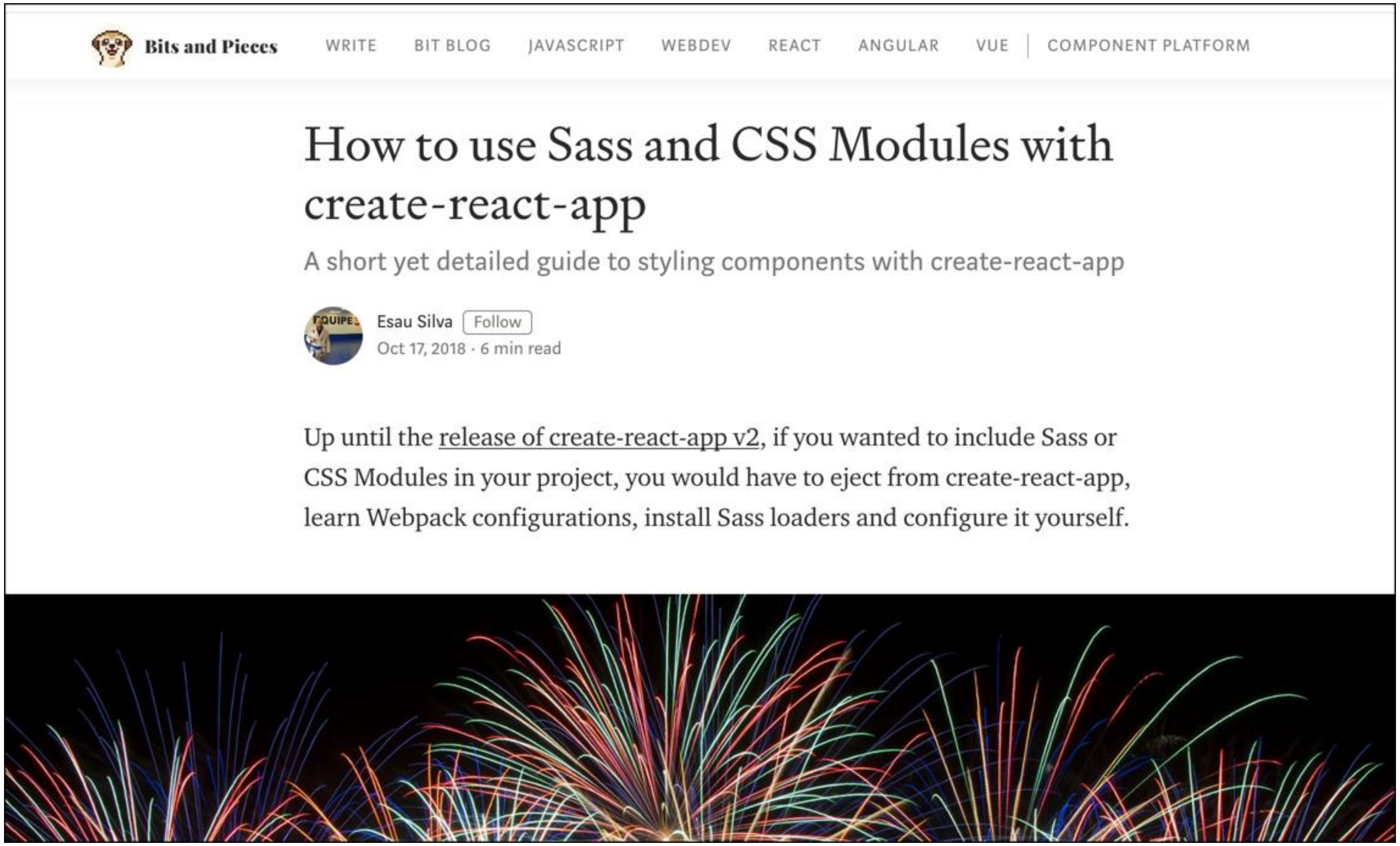
(The links above don't actually work. Look at the (s)css file to see the code and the generated output.)

```
<SassComponent />
```

Sass CSS Modules

- You can also use Sass CSS modules (i.e. combine CSS modules with Sass).
- Naming convention: `myComponent.module.scss`
- Combine the two techniques as usual.
- Don’t forget to import the module and assign it to a (`styles`) variable:
  - `<h1 className={styles.heading}…</h1>`

More info on Sass/Modules



How to use Sass and CSS Modules with create-react-app

A short yet detailed guide to styling components with create-react-app

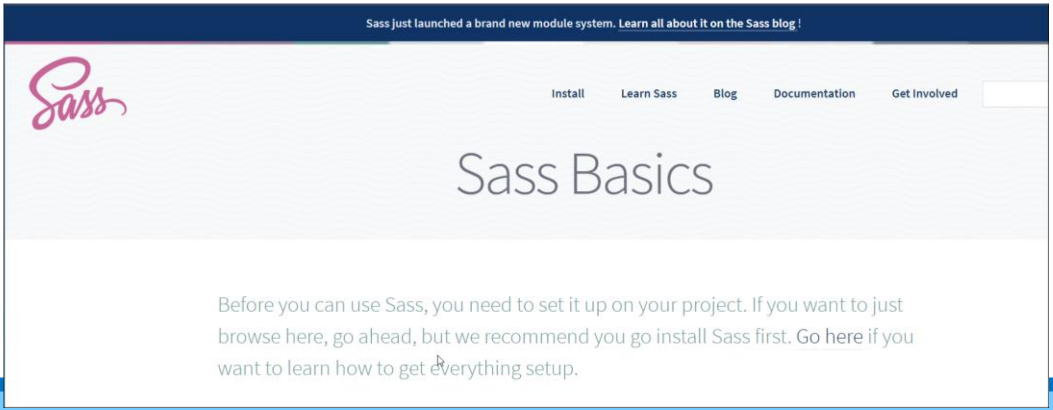
Esau Silva Follow
Oct 17, 2018 · 6 min read

Up until the release of create-react-app v2, if you wanted to include Sass or CSS Modules in your project, you would have to eject from create-react-app, learn Webpack configurations, install Sass loaders and configure it yourself.

https://blog.bitsrc.io/how-to-use-sass-and-css-modules-with-create-react-app-83fa8b805e5e

Workshop

- Create a new project (or use your own project) and install Sass to it. When working with the example project: add a component
- Create an `.scss` file with some variables and possibly nesting or mixins
- Examples: https://sass-lang.com/guide
- Example `../300-styling-components`
  - `../components/SassComponent`

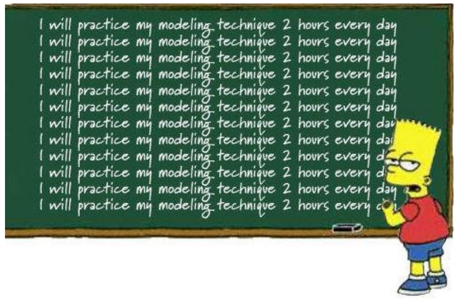

Sass just launched a brand new module system. Learn all about it on the Sass blog !

Sass

Install Learn Sass Blog Documentation Get Involved

Sass Basics

Before you can use Sass, you need to set it up on your project. If you want to just browse here, go ahead, but we recommend you go install Sass first. Go here if you want to learn how to get everything setup.



React UI libraries

Using React-optimized User Interface libraries

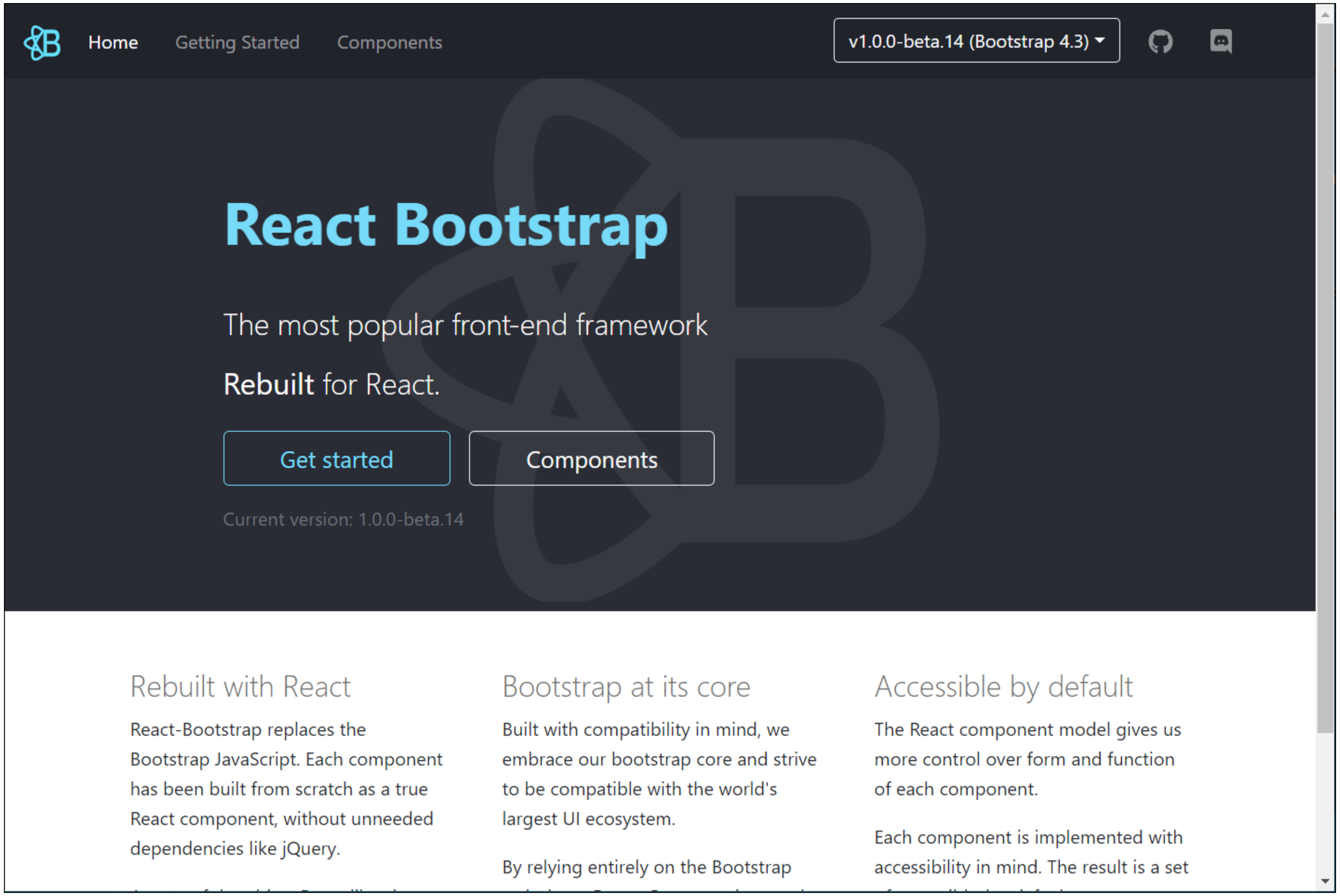
Don’t reinvent the wheel

- There are *a ton* of React UI frameworks available
- The all give you buttons, alerts, datepickers, modal dialogs, and so on
  - Most of them Open Source, free of charge
- Some examples
  - React Bootstrap
  - Material Kit React
  - Material UI
  - Blueprint
  - And many, many more

Basic usage

- The basic usage of all libraries is the same.
- `npm install` the lib to add it to project.
- *Read the docs* for info on
  - importing styles,
  - Adding fonts,
  - Adding icons,
  - Available components,
  - and so on

Example – React Bootstrap



React Bootstrap

The most popular front-end framework

Rebuilt for React.

Current version: 1.0.0-beta.14

Rebuilt with React

React-Bootstrap replaces the Bootstrap JavaScript. Each component has been built from scratch as a true React component, without unneeded dependencies like jQuery.

Bootstrap at its core

Built with compatibility in mind, we embrace our bootstrap core and strive to be compatible with the world's largest UI ecosystem.

By relying entirely on the Bootstrap

Accessible by default

The React component model gives us more control over form and function of each component.

Each component is implemented with accessibility in mind. The result is a set

https://react-bootstrap.github.io/

React Bootstrap

- Popular Bootstrap classes implemented as React Components
- No dependencies on third party libraries (i.e. `jQuery` and `popper.js`)
- The basic bootstrap stylesheet is still required!
- So install both packages

```
npm install react-bootstrap bootstrap --save
```

Update `App.js` or `index.js`

```
// index.js - import global styling. Still required by react-bootstrap
// for some basic styling.
import 'bootstrap/dist/css/bootstrap.min.css'
```

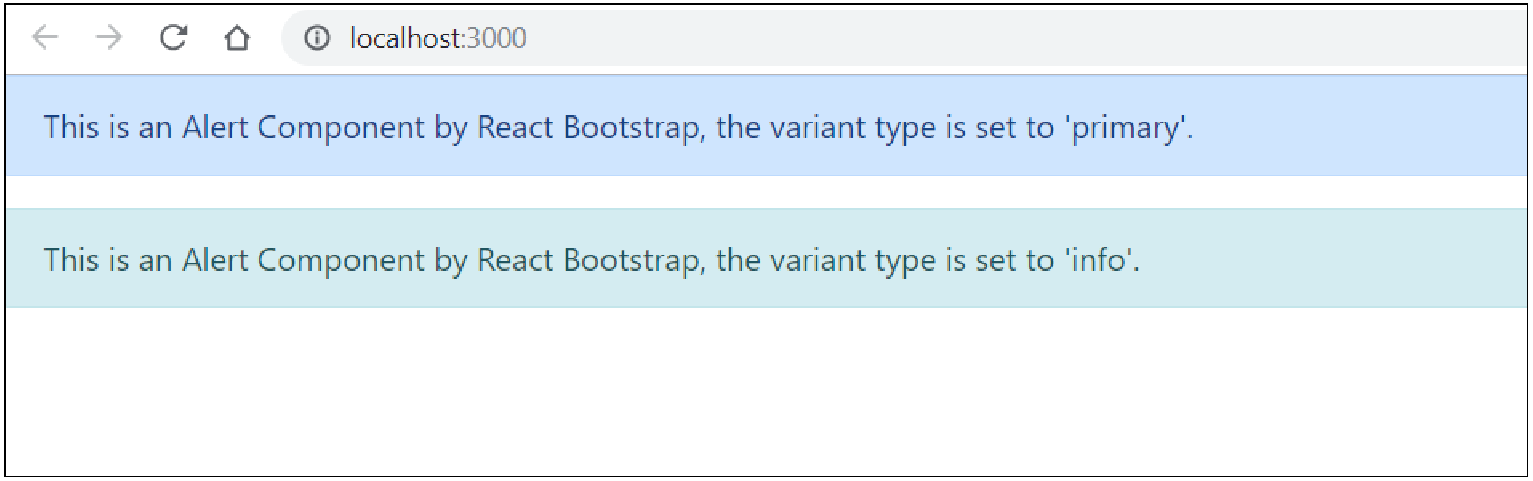
Import just the components you want to use in your UI.

You don't import the complete library

```
import {Alert} from "react-bootstrap";
```

```
<Alert variant={'primary'}>
    This is an Alert Component by React Bootstrap,
    the variant type is set to 'primary'.
</Alert>
```

Result


localhost:3000

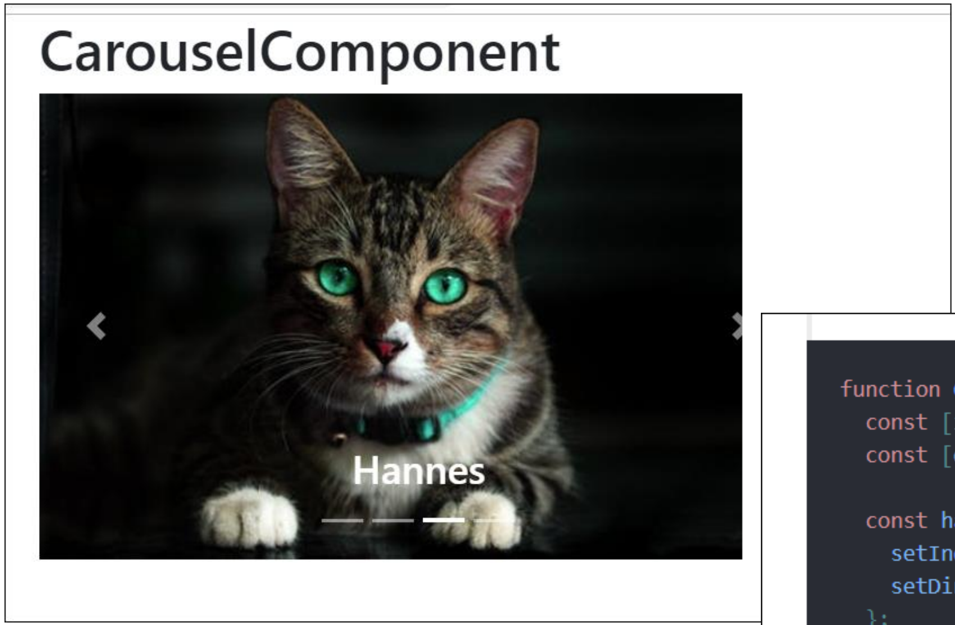
This is an Alert Component by React Bootstrap, the variant type is set to 'primary'.

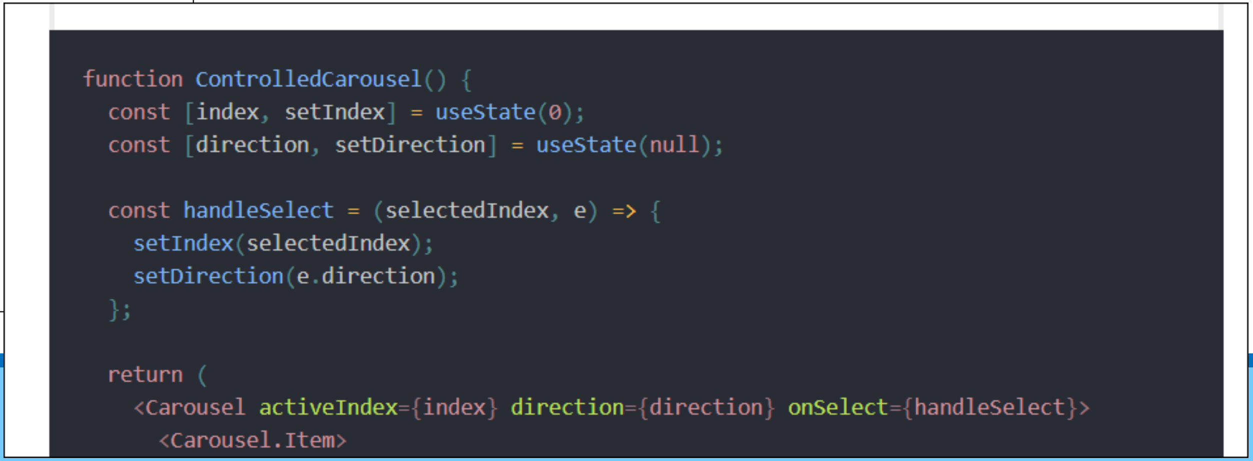
This is an Alert Component by React Bootstrap, the variant type is set to 'info'.

See …`/Components/AlertComponent.js`

Dynamic Components

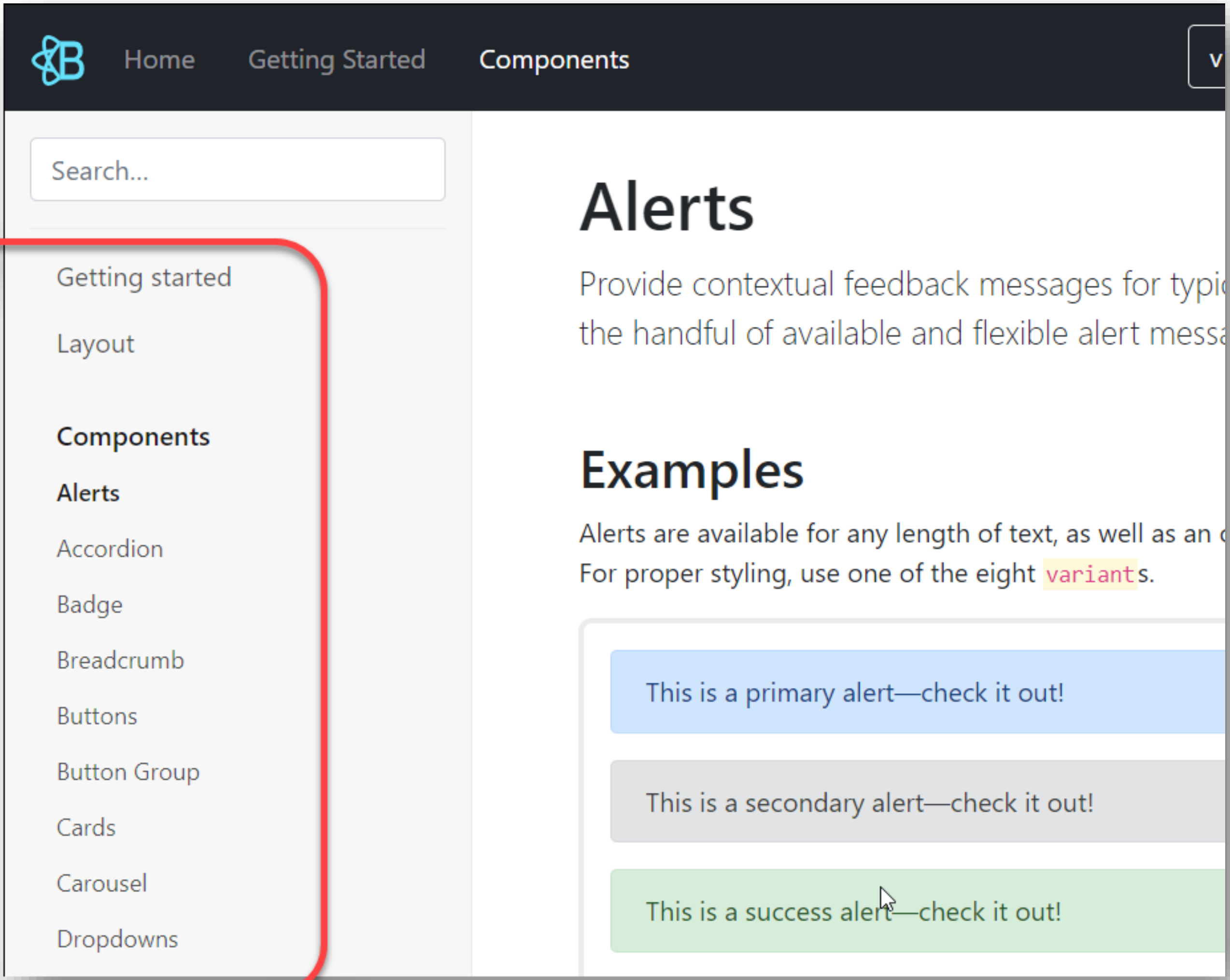
- Don’t require additional libraries
- *May* require additional script
  - For instance if you want to control a carousel via script
  - **RTFM!**





```
function ControlledCarousel() {
  const [index, setIndex] = useState(0);
  const [direction, setDirection] = useState(null);

  const handleSelect = (selectedIndex, e) => {
    setIndex(selectedIndex);
    setDirection(e.direction);
  };

  return (
    <Carousel activeIndex={index} direction={direction} onSelect={handleSelect}>
      <Carousel.Item>
```

More info



Alerts

Provide contextual feedback messages for typi
the handful of available and flexible alert mess

Examples

Alerts are available for any length of text, as well as an
For proper styling, use one of the eight `variant`s.

This is a primary alert—check it out!

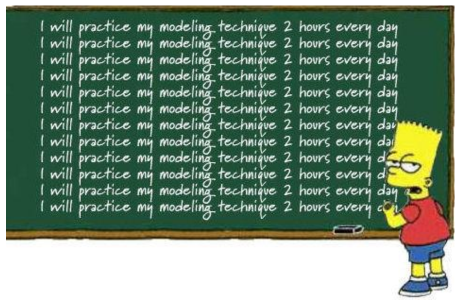
This is a secondary alert—check it out!

This is a success alert—check it out!

https://react-bootstrap.github.io/docs/components/alerts

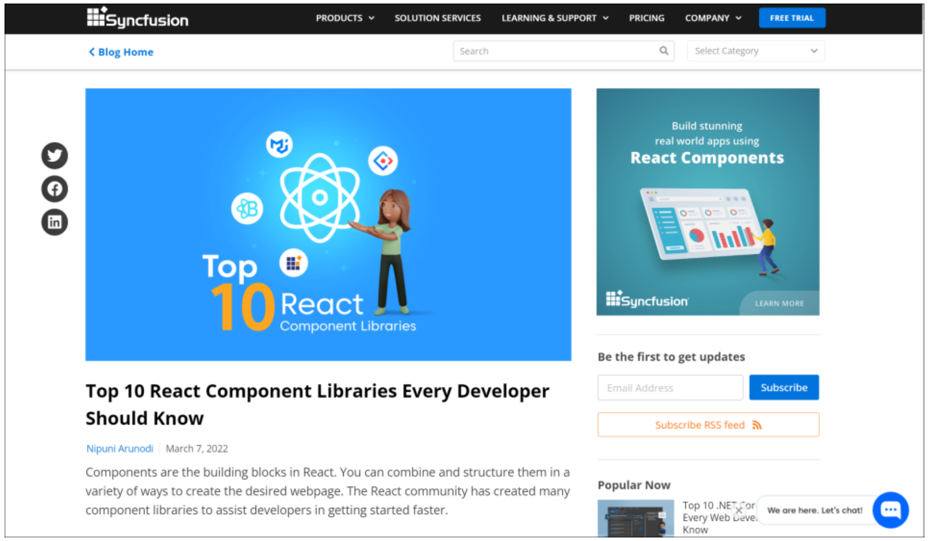
Workshop

- Create a new project
- Add a React UI library of your choice to the project
- Create one or two components, using Components from the library. Examples:
    - Modal dialog
    - Accordeon
    - Datepicker
    - …
- Ready made example `../310-react-bootstrap`



More info on CSS

- https://www.codeinwp.com/blog/react-ui-component-libraries-frameworks/
- https://www.syncfusion.com/blogs/post/top-10-react-component-libraries-every-developer-should-know.aspx



Top 10 React Component Libraries Every Developer Should Know

Nipuni Arunodi | March 7, 2022

Components are the building blocks in React. You can combine and structure them in a variety of ways to create the desired webpage. The React community has created many component libraries to assist developers in getting started faster.

Top 10 .NET for Every Web Deve. Know

Checkpoint

- You know about different types of styles you can use in components
  - Global styles – for your global UI components
  - Inline styles – try to avoid
  - CSS Modules – preferred!
  - Sass – if necessary
- You are familiar with React UI frameworks
- You know where to find them and add to your project