

STYLE GUIDE DRIVEN DEVELOPMENT

with **REACT AND CSS MODULES**

JENNIFER STERN
LEAD UI ENGINEER @ INSTRUCTURE
jstern@instructure.com

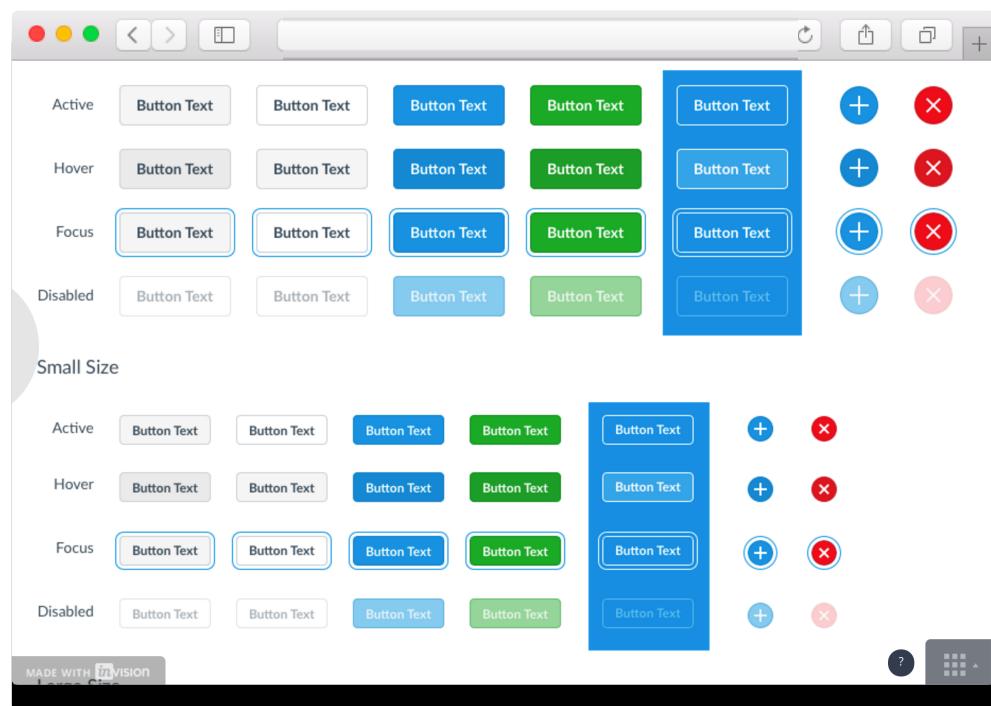
WHAT'S A STYLE GUIDE?

A style guide is a set of standards for the writing and design of documents, either for general use or for a specific publication, organization, or field....

A STYLE GUIDE ESTABLISHES AND ENFORCES STYLE TO IMPROVE COMMUNICATION.

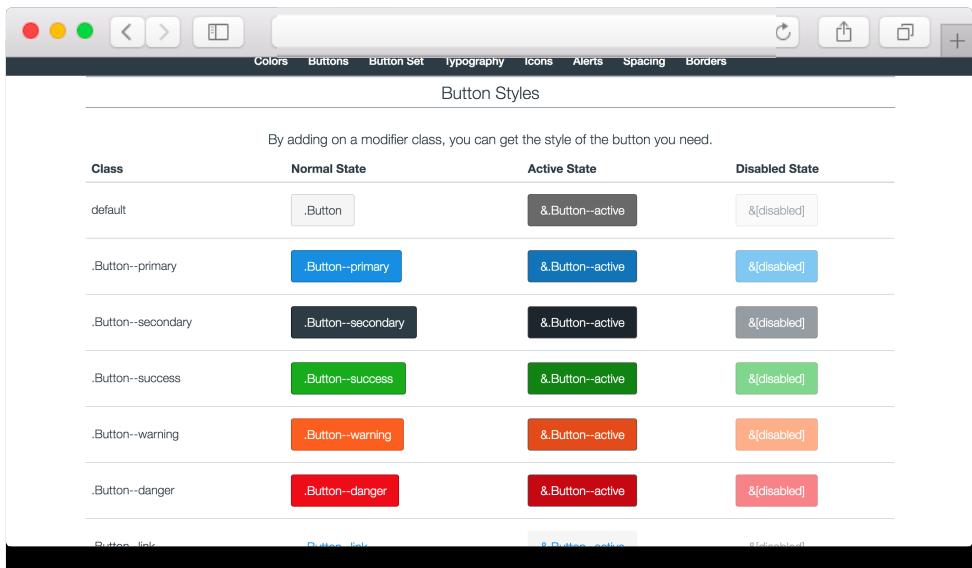
- Wikipedia

STATIC STYLE GUIDE



LIVE STYLE GUIDE

(or pattern library)



The screenshot shows a live style guide interface with a dark header bar containing navigation icons and tabs for Colors, Buttons, Button Set, Typography, Icons, Alerts, Spacing, and Borders. Below the header is a section titled "Button Styles". A sub-instruction "By adding on a modifier class, you can get the style of the button you need." is present. A table displays six rows of button styles, each with three columns: "Normal State", "Active State", and "Disabled State". The rows correspond to different button classes: "default", ".Button--primary", ".Button--secondary", ".Button--success", ".Button--warning", and ".Button--danger". Each row shows a visual representation of the button in its respective state, along with the corresponding CSS selector for each state.

Class	Normal State	Active State	Disabled State
default	.Button	&.Button--active	&[disabled]
.Button--primary	.Button--primary	&.Button--active	&[disabled]
.Button--secondary	.Button--secondary	&.Button--active	&[disabled]
.Button--success	.Button--success	&.Button--active	&[disabled]
.Button--warning	.Button--warning	&.Button--active	&[disabled]
.Button--danger	.Button--danger	&.Button--active	&[disabled]

IN A LIVE STYLE GUIDE

DOCUMENTATION LIVES WITH THE SOURCE CODE IN THE SAME REPOSITORY

```
```

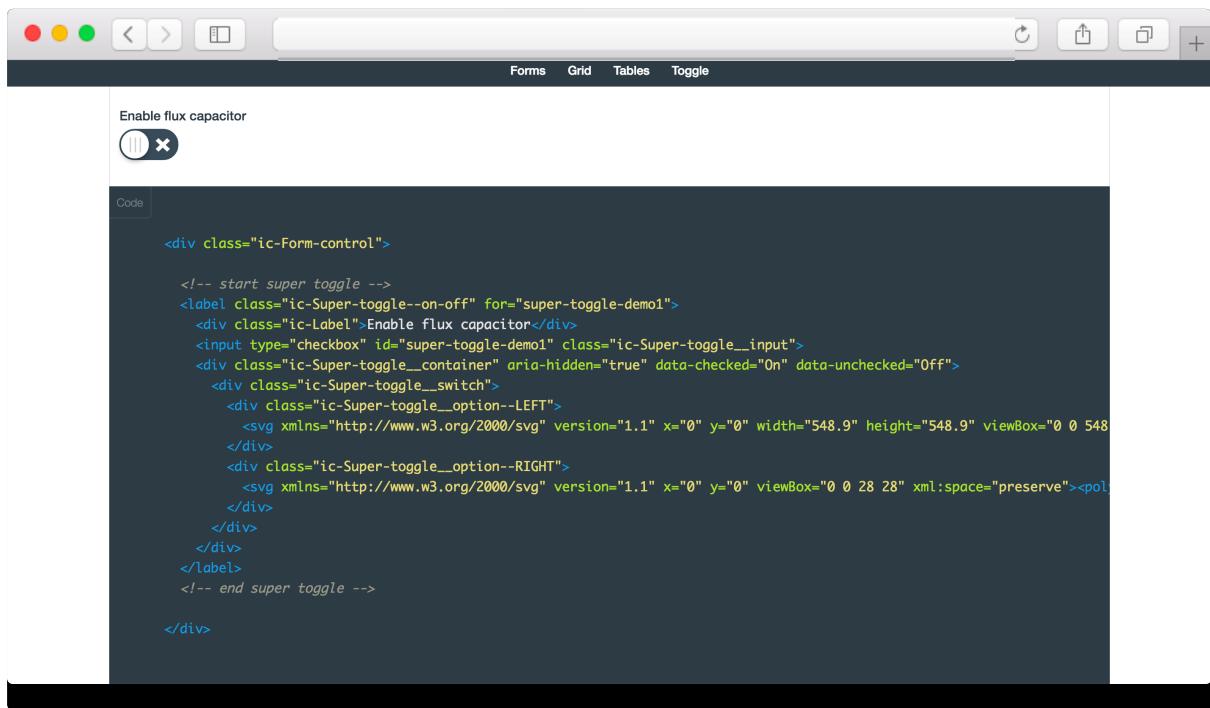
4.
5.
6. // Base styles
7. // -----

8.
9. // Core
10.
11. /*
12. @styleguide Buttons
13.
14. ## The Basic Button
15.
16. ````html
17. <button class="Button" type="button">My
Awesome Button</button>
18. `````
19. ## Button Styles
20.
21. By adding on a modifier class, you can get the
```





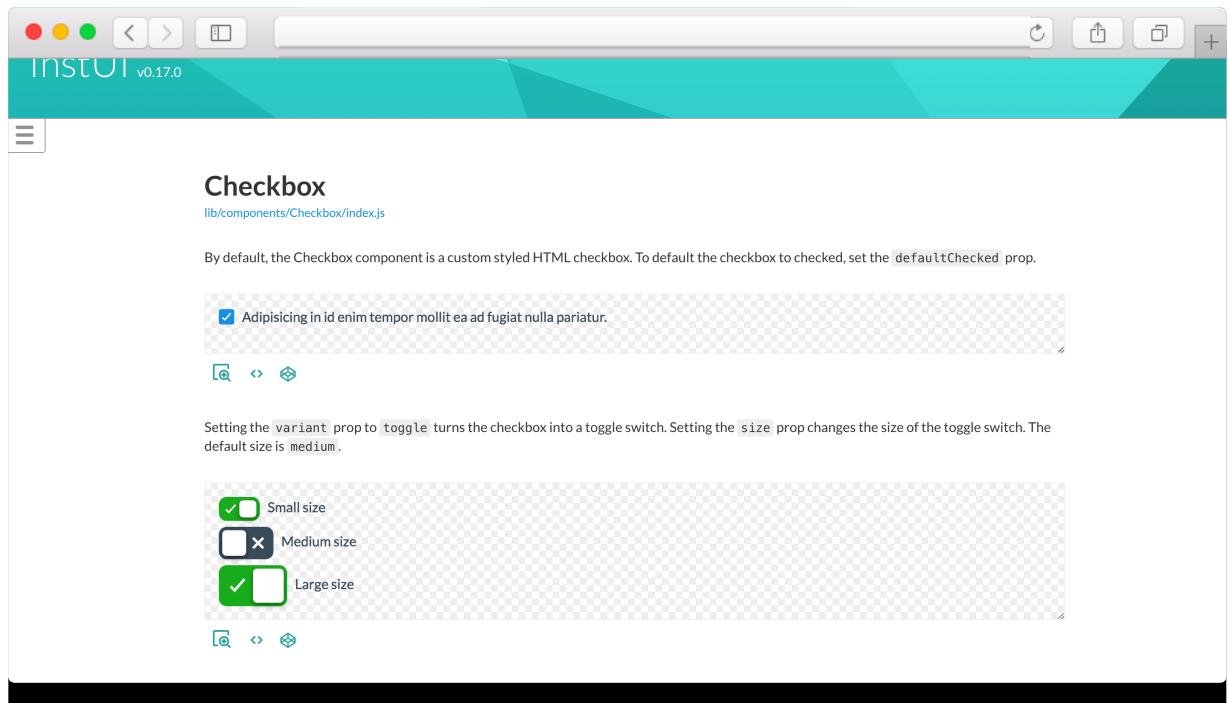
# A CUSTOM CHECKBOX IN SASS



The screenshot shows a web browser window with a dark theme. At the top, there are standard OS X window controls (red, yellow, green buttons) and a toolbar with icons for refresh, forward, back, and others. Below the toolbar, the menu bar includes "Forms", "Grid", "Tables", and "Toggle". The main content area displays a form field labeled "Enable flux capacitor" with a switch icon. To the right of the form, a "Code" tab is selected, showing the following SASS code:

```
<div class="ic-Form-control">
 <!-- start super toggle -->
 <label class="ic-Super-toggle--on-off" for="super-toggle-demo1">
 <div class="ic-Label">Enable flux capacitor</div>
 <input type="checkbox" id="super-toggle-demo1" class="ic-Super-toggle__input" aria-hidden="true" data-checked="On" data-unchecked="Off">
 <div class="ic-Super-toggle__container" aria-hidden="true" data-checked="On" data-unchecked="Off">
 <div class="ic-Super-toggle__switch">
 <div class="ic-Super-toggle__option--LEFT">
 <svg xmlns="http://www.w3.org/2000/svg" version="1.1" x="0" y="0" width="548.9" height="548.9" viewBox="0 0 548.9 548.9"><path d="M 548.9 0 L 0 548.9 Z" /></svg>
 </div>
 <div class="ic-Super-toggle__option--RIGHT">
 <svg xmlns="http://www.w3.org/2000/svg" version="1.1" x="0" y="0" viewBox="0 0 28 28" xml:space="preserve"><path d="M 0 0 L 28 28 Z" /></svg>
 </div>
 </div>
 </div>
 </label>
 <!-- end super toggle -->
</div>
```

# A CUSTOM CHECKBOX IN REACT



CSS, JS, HTML & DOCUMENTATION  
IN ONE PLACE

```
17.
18. /**
19. By default, the Checkbox component is a
20. custom styled HTML checkbox. To default the
21. checkbox to checked,
22. set the `defaultChecked` prop.
23. ````jsx_example
24. <Checkbox label={lorem.sentence()}>
25. value="medium" defaultChecked />
26. `````
27. Setting the `variant` prop to `toggle` turns
28. the checkbox into a toggle switch.
29. Setting the `size` prop changes the size of
30. the toggle switch. The default size
31. is `medium`.
32. ````jsx_example
33. <div>
34. <Checkbox label="Small size" value="small"
```

| Make the right things easy and the wrong things hard.

- *Jonathan Snook*

And now I'll attempt to live code...



## CSS in JS?

1. Global Namespaces
2. Dependencies
3. Dead Code Elimination
4. Minification
5. Sharing Constants
6. Non-Deterministic Resolution
7. Isolation

- Christopher "vjeux" Chedeau

# CSS MODULES

(ISOLATE COMPONENT STYLES **WITHOUT WRITING CSS IN JS**)

# button.css

```
1. .root {
2. all: initial; /* this prevents external CSS
from breaking this component */
3. appearance: none;
4. box-sizing: border-box;
5. display: inline-block;
6. vertical-align: middle;
7. position: relative;
8. overflow: visible;
9.
10. margin: 0;
11.
-->
```

# CSS MODULES

LOADING A CSS FILE IN A JS COMPONENT

# button.js

```
1. import React, { Component, PropTypes } from
'react'
2. import ReactDOM from 'react-dom'
3. import classNames from 'classnames'
4. import themeable from '../util/themeable'
5. import keycode from 'keycode'
6. import { omitProps } from
'../util/passthroughProps'
7.
8. import styles from './styles.css'
9. import theme from './theme.js'
10.
```

# WITH PRODUCTION CONFIG

The screenshot shows a browser window displaying a component preview. The page contains four examples of buttons:

- The default button in its disabled state:** A button labeled "OK" with a greyed-out appearance.
- A button with an href passed outputs a link element styled like a button:** A button labeled "Click Here" with a standard button style.
- Button variants for different contexts using the `variant` prop:** Four buttons labeled "Primary button" (blue), "Success button" (green), "Link button" (light blue), and "Ghost button" (outline).
- Buttons with icons:** Buttons with small icons next to them.

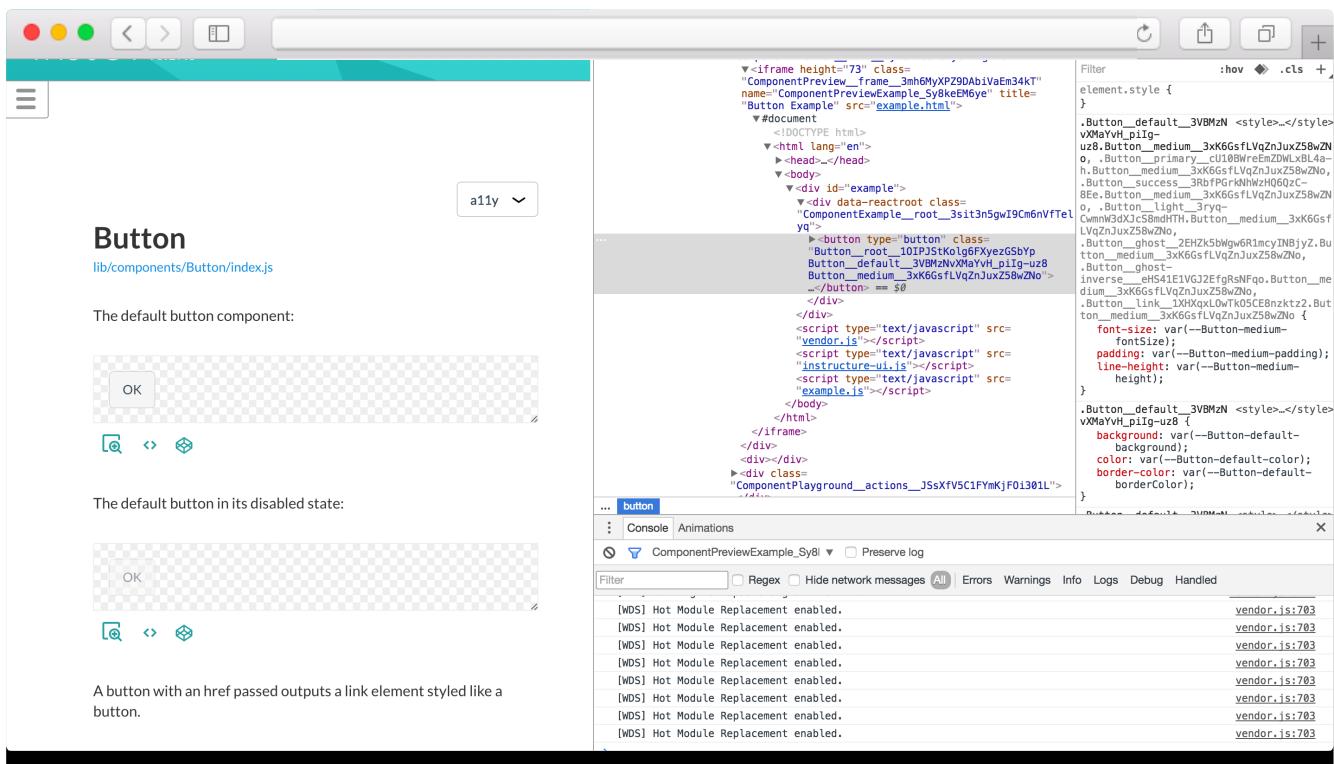
The right side of the browser window shows the generated HTML and CSS code for these components. The CSS includes styles for the `:disabled` state, button variants, and icons. The browser's developer tools are open, showing the DOM structure and the Network tab.

```
<h2 class="23-pykiK-Ohl-TEzGnHlsW">Button</h2>
<div class="3TjxKeMvAK0cQGrp6gu"></div>
<div class="1w7jGSMN2jexapZnRrl2Ac">
 <div>
 <div></div>
 <div class="yk3wVAp-n_K98AD3jksNT"></div>
 <div></div>
 <div class="yk3wVAp-n_K98AD3jksNT"></div>
 <div></div>
 <div class="yk3wVAp-n_K98AD3jksNT"></div>
 <div class="1y462rcUD1eyExBgHB8l2">
 <iframe height="73" class="3m6NyXP290AbiVaEm34kt" name="ComponentPreviewExample_rJIBn7f6ke" title="Button Example" src="example.html">
 <#document
 <!DOCTYPE html>
 <html lang="en">
 <head></head>
 <body>
 <div id="example">

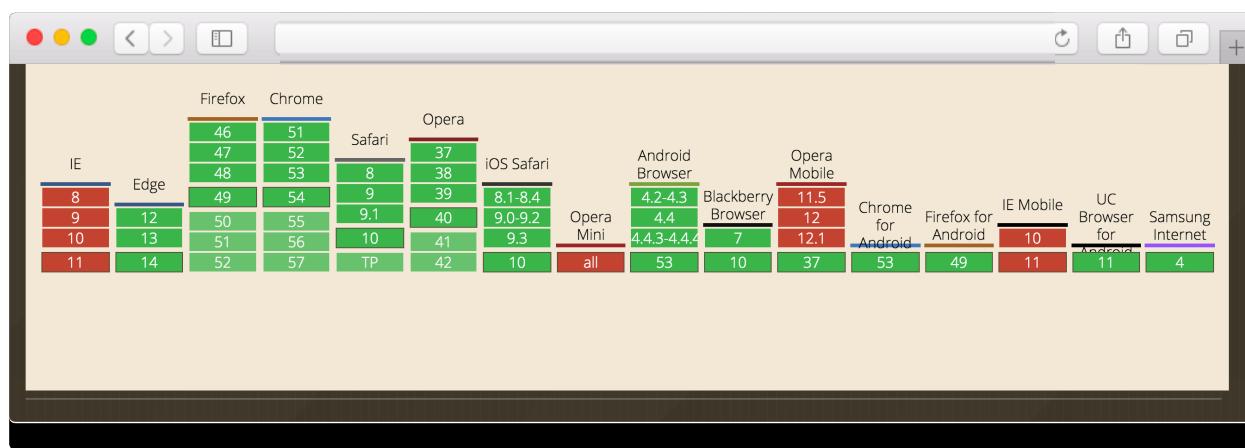
 Click Here

 </div>
 </body>
 </html>
 </#document>
 <script type="text/javascript" src="vendor.js"></script>
 <script type="text/javascript" src="instructions.js"></script>
 <script type="text/javascript" src="10TP1StKoLo6EYuaZGShYn..."/>
 </div>
 </div>
</div>
```

# WITH DEVELOPMENT CONFIG

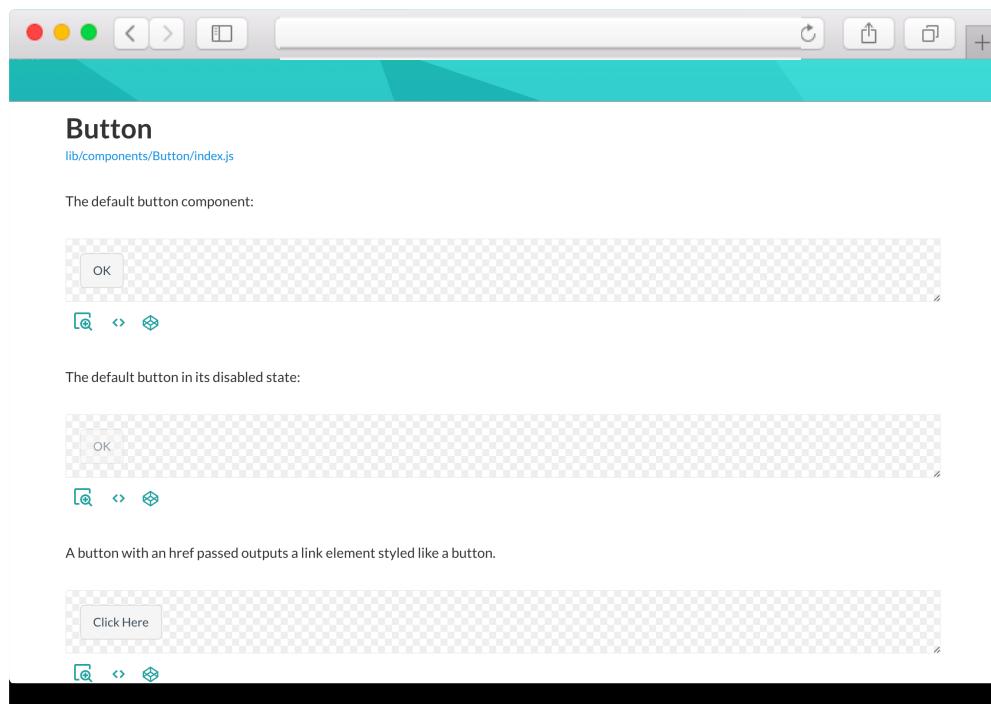


# PREVENT THE CASCADE WITH `all: initial` AND POSTCSS-INITIAL



# ACCESSIBILITY

# ACCESSIBLE BUTTONS



ADDING (JS) BEHAVIOR TO COMPONENTS FOR **ACCESSIBILITY**

# button.js

```
1. import React, { Component, PropTypes } from
'react'
2. import ReactDOM from 'react-dom'
3. import classNames from 'classnames'
4. import themeable from '../util/themeable'
5. import keycode from 'keycode'
6. import { omitProps } from
'../util/passthroughProps'
7.
8. import styles from './styles.css'
9. import theme from './theme.js'
10.
--
```

UNIT TESTS FOR ACCESSIBILITY USING **AXE-CORE**

# checkbox.test.js

```
1. import React from 'react'
2. import Checkbox from '../index'
3.
4. describe('<Checkbox />', function () {
5. const testbed = createTestbed(
6. <Checkbox label="fake label"
defaultChecked value="someValue" name="someName" />
7.)
8.
9. it('renders an input with type "checkbox"',
function () {
10. const subject = testbed.render()
```

TESTING COLOR CONTRAST FOR A11Y

# theme.test.js

```
1. import theme from '../theme'
2. import Brands from '....//theme/brands'
3. import { contrast } from '....//util/color'
4.
5. describe('Button.theme', function () {
6. describe('with the default theme', function
() {
7. const variables = theme(Brands.inst)
8.
9. describe('default', function () {
10. it('should have a background and text
colors that meet 3:1 contrast', function () {
```

# THEMES

# SASS VARIABLES **IN THE MONOLITH**

# variables.scss

```
1. //<!-- TOC
2.
//=====
=====

3. //
4. // - Canvas Brand Variants
5. // - Custom Branding
6. // - Canvas LMS Color Sheet
7. // - Canvas Theme Color Variables
8. //
9.
//=====</pre>
```

# GLOBAL BRAND VARIABLES DEFINED IN JS

# brand.js

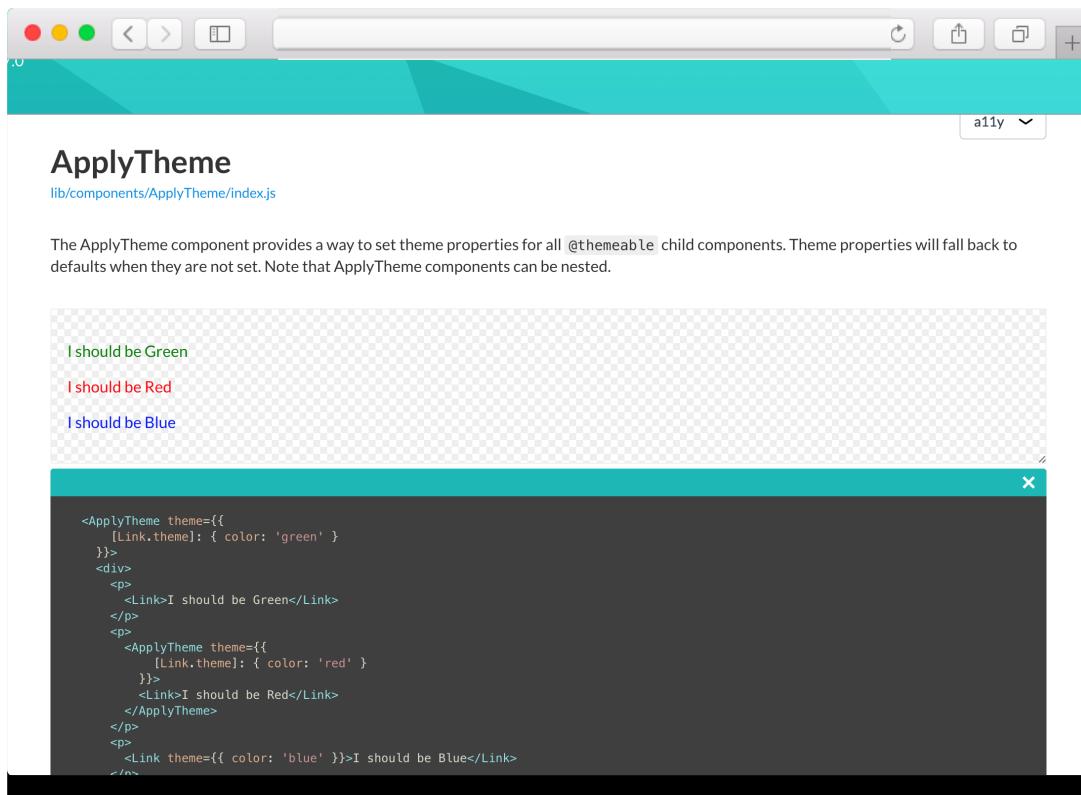
```
1. export default Object.freeze({
2. // These need to meet 3:1 contrast
requirements when used as text against background
3. // colors lightest or when used as a
background with text colors lightest
4. brand: '#008EE2',
5. info: '#008EE2',
6. success: '#00AC18',
7. action: '#BF32A4',
8. danger: '#EE0612',
9. warning: '#FC5E13',
10.
```

# COMPONENT VARIABLES DEFINED IN JS

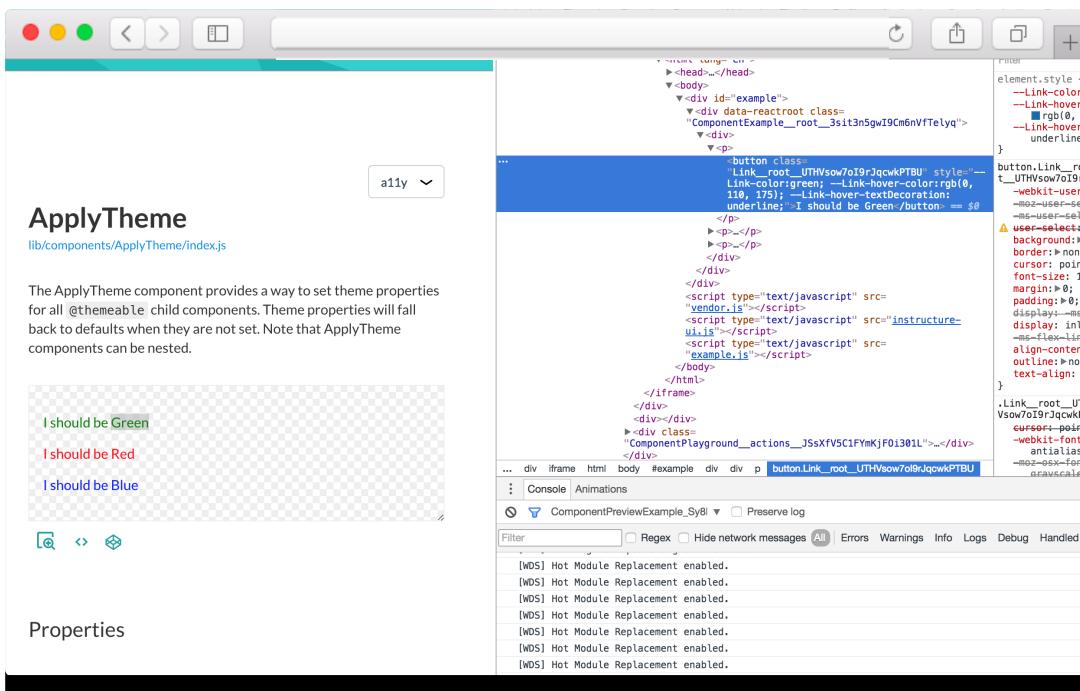
# theme.js

```
1. import { darken } from ' ../../util/color'
2.
3. export default function generator ({ colors,
typography }) {
4. return {
5. fontFamily: typography.fontFamily,
6. fontWeight: typography.fontWeightNormal,
7.
8. color: colors.brand,
9. textDecoration: 'none',
10. hover: {
11. color: darken(colors.brand, 10),
12. }
13. }
14.}
```

# APPLYING THEMES AT RUN TIME



# CSS CUSTOM PROPERTIES

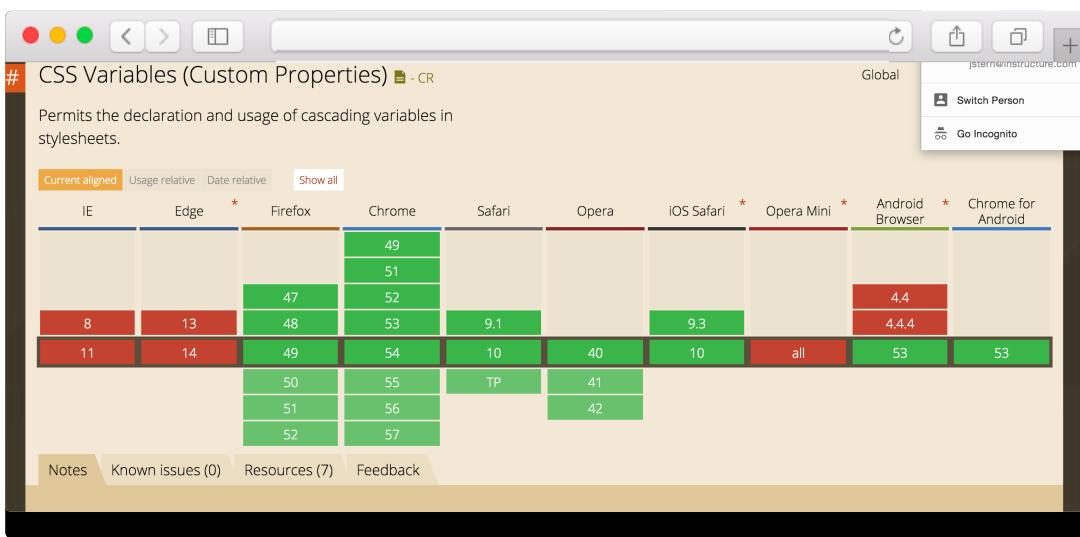


## CSSStyleDeclaration.setProperty()

```
const setNodeProperties = function (node, properties) {
 Object.keys(properties).forEach(function (propertyName) {
 const value = properties[propertyName]
 if (value) {
 node.style.setProperty(propertyName, value)
 }
 })
}
```

# CSS CUSTOM PROPERTIES

(we wrote a polyfill for IE)



# BUILT TO SCALE

We're using our UI components in multiple applications as we start breaking up the monolith into microservices

&

Our professional services team is using the library to build custom integrations with a seamless UX.

# QUESTIONS?

<https://github.com/instructure/instructure-ui>

**WE'RE HIRING!**

Contact Ariel: [apao@instructure.com](mailto:apao@instructure.com)

# RESOURCES AND LINKS

- react-docgen
- eslint
- stylelint
- axe-core
- eslint-plugin-jsx-a11y
- postcss-initial
- css modules
- webpack css-loader

