

CSS in JS: Patterns

Vojtech Miksu



Max Stoiber

@mxstbr



How well do you know CSS? 🧑🏫

Given these classes:

```
.red {  
  color: red;  
}
```

```
.blue {  
  color: blue;  
}
```

Which color would these divs be?

`<div class="red blue">`

`<div class="blue red">`

First red, second blue

First blue, second red

Both blue

Both red

14,517 votes · Final results

4:36 AM · Sep 7, 2018



Max Stoiber

@mxstbr



How well do you know CSS? 🧑🏻💻

Given these classes:

```
.red {  
  color: red;  
}
```

```
.blue {  
  color: blue;  
}
```

Which color would these divs be?

`<div class="red blue">`

`<div class="blue red">`

First red, second blue

9%

First blue, second red

44%

Both blue

43%

Both red

3%

14,517 votes · Final results

4:36 AM · Sep 7, 2018





CLOUDFLARE®

UI Framework

- 50+ packages/components
- [Lerna](#)
- Yarn Workspace
- Monorepo
- React
- Independent versioning
- [cloudflare/cf-ui](#)

But where are the STYLES?

@cloudflare/component-button

```
class Button extends React.Component {  
  render() {  
    let className = "cf-btn cf-btn--" + this.props.type;  
  
    return (  
      <button  
        className={className}  
        disabled={this.props.disabled}  
        onClick={this.props.onClick}  
      >  
        {this.props.children}  
      </button>  
    );  
  }  
}
```

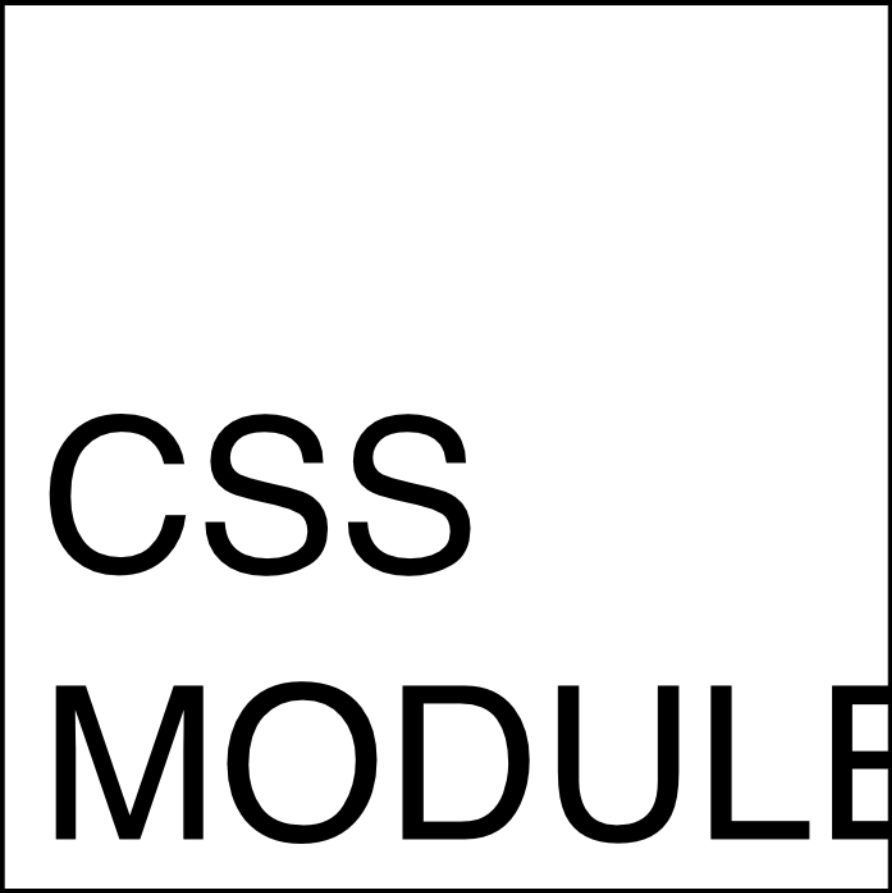

Usage

```
import { Button } from "@cloudflare/component-button";  
  
<Button>Click me</Button>;
```

But...

- Who owns the stylesheet?
- How to load it?
- **Versions?**

Meeting!



CSS MODULES

Modular and reusable

- No more conflicts.
- **Explicit dependencies.**
- No global scope.

CSS Modules

CSS in JS

<https://speakerdeck.com/vjeux/react-css-in-js> (2014)

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

But. So. Many. Libraries.

1. Picking the library

- styled-components
- styled-jsx
- fela
- emotion
- styletron
- ...

Just pick one.



<http://fela.js.org/>

2. Transition

Specificity issues

Legacy global styles

```
input {  
  margin-bottom: 1em;  
}
```

New 'isolated' input component

```
import { createComponent } from "react-fela";

const StyledInput = createComponent(
  () => ({
    border: "1px solid black"
  }),
  "input"
);

// <StyledInput />;

// <style>.a { border: 1px solid black }</style>
// <input class="a" />
```


People issues

- Why are we switching to CSS in JS?
- How do I refactor my old styles?
- Why X doesn't work anymore?

[illegible]

<https://github.com/tajo/fela-workshop>

- Basic Syntax
- Dynamic Styles
- Media Queries
- Selectors
- Prefixes
- Keyframes
- Testing
- ...

3. Testing

Snapshots with Jest

```
import React from "react";
import { Button } from "@cloudflare/component-button";
import renderer from "react-test-renderer";

it("renders tree correctly", () => {
  const tree = renderer.create(<Button>Click me</Button>);
  expect(tree.toJSON()).toMatchSnapshot();
});
```

button.js.snap

```
exports[`renders tree correctly 1`] = `  
  <button  
    className="a"  
  >  
    Click me  
  </button>  
`;  
;
```

Snapshot styles too!

```
import React from "react";
import { Button } from "@cloudflare/component-button";
import { felaSnapshot } from "@cloudflare/style-provider";

test("renders tree and styles correctly", () => {
  const snapshot = felaSnapshot(<Button>Click me</Button>);
  expect(snapshot.component).toMatchSnapshot();
  expect(snapshot.styles).toMatchSnapshot();
});
```

button.js.snap

```
exports[`renders tree and styles correctly 1`] = `  
  <button  
    className="a"  
  >  
    Click me  
  </button>  
`;  
  
exports[`renders tree and styles correctly 2`] = `  
"  
  .a {  
    color: #2869a2  
  }  
"
```


- renders tree and styles correctly

```
expect(value).toMatchSnapshot()
```

Received value does not match stored snapshot "renders tree and styles correctly 2".

– Snapshot

+ Received

```
"
  .a {
-   color: #2869a2
+   color: #2869b2
  }
+
+ .b {
+   border: 0
+ }
"
```

```
6 |   const snapshot = felaSnapshot(<Button>Click me</Button>);
7 |   expect(snapshot.component).toMatchSnapshot();
> 8 |   expect(snapshot.styles).toMatchSnapshot();
   |                           ^
9 |   });
10 |
```

at Object.<anonymous>.test (src/common/component/component-button/__tests__/Button.js:8:27)

> 2 snapshots failed.

~~Mount~~ **Shallow Rendering**

Keep snapshots lean!

@cloudflare/component-card

DNSSEC

DNSSEC protects against forged DNS answers. DNSSEC protected zones are cryptographically signed to ensure the DNS records received are identical to the DNS records published by the domain owner.

Enable DNSSEC

[Help ▶](#)

Don't snapshot **component-button** twice.

End to End Testing

```
▶<div class="c_dq c_dr c_ds c_iu c_ma c_mb">...</div>
```

```
▶<div class="c_dq c_dr c_ds c_iu c_ma c_mb">...</div>
```

```
▼<div class="c_dq c_dr c_ds c_iu c_ma c_mb">
```

```
▼<section class="c_hi c_hj c_iu c_bk c_iv c_iw c_ix c_iy c_fi c_b">
```

```
▼<div class="c_kp c_j c_kq c_kr c_ks c_kt">
```

```
▼<div class="c_s c_ku">
```

```
<h3 class="c_gl c_fi">DNSSEC</h3> == $0
```

"DNSSEC protects against forged DNS answers. DNSSEC protected zones are cryptographically signed to ensure the DNS records received are identical to the DNS records published by the domain owner."

```
</div>
```

```
▼<div class="c_cf c_kw c_kx c_gy c_ef c_eg c_ky c_kz c_la c_lb c_lc c_ld c_le c_ku c_lf">
```

```
▼<div>
```

```
<button type="button" class="c_fw c_fx c_fy c_fz c_ga c_gb c_gc c_gd c_ge c_gf c_gg c_gh c_gi c_gj c_v  
c_q c_gk c_ex c_gl c_gm c_gn c_go c_gp c_dg c_gq c_gr c_gs c_gt c_gu c_gv c_gw c_gx c_dd c_cf c_gy  
c_gz c_ha c_hb c_u c_hc c_hd c_he c_cq c_fh c_cr c_ch c_av c_aw c_bf c_hf c_dp c_ah c_hg c_db c_ed">  
Enable DNSSEC</button>
```

```
</div>
```

```
</div>
```

```
</div>
```

data-test-id="dnssec-btn"

[data-test-id="dnssec-btn"]

4. Selectors vs Isolation

Combinators and Pseudo-classes

- Sibling, child, descedant...
- Odd, even, nth-child...
- Too powerful. Kills encapsulation.
- **Just don't use them.**

But I need them?

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

Use proper component APIs instead 💡

- React Context
- React.cloneElement()
- Props
- Render Props

Striped table

5. Customization

First version

```
// .cf-btn {  
//   margin: 1em  
// }  
  
export const Button = ({ children }) => (  
  <button className="cf-btn">{children}</button>  
);  
  
// .hackywrapper .cf-btn {  
//   margin: 0em  
// }  
  
<div className="hacky-wrapper">
```

CSS in JS used

```
export const Button = createComponent(  
  ({ children }) => ({  
    margin: "1em"  
  }),  
  "button"  
);
```

How to hack it now?

Class Selectors (class names)

- Computed.
- Heavily shared (atomic CSS).
- Changed frequently.
- 🙅 **Never reference them anywhere.**

CSS. ANYTHING IS POSSIBLE. 🧙

```
export const Button = createComponent(  
  ({ children }) => ({  
    margin: "1em"  
  }),  
  "button"  
);
```

```
// .even-hackier-wrapper button {  
//   margin: 0em  
// }
```

```
<div className="even-hackier-wrapper">  
  <Button>Why you hate me?</Button>  
</div>
```


Ideal(istic) solution

```
export const Button = createComponent(  
  ({ children, noMargin }) => ({  
    margin: noMargin ? 0 : "1em"  
  }),  
  "button"  
);  
  
<Button noMargin>Happy me</Button>;
```

Product developers just don't want to do that.

jxnblk/styled-system

```
// space scale: 0px, 4px, 8px, 16px, 32px, 64px, 128px  
// gray scale: #1d1f20, #36393a, #4e5255...
```

```
<Button mx={3} />  
<Button my={2} />  
<Button m={1} />  
<Button mb={5} />  
<Button width={1/2} />  
<Card bg="gray.1" />  
<Table p={[2, 4]} />
```

6. Themes

Basic usage

```
const Text = createComponent(({ theme }) => ({  
  color: theme.color  
}));
```

```
const Fragment = () => (  
  <ThemeProvider theme={{ color: "red" }}>  
    <Text>I am red</Text>  
  </ThemeProvider>  
);
```

Nested

```
const Text = createComponent(({ theme }) => ({
  color: theme.color,
  background: theme.background
}));
```

```
const Fragment = () => (
  <ThemeProvider theme={{ color: "red" }}>
    <ThemeProvider
      theme={parent => ({ color: "blue", background: parent.color })}
    >
      <Text>I am blue with red background.</Text>
    </ThemeProvider>
  </ThemeProvider>
);
```

Yes. We invented the cascade. 🤔

7. Tooling

<https://polished.js.org/>

- A lightweight toolset for writing styles in JavaScript.
- Lodash for CSS in JS.
- Mixins, shorthands, colors...

Ellipsis mixin

TypeScript 🚂

Main.tsx



You, a few seconds ago | 1 author (You)

```
import { createComponent } from '@cloudflare/style-container';
```


```
const MyInput = createComponent(({ theme }) => ({  
  color: theme.colors.  
}))
```

You, a few seconds ago · Uncommitted changes



theme.colors.

Emmet Abbreviation ⓘ

-  black
-  blue
-  cfOrange
-  cyan
-  gold
-  gray
-  green
-  indigo
-  marketing
-  orange
-  red

Main.tsx



You, a few seconds ago | 1 author (You)

```
import { createComponent } from '@cloudflare/style-container';
```

```
const MyInput = createComponent(({ theme }) => ({
```

```
  color: theme.colors.gray[2],
```

```
  bor
```

You, a few seconds ago · Uncommitted changes

```
}))
```

border (property) border: string | numb...

borderBlockEnd

borderBlockEndColor

borderBlockEndStyle

borderBlockEndWidth

borderBlockStart

borderBlockStartColor

borderBlockStartStyle

borderBlockStartWidth

borderBottom

borderBottomColor

borderBottomLeftRadius

Summary

1. Picking the library
2. Transition
3. Testing
4. Selectors vs Isolation
5. Customization
6. Themes
7. Tooling

CSS in JS works.

@vmiksu

<https://css-in-js.now.sh/>