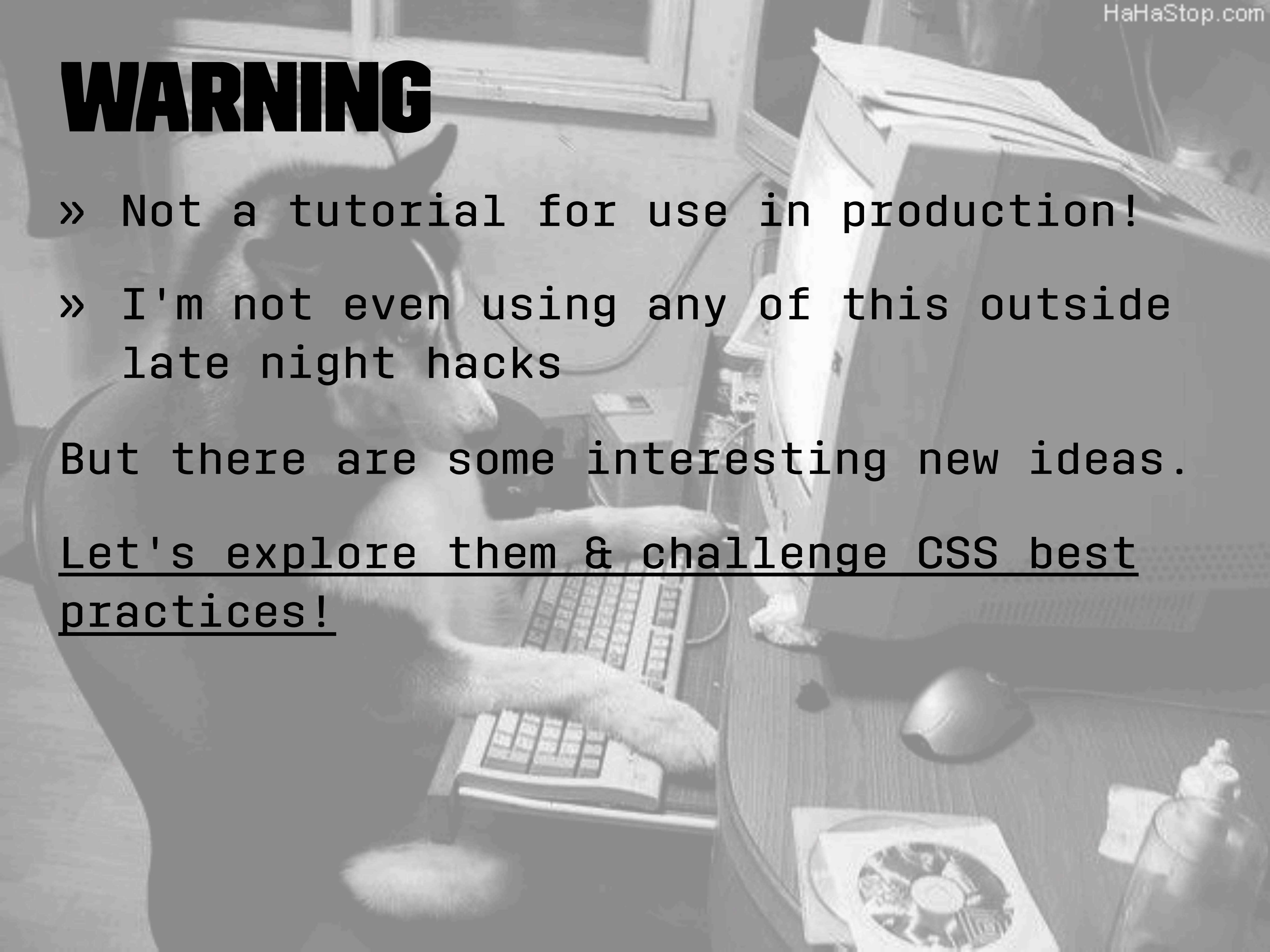
# STYLING COMPONENTS WITH JAVASCRIPT

@BENSMITHETT



## WARNING

- Not a tutorial for use in production!
- I'm not even using any of this outside late night hacks

But there are some interesting new ideas.

Let's explore them & challenge CSS best practices!

## COMPONENTS ARE AWESOME!

Nobody builds pages any more.

Here's an example Profile component:

```
components/
  Profile/
    index.hbs
    index.css
    index.js
```

## HTML TEMPLATE

```
<div class="profile">
  <img class="profile__avatar" src="{{avatarUrl}}.jpg" />
  <strong>{{username}}</strong>
</div>
```

## STYLE

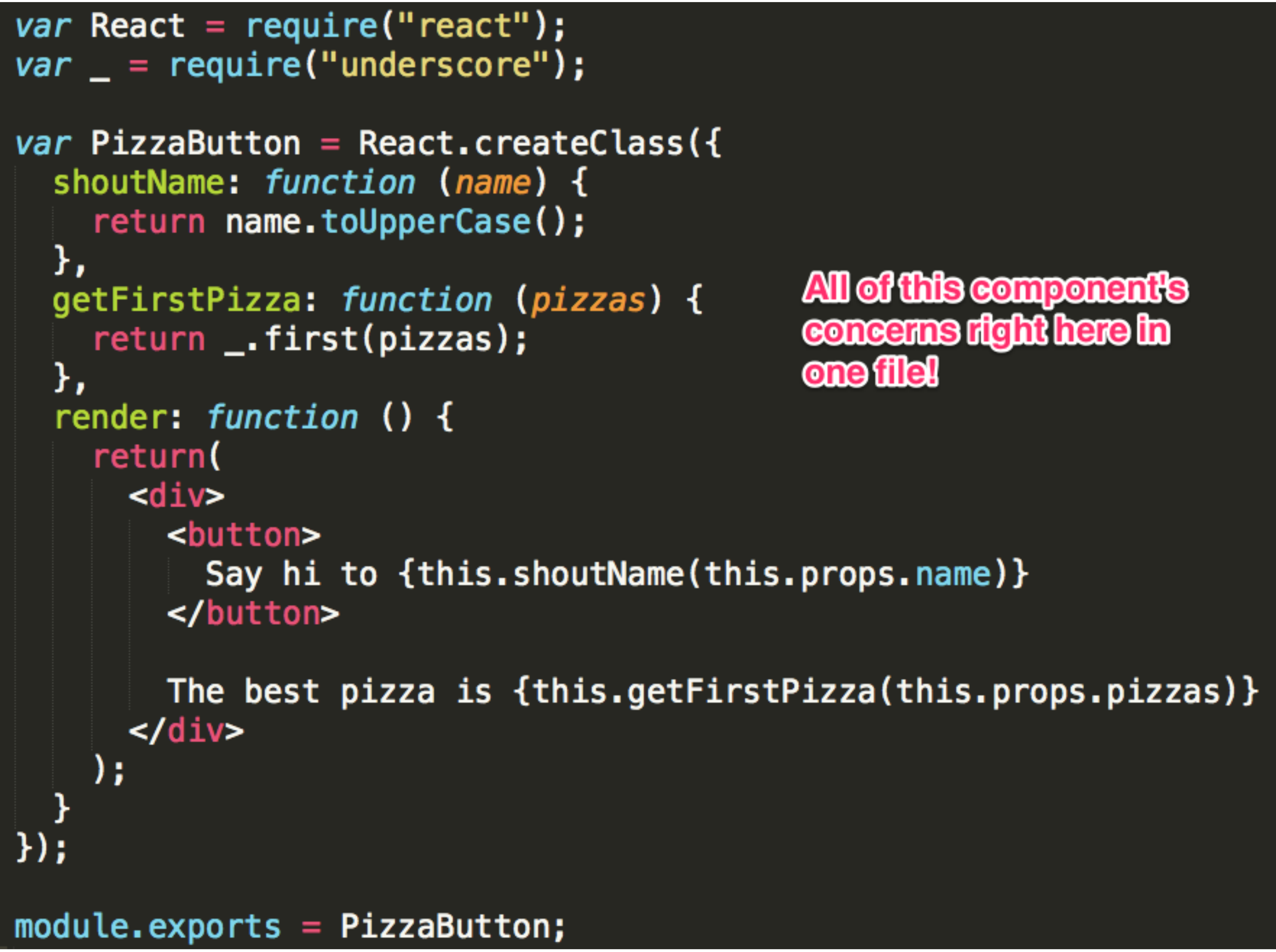
```
.profile {
  border: 1px solid #ddd;
  overflow: hidden;
}

.profile__avatar {
  float: left;
  margin-right: 10px;
}
```

## BEHAVIOUR

```
var Profile = function (el) {
  el.addEventListener("click", function () {
    console.log("hai!");
  });
  this.el = el;
  this.tmpl = Handlebars.compile(someTemplateString);
};

Profile.prototype.render = function (state) {
  this.el.innerHTML = this.tmpl(state);
};
```

## REACT COMBINES HTML STRUCTURE & BEHAVIOUR

```
var React = require("react");

var Profile = React.createClass({
  handleClick: function () {
    console.log("hai");
  },
  render: function () {
    return (
      <div class="profile">
        <img class="profile__avatar" src="{this.props.avatarUrl}.jpg"
             onClick={this.handleClick} />
        <strong>{this.props.username}</strong>
      </div>
    );
  }
});

module.exports = Profile;
```



```
var React = require("react");
var _ = require("underscore");

var PizzaButton = React.createClass({
  shoutName: function (name) {
    return name.toUpperCase();
  },
  getFirstPizza: function (pizzas) {
    return _.first(pizzas);
  },
  render: function () {
    return(
      <div>
        <button>
          Say hi to {this.shoutName(this.props.name)}
        </button>

        The best pizza is {this.getFirstPizza(this.props.pizzas)}
      </div>
    );
  }
});

module.exports = PizzaButton;
```

All of this component's concerns right here in one file!

## THAT'S A BIG DIRTY LIE

The component's CSS is one of its concerns, but it's off in some random other file.

```
components/
  Profile/
    index.css
    index.jsx
```

## The only connection: a class name

```
// JS
render: function () {
  return (
    <div class="profile">
      // ...
    </div>
  )
}

/* CSS */
.profile
  border: 1px solid #ddd;
  overflow: hidden;
```

## MOST THINGS

- » JS Dependencies are explicitly required
- » HTML structure is right there in the file
- » JS behaviour is right there in the file

## CSS

- » In another file, the classes might have the same name ¯_(ツ)_/¯

It's a crappy, vague connection.

## CSS BUILDS ARE A BIT BACKWARDS

```
//app.scss
@import vendor/Normalize.css;
@import base;

@import components/Header;
@import components/Profile;
@import components/Footer;
```

You need to <u>know</u> which bits of CSS your app requires. Lame.

## What if our JS build automatically created a stylesheet based only on the components we use?

```
// app.js
var Profile = require("./components/Profile");
var Header = require("./components/Header");
var Footer = require("./components/Footer");


// app.css

// Somehow...
// components/Profile/index.css
// components/Header/index.css
// components/Footer/index.css
// ... end up here?
```

http://webpack.github.io/

```
var React = require("react");
require("./index.css");

var Profile = React.createClass({
  render: function () {
    return (
      <div class="profile" />
    );
  }
});

module.exports = Profile;
```

```
// app.js
var Profile = require("./components/Profile");
var Header = require("./components/Header");
var Footer = require("./components/Footer");


// app.css generated by webpack
.profile { ... }
.profile__avatar { ... }
.header { ... }
.footer { ... }
```

## HOORAY!

CSS is just another dependency we can require() in our component

## HOORAY?

```
components/
  Profile/
    index.css
    index.jsx
```

» Still working across 2 files

» Need to maintain class names across 2 files

... still a bit lame.

## REACT CAN DO INLINE STYLES

```
// Profile/index.js
var Profile = React.createClass({
  styles: {
    border: "1px solid #ddd",
    overflow: "hidden"
  },
  render: function () {
    return(
      <div style={this.styles} />
    );
  }
});

<!-- DOM generated by React -->
<div style="border: 1px solid #ddd; overflow: hidden;">
</div>
```

## NOBODY LIKES INLINE STYLES THOUGH

# WHAT WE REALLY WANT:

- Declare styles <u>in the component</u>, like we do with inline styles
- Build process that...
  - converts them to a CSS class
  - spits out a shiny, auto-generated app.css
  - component knows to use that class name

## REACT-STYLE DOES THAT!

- https://github.com/SanderSpies/react-style
- http://andreypopp.com/posts/2014-08-06-react-style.html

(with a little help from webpack)

```
var React = require("react/addons");
var ReactStyle = require("react-style");

var Profile = React.createClass({
  styles: ReactStyle(function () {
    return {
      backgroundColor: "green",
      border: "1px solid #ddd"
    };
  }),
  render: function () {
    return(
      <div styles={this.styles()} />
    );
  }
});

module.exports = Profile;
```

```
<!-- DOM generated by React -->
<div class="a">
  ...
</div>

// app.css generated by React-style & Webpack
.a {
  background-color: green;
  border: 1px solid #ddd;
}
```

# DEMO

Compiling with default compressed class names

# DEMO

Formatting class names

## DO YOU EVEN NEED A CSS NAMING CONVENTION?

## Not really...

- Styles are tightly coupled part of the component, not a separate thing
- CSS class naming conventions are a project setting, not an inherent property of the component
  - <u>Dev:</u> BEM class names for easy debugging
  - <u>Prod:</u> Tiny compressed class names

# I <3 SASS

```
$red: #f00;
$grid-columns: 12;
$base-font-size: 16px;

@function px-to-em($px, $base: $base-font-size) {
    @return ($px / $base) * 1em;
}

%placeholder {
  color: $red;
}

.thing {
  @extend %placeholder;
  padding: 10px;
}
```

## WHAT IS A PREPROCESSOR?

A language that helps us generate blocks of key:value pairs.

```
selector {
  property: value;
  other-property: other-value;
}
```

# WHAT IS A PREPROCESSOR?

A language that helps us generate blocks of key:value pairs.

```
selector = {
  property: "value",
  other-property: "other-value"
};
```

JavaScript can do that!

## JS already has lots of Real Programming Language Things TM

- Variables
- Functions
- Arrays & Objects
- Loops
- Maths
- String manipulation
- Dependency management

## WARNING!

Total pseudocode, nothing past this point actually works

## EXAMPLE: COLOR VARIABLES

```
var colors = require("./color_palette");

var Profile = React.createClass({
  styles: ReactStyle(function () {
    return {
      color: colors["hotPink"],
    };
  }),
  render: function () {
    return(
      <div styles={this.styles()} />
    );
  }
});
```

## EXAMPLE: GENERATE A GRID

```
var gridColumns = 12;
var styles = {};

for (var i = 1; i <= gridColumns; i++) {
  var width = (i / gridColumns) * 100;
  styles["span-" + i] = ReactStyle(function () {
    return {
      float: "left",
      width: width + "%"
    }
  });
}

var GridColumn = React.createClass({
  styles: styles,
  render: function () {
    var columns = "span-" + this.props.columns;
    return(
      <div styles={this.styles[columns]()} />
    );
  }
});
```

## 2015 HIPSTER PREPROCESSOR
## JAVASCRIPT?!

# THE END :)

@bensmithett

https://github.com/bensmithett/react-style-example
https://github.com/SanderSpies/react-style
https://github.com/chenglou/rcss
https://github.com/hedgerwang/react-styles
https://github.com/elierotenberg/react-css