Modular CSS

with @okonetchnikov

Andrey Okonetchnikov

Front-End Engineer

@okonetchnikov

http://okonet.ru

https://github.com/okonet

andrey@okonet.ru

I design & build

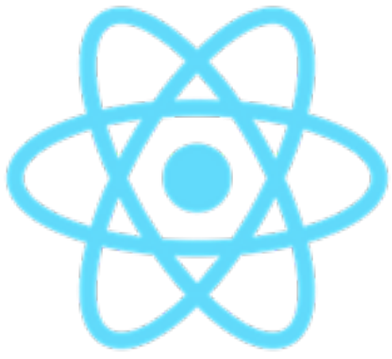
User Interfaces

Building scalable

User Interfaces

User Interfaces $\ni$ Components

UI libraries & frameworks with UI components

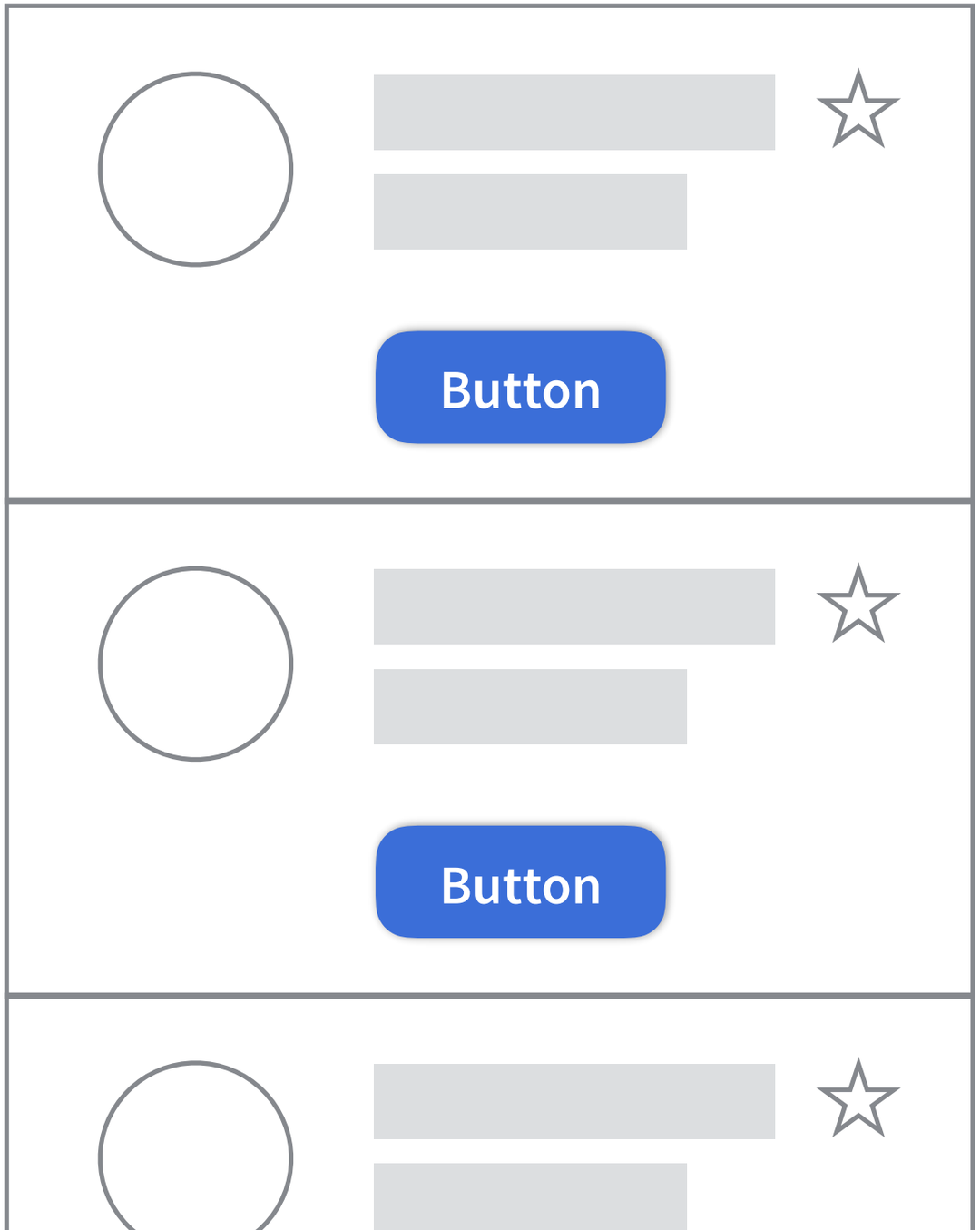


and many others…

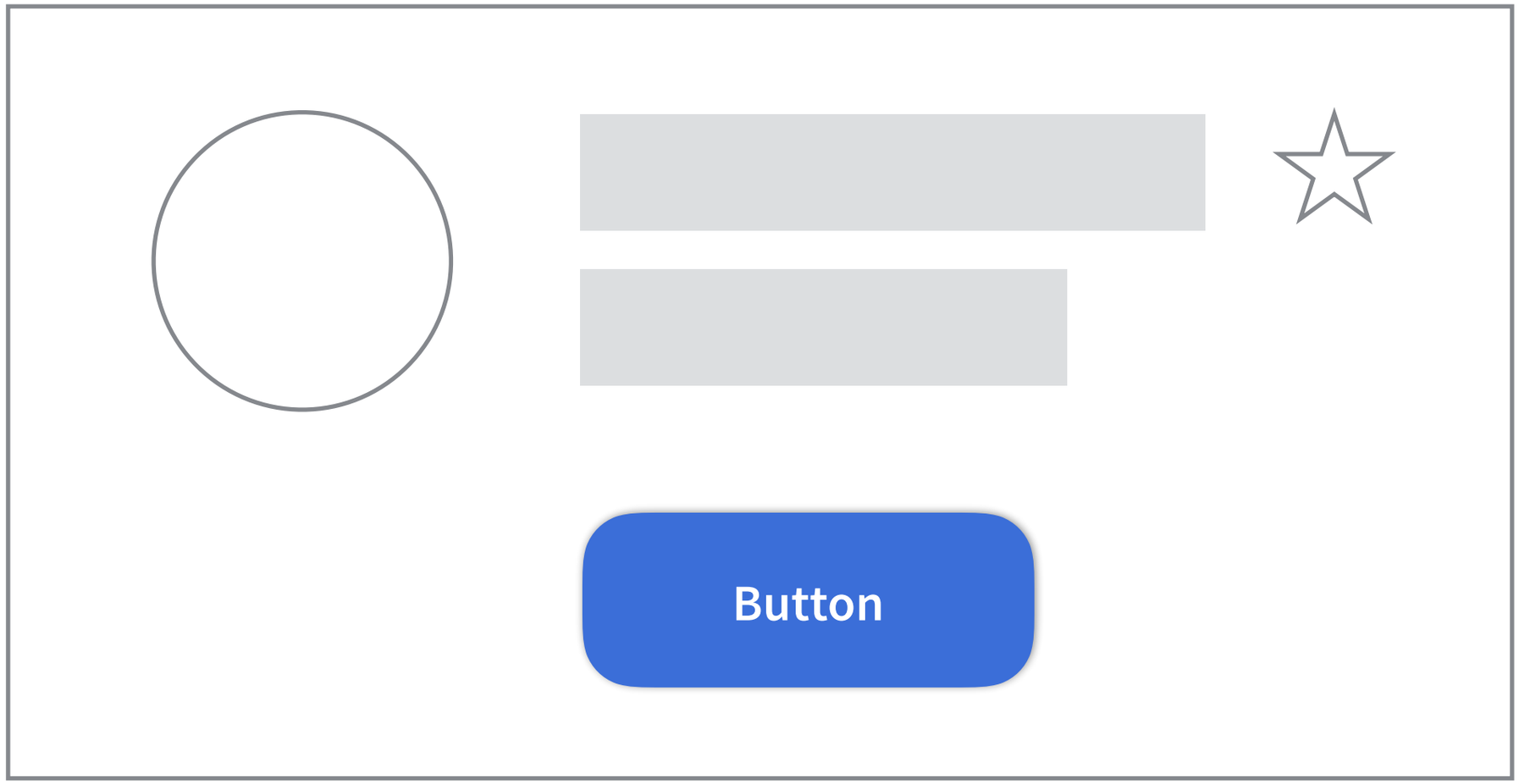
In React.js everything is a component!

What are UI Components?

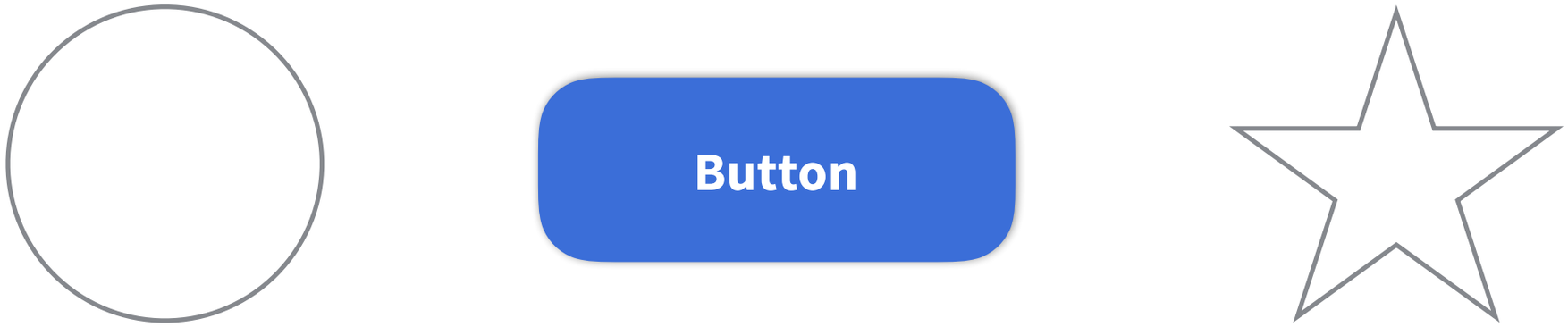
List component



List Item Component



Smaller components



Button component

Default

Normal

Disabled

React.JS component

```
import { Component } from 'react'

export default class Button extends Component {
  render() {
    let btnClass = 'btn'
    if (this.props.disabled) btnClass += ' btn-disabled'
    if (this.state.hovered) btnClass += ' btn-hovered'
    return (
       <button className={btnClass}>
         <span className="label">
          { this.props.children }
         </span>
       </button>
     )
  }
}
```

Anatomy of the UI Component

- External data
- Internal state
- Behavior
- Appearance / Styles

Anatomy of the UI Component

- External data
- Internal state
- Behavior
- Appearance / Styles

- External data
- Internal state
- Behavior
- Appearance / Styles

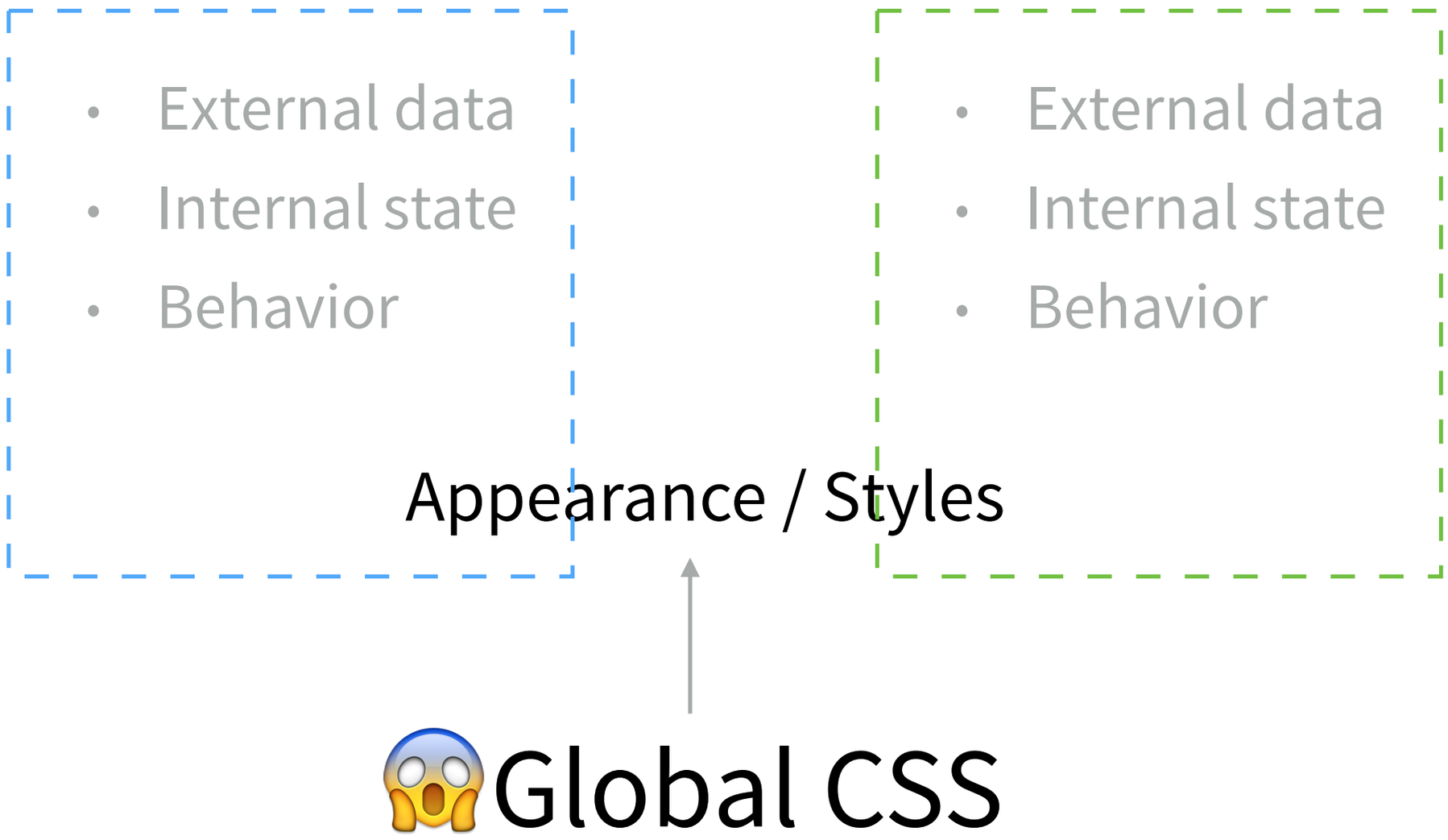
Components should be

self-contained & isolated

and yet…

Anatomy of the UI Component

Anatomy of the UI Component



This leads to…


CSS
Global
HD

Can we do better?

Where is my CSS?

```
import { Component } from 'react'

export default class Button extends Component {
  render() {
    let btnClass = 'btn'
    if (this.props.disabled) btnClass += ' btn-disabled'
    if (this.state.hovered) btnClass += ' btn-hovered'
    return (
       <button className={btnClass}>
         <span className="label">
          { this.props.children }
         </span>
       </button>
     )
  }
}
```

Where is my CSS?

Where is this class defined?

```
import { Component } from 'react'

export default class Button extends Component {
  render() {
   let btnClass = 'btn'
    if (this.props.disabled) btnClass += ' btn-disabled'
    if (this.state.hovered) btnClass += ' btn-hovered'
    return (
       <button className={btnClass}>
         <span className="label">
          { this.props.children }
         </span>
       </button>
     )
  }
}
```

application.css

```
.clearfix {
  *zoom: 1;
}
.clearfix:before,
.clearfix:after {
  display: table;
  content: "";
  line-height: 0;
}
.clearfix:after {
  clear: both;
}
.hide-text {
  font: 0/0 a;
  color: transparent;
  text-shadow: none;
  background-color: transparent;
  border: 0;
}
.input-block-level {
  display: block;
  width: 100%;
  min-height: 30px;
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
}
```


CSS Monolith!

Project structure

Separation of concerns

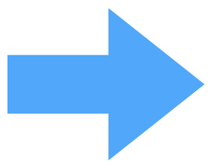
```
public
├── images
│   └── icon.svg
├── javascripts
│   ├── Button.js
│   └── Dropdown.js
└── stylesheets
    ├── Buttons.css
    └── Dropdown.css
```

Separation of ~~concerns~~ technologies

```
public
├── images
│   └── icon.svg
├── javascripts
│   ├── Button.js
│   └── Dropdown.js
└── stylesheets
    ├── Buttons.css
    └── Dropdown.css
```

Separation of concerns

```
public
├── images
│   └── icon.svg
├── javascripts
│   ├── Button.js
│   └── Dropdown.js
└── stylesheets
    ├── Buttons.css
    └── Dropdown.css
```



```
components
├── Button
│   ├── Button.css
│   ├── Button.js
│   ├── Button.spec.js
│   ├── icon.svg
│   └── index.js
└── Dropdown
```

Explicit dependencies

Explicit dependencies

```
import { Component } from 'react'

export default class Button extends Component {
  render() {
    let btnClass = 'btn'
    if (this.props.disabled) btnClass += ' btn-disabled'
    if (this.state.hovered) btnClass += ' btn-hovered'
    return (
      <button className={btnClass}>
        <span className="label">
         { this.props.children }
        </span>
      </button>
    )
  }
}
```

Explicit dependencies

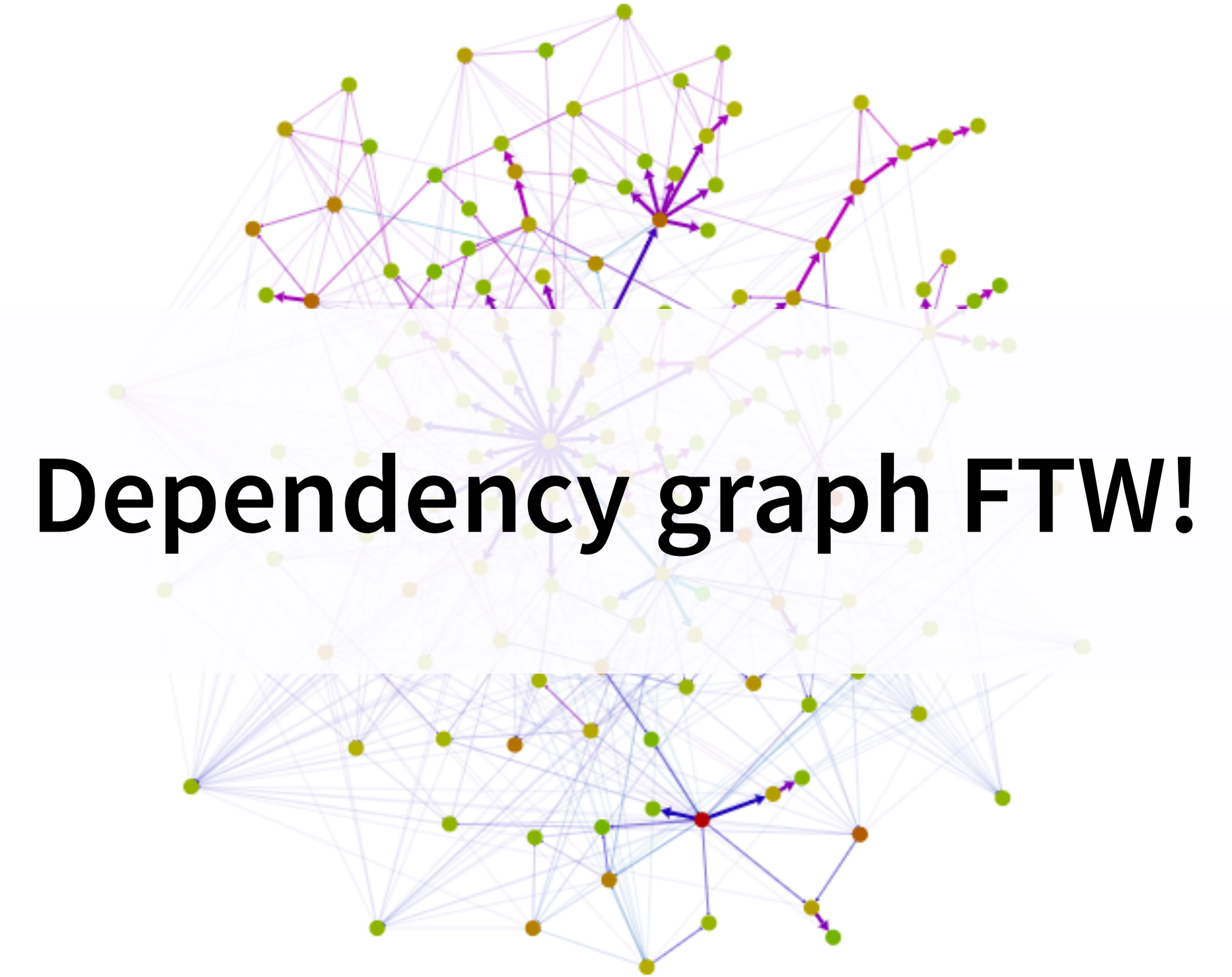
```
import { Component } from 'react'
import './Button.css'

export default class Button extends Component {
  render() {
   let btnClass = 'btn'
   if (this.props.disabled) btnClass += ' btn-disabled'
   if (this.state.hovered) btnClass += ' btn-hovered'
   return (
      <button className={btnClass}>
        <span className="label">
         { this.props.children }
        </span>
      </button>
    )
  }
}
```

1. We can't `require` CSS in JS

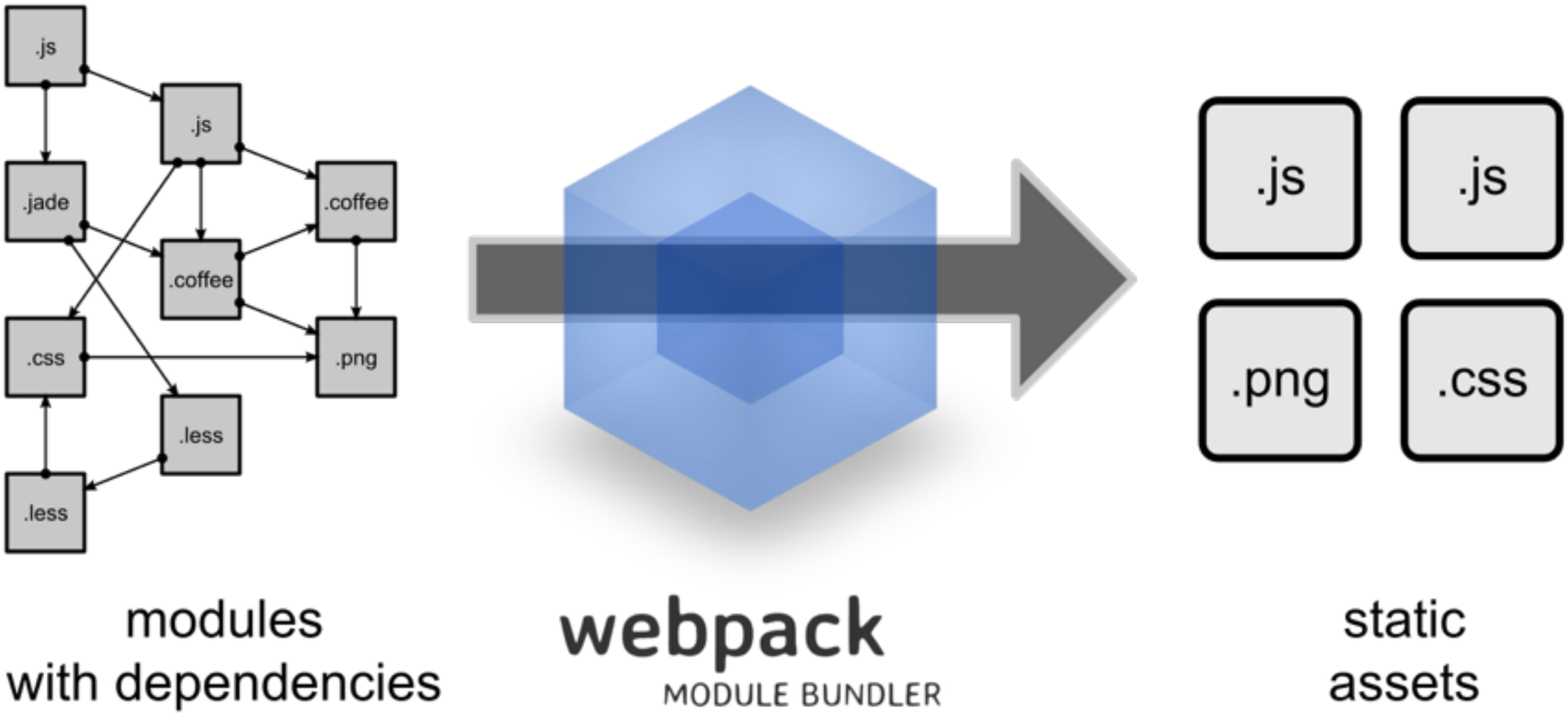
2. Browsers expect `<link>` and `<script>` tags

3. We should minimize the number of `*.css` and `*.js` requests

Common build tools operate on *file trees*



Dependency graph FTW!



http://webpack.github.io/

Allow requiring CSS with Webpack

```
module.exports = {
  module: {
    loaders: [
      { test: /\.css$/, loader: "style-loader!css-loader" },
      { test: /\.png$/, loader: "url-loader?limit=10000" },
      { test: /\.jpg$/, loader: "file-loader" }
    ]
  }
};
```

Load css files, embed small png images as Data URLs and add jpg images as files

Now we can “just” import CSS!

```
import { Component } from 'react'
import './Button.css'

export default class Button extends Component {
  render() {
   let btnClass = 'btn'
   if (this.props.disabled) btnClass += ' btn-disabled'
   if (this.state.hovered) btnClass += ' btn-hovered'
   return (
      <button className={btnClass}>
        <span className="label">
         { this.props.children }
        </span>
      </button>
    )
  }
}
```

Better structure

+ One CSS file per UI component
+ One directory per UI component
+ Co-locating JS, CSS and images
+ Explicit dependencies

Problems with CSS on scale

- Global namespace
- Cascade
- No styles isolation
- Minification

Debugging CSS

BEM

https://en.bem.info/

BEM

- ☑ Global namespace
- ☑ Cascade
- ☑ No styles isolation
- ☐ Not beginner’s friendly
- ☐ Very verbose
- ☐ Requires discipline
- ☐ Minification

Mark Dalgleish
@MelbJS + @DecompressAU organiser, full-stack ECMAScript addict + interaction craftsman at @S...
May 20, 2015 · 7 min read

The End of Global CSS

CSS selectors all exist within the same global scope.

Anyone who has worked with CSS long enough has had to come to terms with its aggressively global nature—a model clearly designed in the age of documents, now struggling to offer a sane working environment for today's modern web applications.

Every selector has the potential to have unintended side effects by targeting unwanted elements or clashing with other selectors. More surprisingly, our selectors may even lose out in the global specificity war, ultimately having

“In other languages, it’s accepted that modifying the global environment is something to be done rarely, if ever.”

– Mark Dalgleish

Local CSS

CSS
MODULES

https://github.com/css-modules/css-modules

How CSS-modules work

How CSS-modules work

```
// Button.js

import React from 'react'
import './Button.css'

export default class Button extends React.Component {
  render() {
    return (
      <button className="button">
        <span className="label">
          Click me!
        </span>
      </button>
    )
  }
}
```

How CSS-modules work

```
/* Button.css */

.button {
  border: 1px solid #f00;
}

.label {
  font-weight: bold;
}
```

How CSS-modules work

```
// webpack.config.js

module.exports = {
  module: {
    loaders: [
      {
        test: /\.css$/,
        loader: 'css?modules
            &localIdentName=[name]__[local]___[hash:base64:5]'
      }
    ]
  }
};
```

How CSS-modules work

```
// Button.js

import React from 'react'
import styles from './Button.css'

export default class Button extends React.Component {
  render() {
    return (
      <button className={styles.button}>
        <span className={styles.label}>
          Click me!
        </span>
      </button>
    )
  }
}
```

How CSS-modules work

```
import styles from './Button.css'

/*
styles = {
  button: 'Button_button_3fslE',
  label: 'Button_label_I8bKh'
}
*/
```

How CSS-modules work

```
import styles from './Button.css'

/*
styles = {
  button: 'Button_button_3fslE',
  label: 'Button_label_I8bKh'
}
*/
```

name

How CSS-modules work

```
import styles from './Button.css'

/*
styles = {
  button: 'Button_button_3fslE',
  label: 'Button_label_I8bKh'
}
*/
```

name *local*

How CSS-modules work

```
import styles from './Button.css'

/*
styles = {
  button: 'Button_button_3fslE',
  label: 'Button_label_I8bKh'
}
*/
```

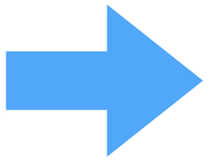
name *local* *hash*

How CSS-modules work

```
// webpack.config.js

module.exports = {
  module: {
    loaders: [
      {
        test: /\.css$/,
        loader: 'css?modules
            &localIdentName=[name]__[local]___[hash:base64:5]'
      }
    ]
  }
};
```

The result

```
<button class="button">
  <span class="label">
    Click me!
  </span>
</button>
```



```
<button class="Button_button_3fslE">
  <span class="Button_label_I8bKh">
    Click me!
  </span>
</button>
```

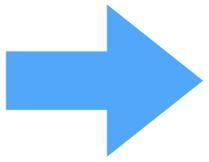
BEM for free!

Minification

```
// webpack.config.js

module.exports = {
  module: {
    loaders: [
      {
        test: /\.css$/,
        loader: 'css?modules&
          localIdentName=[hash:base64:4]'
      }
    ]
  }
};
```

Minification

```html
<button class="button">
  <span class="label">
    Click me!
  </span>
</button>
```



```html
<button class="1s6s">
  <span class="sdp9">
    Click me!
  </span>
</button>
```

Catch missed class names

```
.undefined {
  display: flex !important;
  position: fixed !important;
  top: 0 !important;
  right: 0 !important;
  bottom: 0 !important;
  left: 0 !important;
  background: red !important;
  color: white !important;
  justify-content: center !important;
  align-items: center !important;
  font-size: 30px !important;
  font-weight: bold !important;
}
.undefined::after {
  display: block !important;
  padding: 15px 30px !important;
  content: 'ERROR! You are missing a class definition in your css module!
  Inspect me to find out where.' !important;
}
```

http://davidwells.io/talks/react-css/#/47?_k=zlozk6

Problems with CSS solved with CSS-modules

+ Global namespace
+ Cascade
+ No styles isolation
+ Minification

CSS-modules overview

- Local by default but allow global exceptions
- Composition
- Explicit dependencies
- Works with pre-processors (Sass, less)
- Works with any server- or client-side framework

http://glenmaddern.com/articles/css-modules

Local by default but allow exceptions

```
.local-class {
  color: red;
}

:global(.prefix-modal-open) .local-class {
  color: green;
}
```

Composition

```
.common {
  /* all the common styles you want */
}
.normal {
  composes: common;
  /* anything that only applies to Normal */
}
.disabled {
  composes: common;
  /* anything that only applies to Disabled */
}
```

Explicit dependencies

```
.otherClassName {
  composes: className from "./style.css";
}
```

Import variables

```
/* variables.css */
@value small: (max-width: 599px);

/* component.css */
@value small from "./breakpoints.scss";

.pageContent {
  background: green;
  @media small {
    background: red;
  }
}
```

Use with pre-processors

```
:global {
  .this-is-global {
    color: yellow;
  }
  .potential-collision-city {
    color: crap;
  }
}
```

Works on server-side!

https://github.com/css-modules/postcss-modules

PostCSS

http://postcss.org

PostCSS - a tool for transfo

postcss.org

Setup Docs Learn Plugins Twitter Gitter Github

PostCSS

A tool for transforming CSS with JavaScript

Increase code readability

Add vendor prefixes to CSS rules using values from Can I Use. Autoprefixer will use the data based on current browser popularity and property support to apply prefixes for you.

CSS input

```
:fullscreen {
}
```

CSS output

```
:-webkit-:full-screen {
}
:-moz-:full-screen {
}
:full-screen {
}
```

CSS input

```
:root {
  --red: #d33;
}
a {
  &:hover {
    color: color(var(--red) a(54%));
  }
}
```

CSS output

```
a:hover {
  color: #dd3333;
  color: rgba(221, 51, 51, 0.54);
}
```

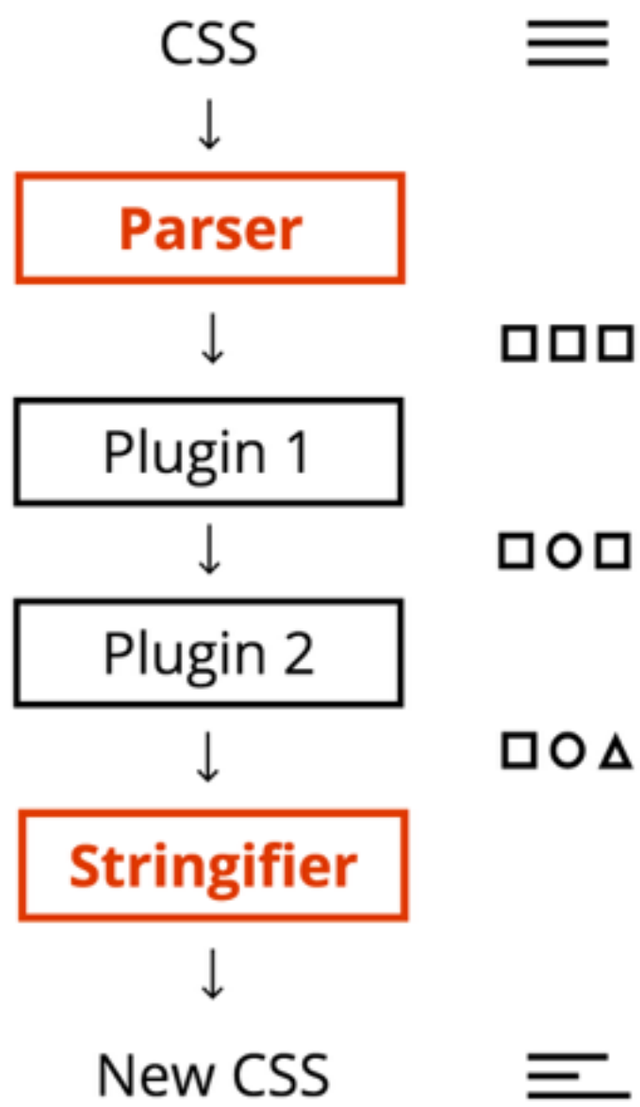
Use tomorrow's CSS, today!

Write future-proof CSS and forget old preprocessor specific syntax. Use the latest CSS syntax today with cssnext. It transforms CSS specs into more compatible CSS so you don't need to wait for browser support.

What is PostCSS?

Like Babel, but for CSS

How PostCSS works



What is possible?

Autoprefixer

```
:full-screen {
}
```

```
:-webkit-:full-screen {
}
:-moz-:full-screen {
}
:full-screen {
}
```

What is possible?

Sass/less syntax

```
.block {
    &_title {
        font-size: 200%;   }
    }
}
```

```
.block_title {
    font-size: 200%;
```

What is possible?

Fallbacks

```
.foo {
  opacity: 0.8;
}
```

```
.foo {
  opacity: 0.8;
  filter: alpha(opacity=80)\9;
}
```

What is possible?

CSS
MODULES

CSS-modules

```
.name {
  color: gray;
}
```

```
.Logo__name__SVK0g {
  color: gray;
}
```

What is possible?

CSS next

```
:root {
  --red: #d33;
}
a {
  &:hover {
    color: color(var(--red) a(54%));
  }
}
```

```
a:hover {
  color: #dd3333;
  color: rgba(221, 51, 51, 0.54);
}
```

What is possible?

Stylelint

```
a {
  color: #d3;
}
```

```
app.css
2:10 Invalid hex color
```

Example setup with Webpack

```
module.exports = {
  module: {
    loaders: [
      {
        test: /\.css$/,
        loader: 'style-loader!css-loader!postcss-loader'
      }
    ]
  },
  postcss: function() {
    return [
      require('precss'),
      require('postcss-calc'),
      require('autoprefixer')({ browsers: ['last 2 version'] })
    ];
  }
};
```

Lint CSS as pre-commit hook

```
{
  "lint-staged": {
    "*.css": "stylelint"
  },
  "pre-commit": “lint-staged",
  "stylelint": {
    "extends": "stylelint-config-standard"
  }
}
```

https://github.com/okonet/lint-staged

http://postcss.parts/

by @mxstbr

PostCSS overview

+ More powerful than pre-processors (AST 💪!)
+ ⚡Fast (a few times faster than Sass)
+ Lots of plugins
+ Tools like linting or automatic properties sorting
- No IDE support
- No shared variables

Inline styles in JSX

+ JS is more powerful
+ Runtime re-calculation
+ Better dead code elimination

- No pseudo-selectors / media queries
- React.js-only
- Harder to debug in DevTools
- No IDE support
- No related tools

CSS-in-JS

+ JS is more powerful language
+ Shared variables
+ Better dead code elimination

- No sourcemaps
- No IDE support
- No related tools

Do I need it?

- ✓ You work in a team
- ✓ You plan to update your CSS in the future
- ✓ You use third-party CSS (Bootstrap etc.)
- ✓ You care about code readability and reusability
- ✓ You hate manual work

Thank you!

Andrey Okonetchnikov

Front-End Engineer

@okonetchnikov

http://okonet.ru

https://github.com/okonet

andrey@okonet.ru