CSS Modules, my dream CSS-in-JS solution

AMAZEE WEBINAR #3 / 29 March 2019

CSS Modules,
my dream CSS-in-JS solution

TALKING POINTS

John Albin Wilkins

Senior Front-end Developer

2017	Amazee Labs
2004	Drupal
1997	CSS
1993	Web

I ❤️ Sass

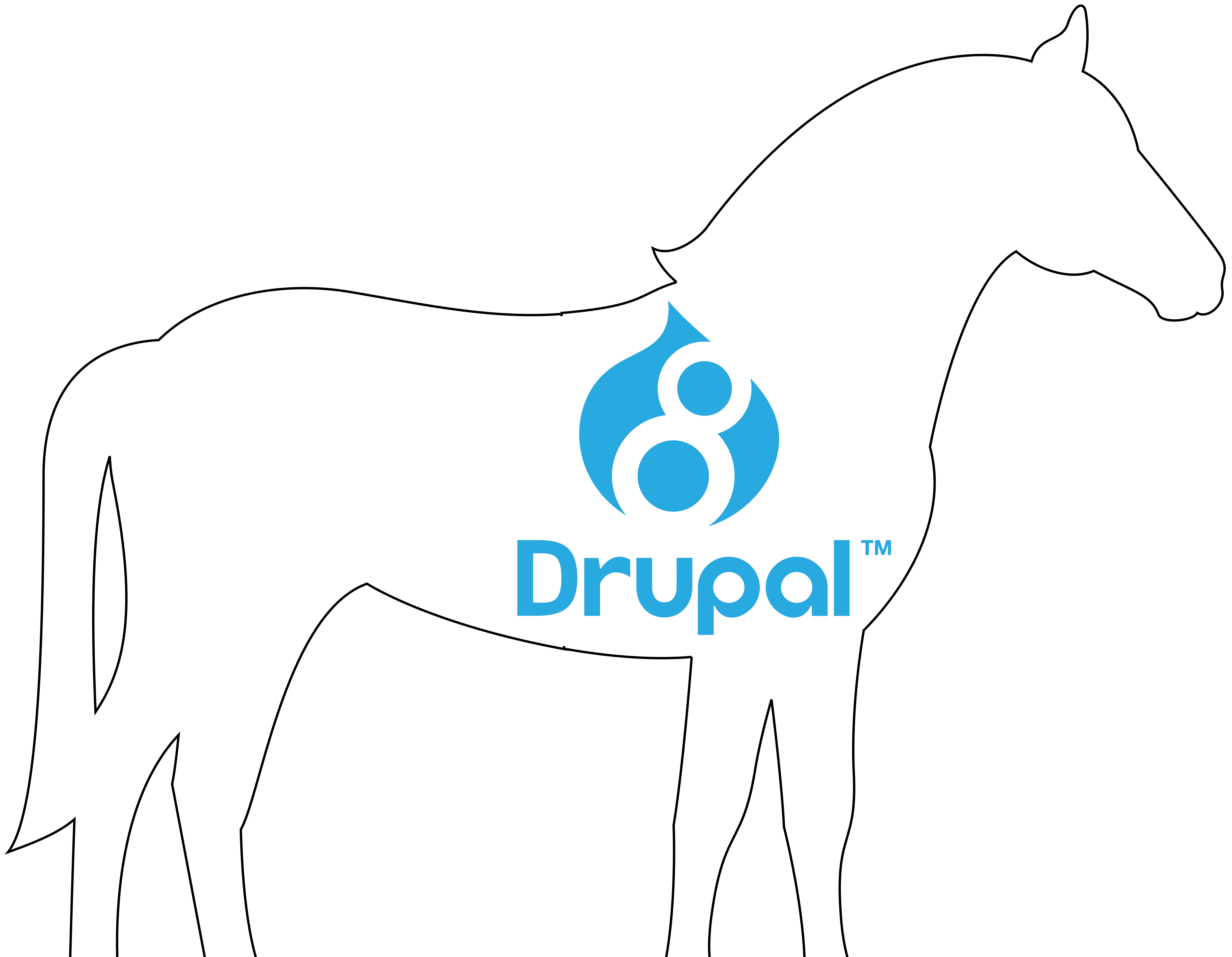
@JohnAlbin, 2011

I ❤️'d Sass

@JohnAlbin, 2017

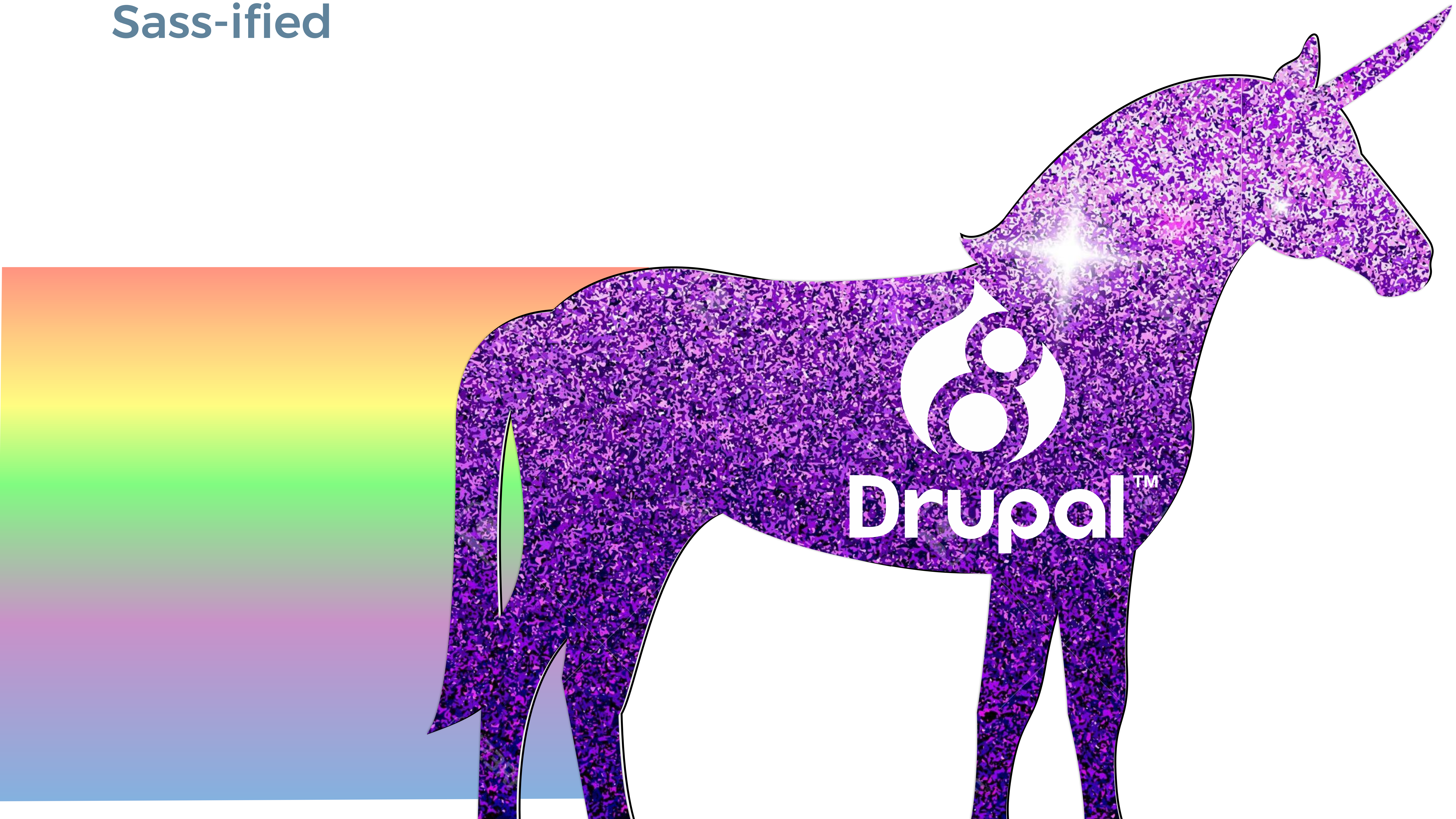
TRADITIONAL WEBSITE

ALL-IN-ONE



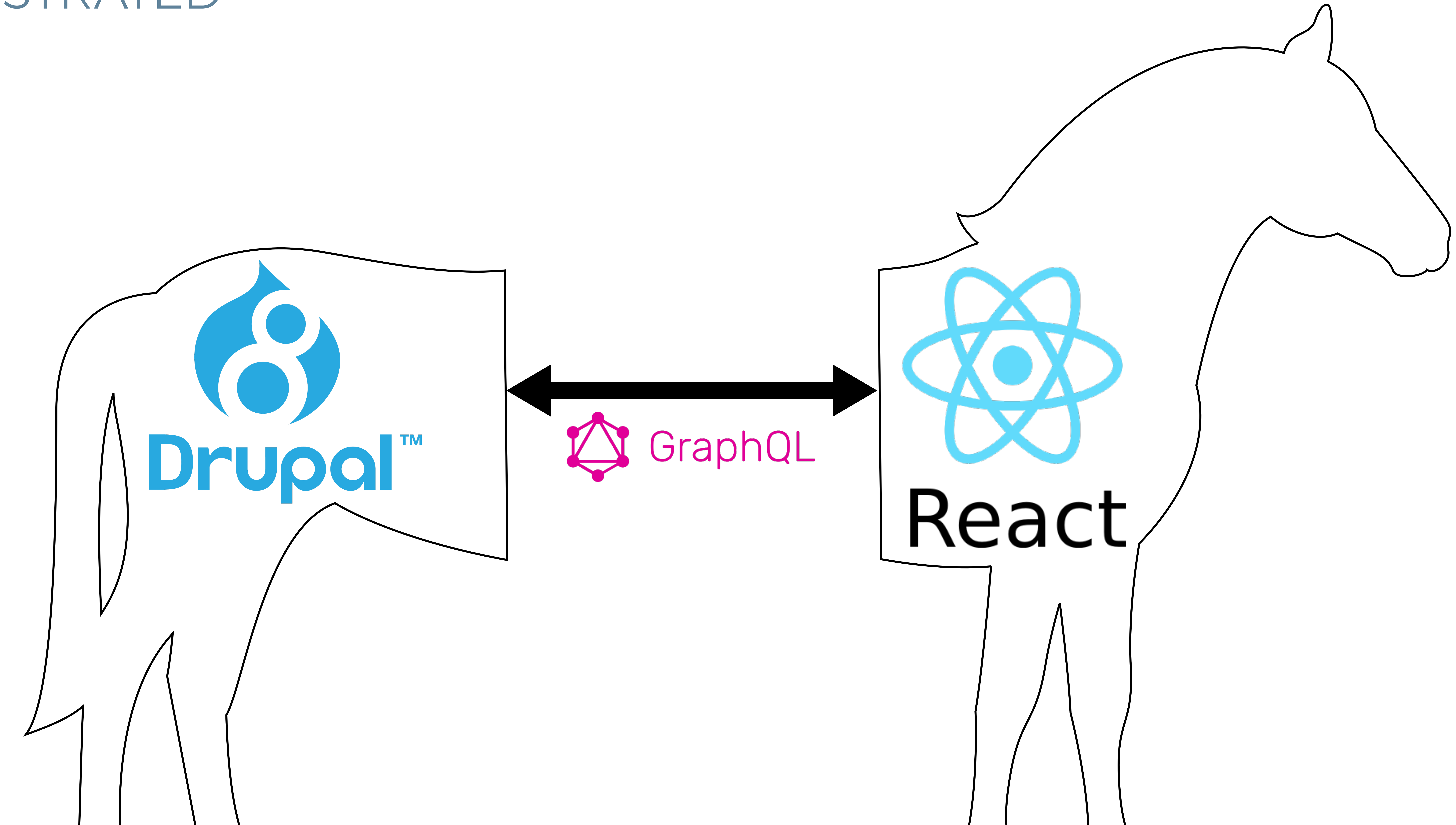
TRADITIONAL WEBSITE

Sass-ified



DECOUPLED WEBSITE

ILLUSTRATED



LET'S EXPLORE CSS-IN-JS

68+ CSS-IN-JS PROJECTS:

CHOOSE ONE

superstyle
glamorous
stile
react-fela
react-cssom
styling
i-css
style-it
cssobj
glamor
stilr
reactcss
css-loader
react-style
Aphrodite
css-ns
csjs
react-inline-style
babel-plugin-css-in-js
react-native-web
typestyle
react-jss
styled-jsx
react-stylematic
uranium
electron-css
restyles
emotion
react-free-style
styled-components
styletron-react
cssx
es-css-modules
react-look
classy
react-inline-css
j2c
react-statics-styles
scope-styles
react-styl
babel-plugin-pre-style
react-styled
freestyler
react-inline
react-css-components
react-css-modulescss-light
smart-css
cxs
react-vstyle
css-constructor
react-css-builder
hyperstyles
react-theme
react-styleable
stylable
hiccup-css

3 API APPROACHES:

CHOOSE ONE

- CSS in a **object literal**	**(JavaScript Object)**
- CSS in a **template literal**	**(JavaScript String)**
- CSS in a **file**	**(CSS File)**

CSS IN A OBJECT LITERAL

```
// MyComponent.js
import { apiFunc } from 'some-project';

const styles = {
  base: {
    color: '#fff',
    ':hover': {
      backgroundColor: '#0074d9',
    },
  },
  warning: {
    'background-color': someValue(),
  },
};
```

```
// MyComponent.js (cont.)
export const MyComponent = () => (
  <div className={
    apiFunc(styles.base, styles.warning)
  }>
    Some content
  </div>
);
```

CSS IN A TEMPLATE LITERAL

```
// MyComponent.js
import { apiFunc } from 'some-project';

const styles = apiFunc`
  .base {
    color: #fff;

    &:hover {
      background-color: #0074d9;
    }
  }

  .warning {
    background-color: ${someValue()},
  }
`;
```

```
// MyComponent.js (cont.)
export const MyComponent = () => (
  <div className={
    `${styles.base} ${styles.warning}`
  }>
    Some content
  </div>
);
```

CSS IN A FILE

```css
/* styles.css */
.base {
  color: #fff;

  &:hover {
    background-color: #0074d9;
  }
}

.warning {
  background-color: var(--imported-value),
}
```

```js
// MyComponent.js
import styles from './styles.css';
import classNames from 'classnames';

export const MyComponent = () => (
  <div className={
    classNames(styles.base, styles.warning)
  }>
    Some content
  </div>
);
```

FEATURE COMPARISON

EVALUATION CRITERIA

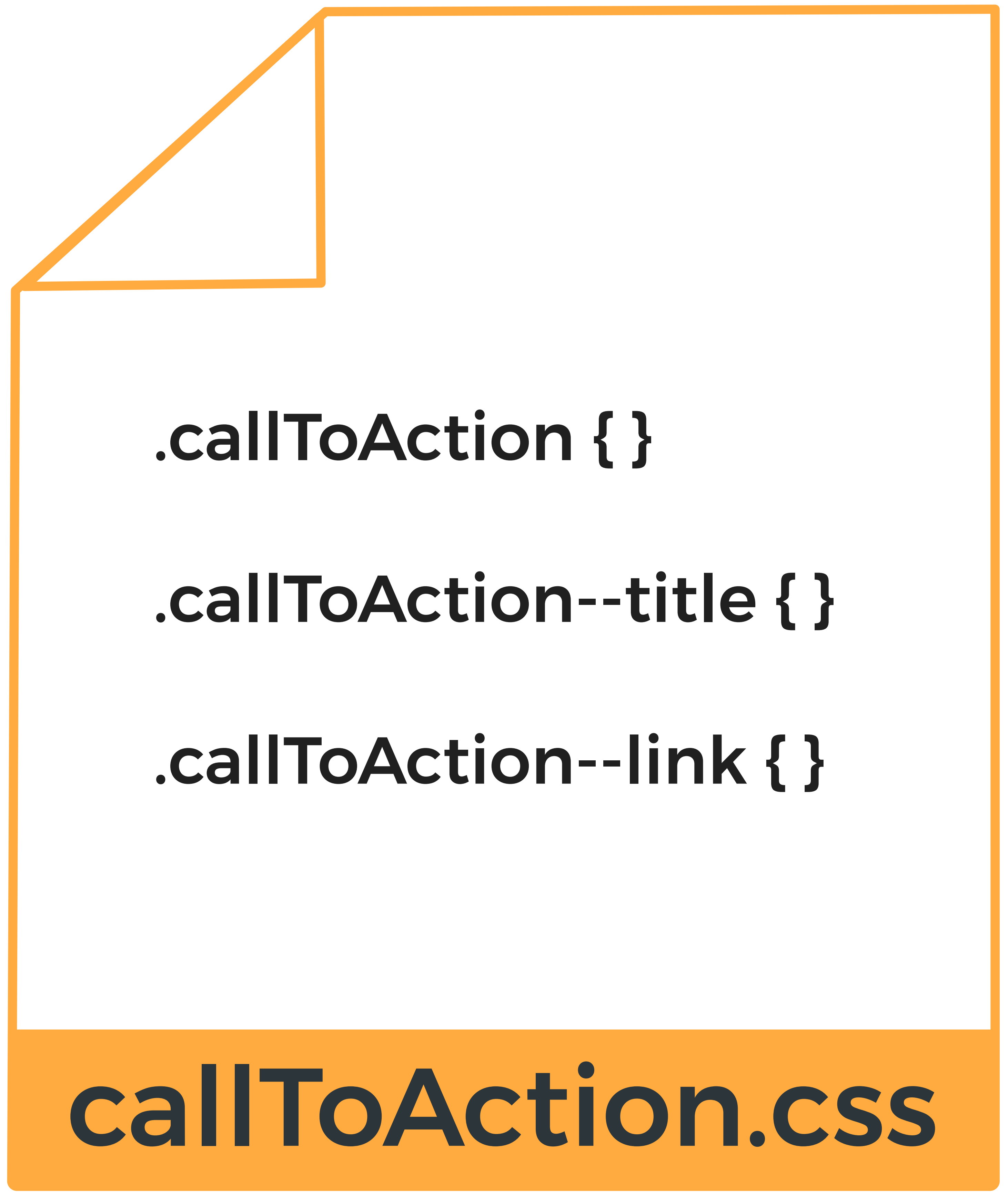
	locally scoped class	cross-component composition	dead code elimination	nested rulesets	shared JS / CSS variables	multi-platform
CSS in object literal	✓	✓	✓	✓	✓	JS projects only
CSS in template literal	✓	✓	✓	✓	✓	JS projects only
CSS in file	✓	✓	✓	✓	✓	😎

CSS MODULES

Amazee Labs

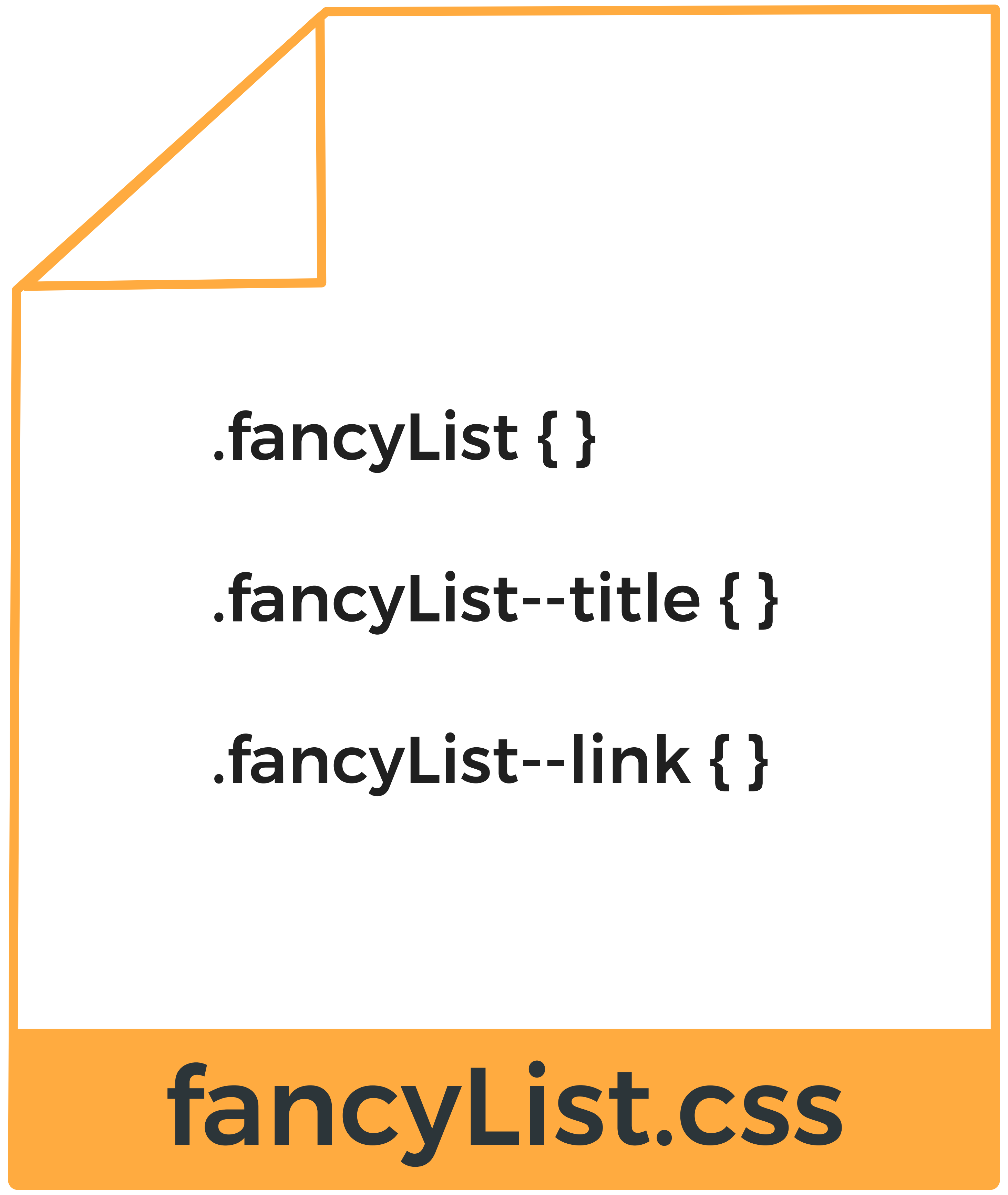
LOCALLY-SCOPED CLASS NAMES

OLD RULES FOR BEM NAMING



```
.callToAction { }

.callToAction--title { }

.callToAction--link { }
```

callToAction.css

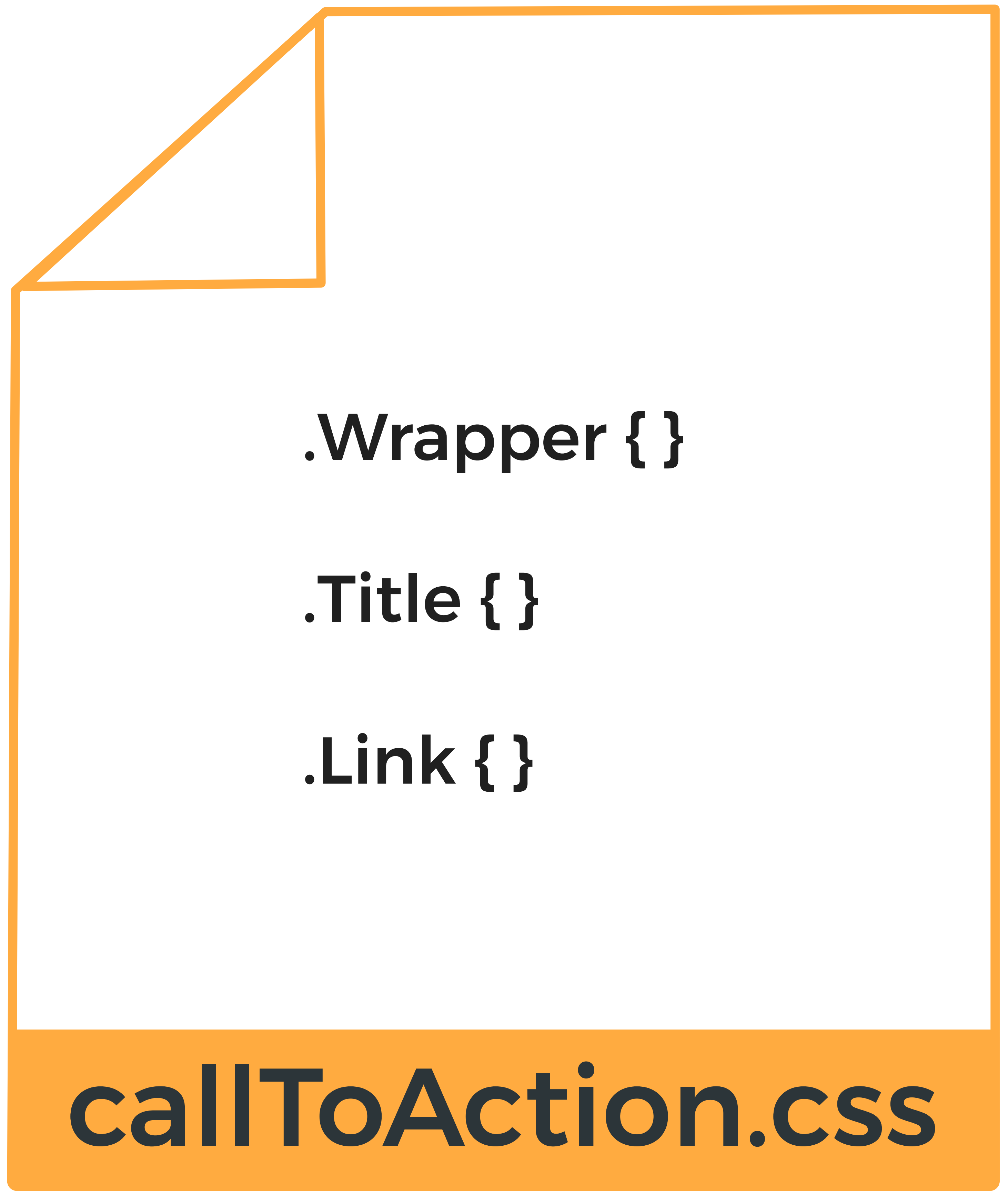


```
.fancyList { }

.fancyList--title { }

.fancyList--link { }
```

fancyList.css

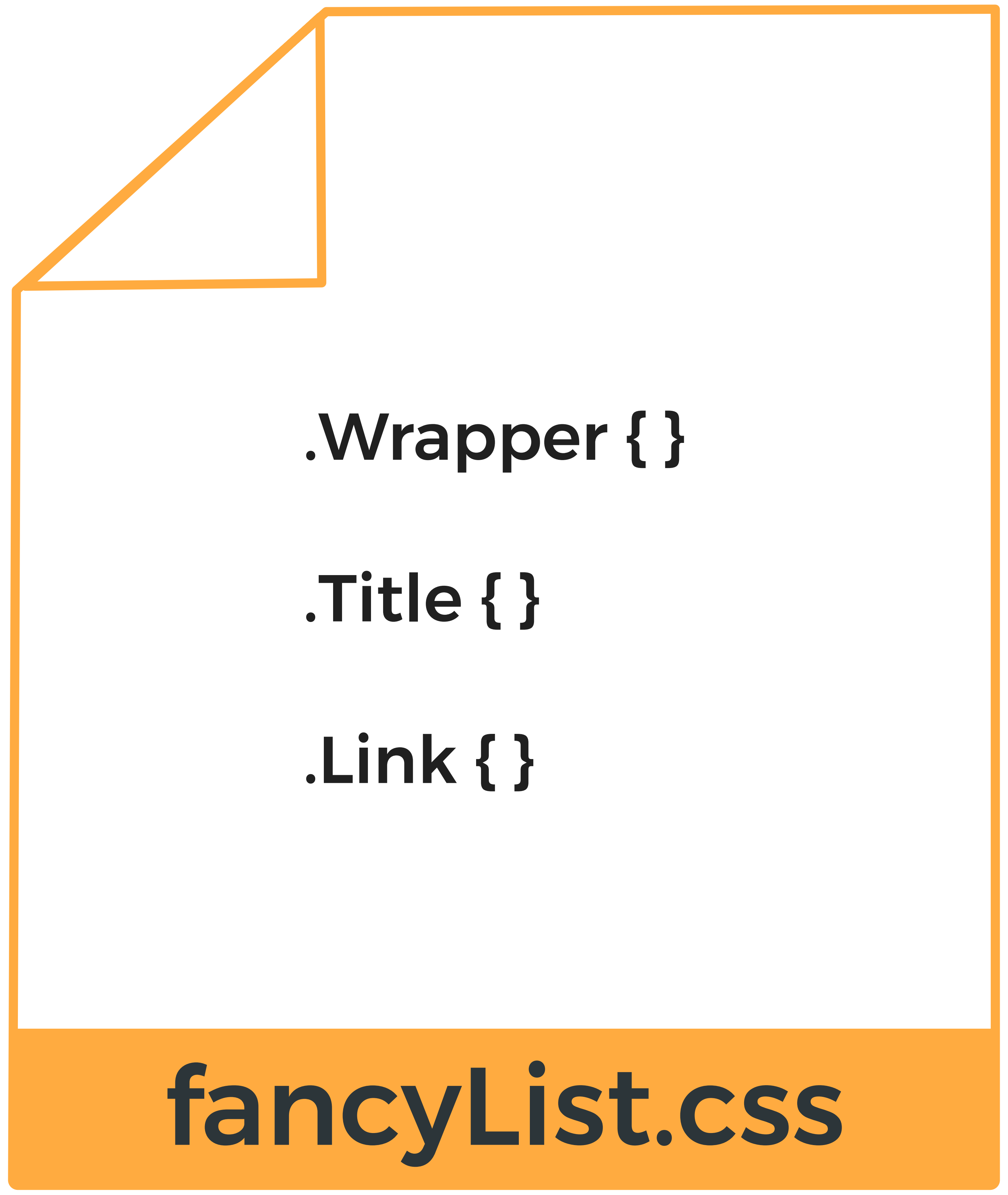
LOCALLY-SCOPED CLASS NAMES

NO MORE BEM RULES



```
.Wrapper { }

.Title { }

.Link { }
```

callToAction.css

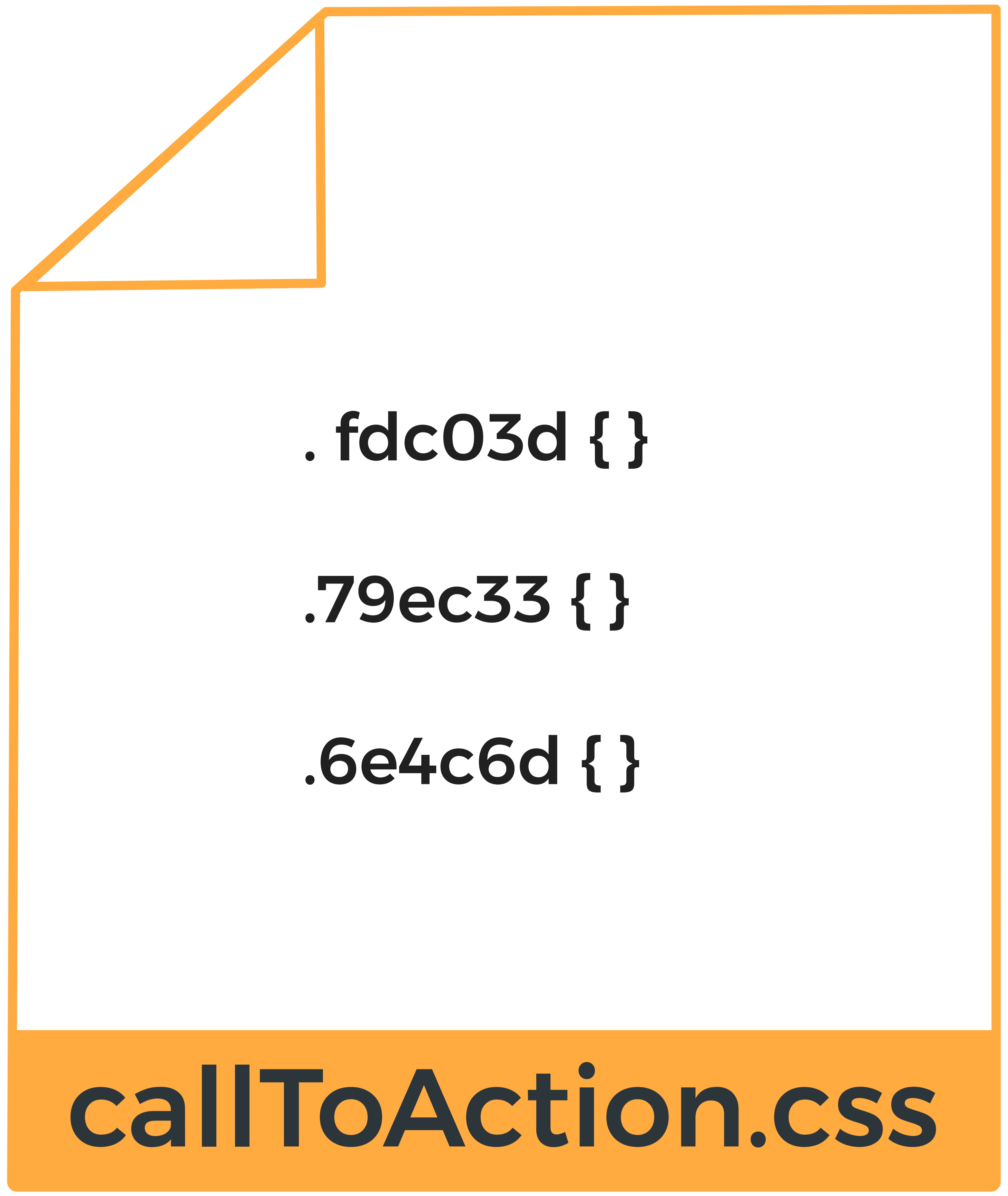


```
.Wrapper { }

.Title { }

.Link { }
```

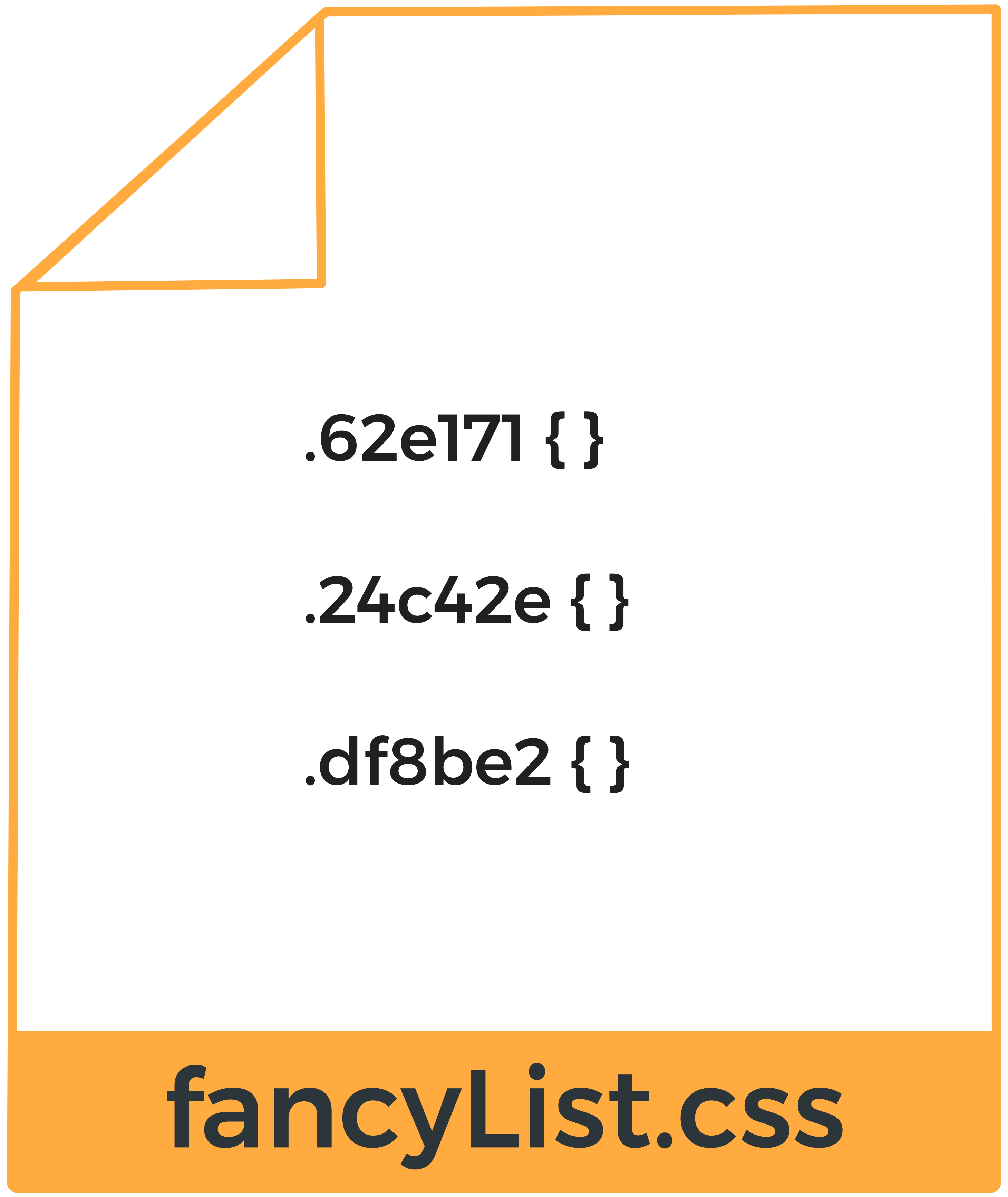
fancyList.css

LOCALLY-SCOPED CLASS NAMES

i.e. ENSURE UNIQUE GLOBAL CLASS NAMES



```
. fdc03d { }

.79ec33 { }

.6e4c6d { }
```



```
.62e171 { }

.24c42e { }

.df8be2 { }
```

LOCALLY-SCOPED CLASS NAMES

MAPPING KNOWN CLASS NAMES TO AUTO-GENERATED ONES

```
// callToAction.js
import styles from './styles.css';
import classNames from 'classnames';

export const CallToAction = () => (
  <article className={styles.Wrapper}>
    <h3 className={styles.Title}>
      Look at me.
    </h3>
    <p>
      <a link="#" className={styles.Link}>
        Now do this.
      </a>
    </p>
    Some content
  </article>
);
```

```
// JS file auto-generated by Webpack
const styles = {
  Wrapper: 'fdc03d',
  Title: '79ec33',
  Link: '6e4c6d',
};

export default styles;
```

CROSS-COMPONENT COMPOSITION

SHARING CSS ACROSS COMPONENTS

```css
.TimeStamp {
  /* Add this class from another file. */
  composes: TimeStamp from '../BlogPost/styles.css';

  /* properties here */
}

.List {
  // Assume this class exists in the global space.
  composes: ListReset from global;

  /* properties here */
}

:global(.Loader) {
  /* properties here */
}
```

```js
// JS file auto-generated by Webpack
const styles = {
  // Includes "79ec33" class from BlogPost.
  TimeStamp: 'fdc03d 79ec33',

  // Includes "ListReset" class.
  List: '6e4c6d ListReset',

  // No class name transforms.
  Loader: 'Loader'
};

export default styles;
```

DEAD-CODE ELIMINATION

AUTOMATIC REMOVAL OF UNUSED CSS

- When you add a component to a page,
  it adds the HTML, CSS, JS, etc. to the page.

- When you DO NOT add a component to a page,
  the HTML, CSS, JS, etc. is NOT added to the page.

- The **eslint-plugin-css-modules** plugin
  will find unused class names in your component.

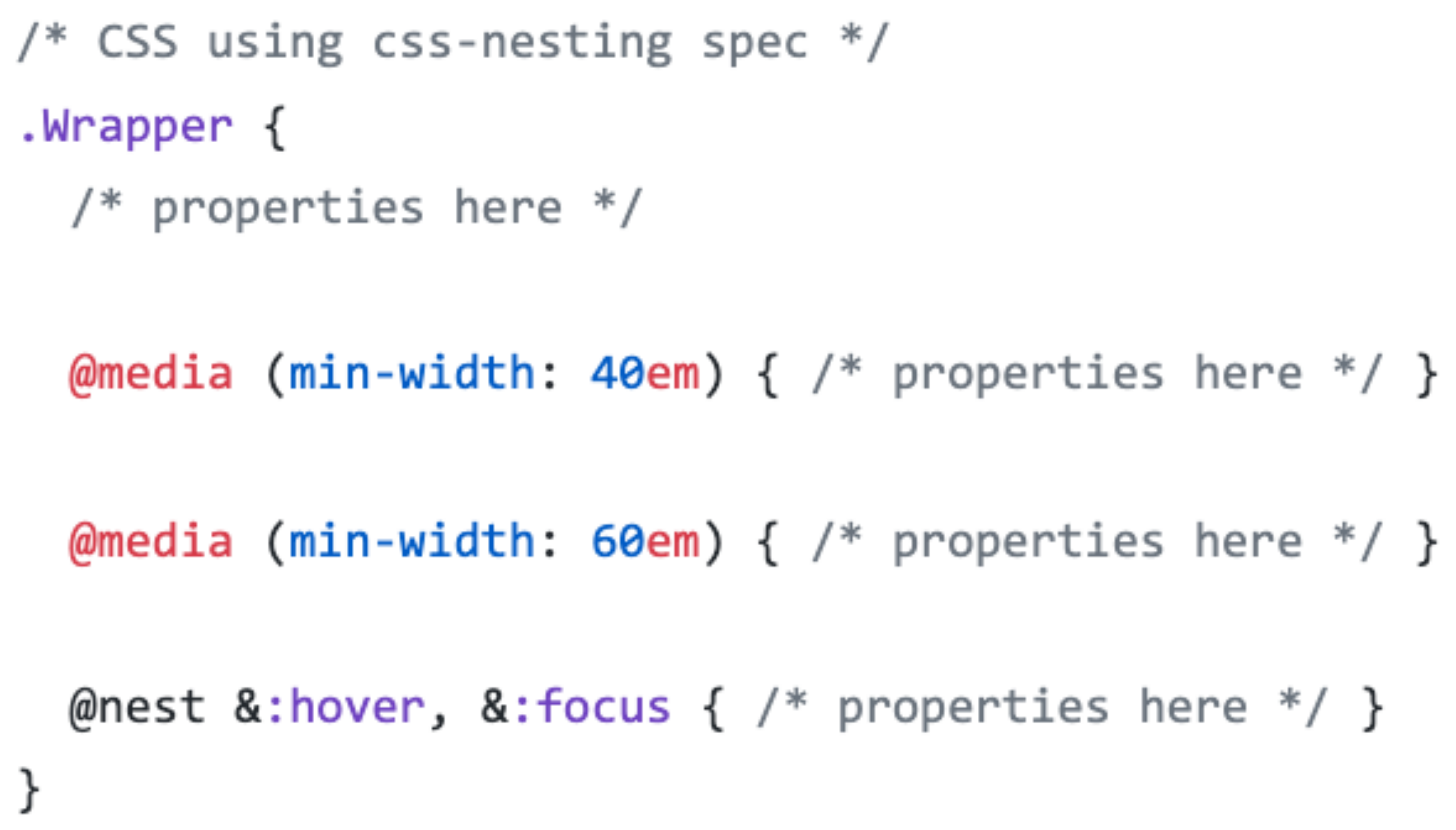
NESTED RULESETS

NO MORE DUPLICATE SELECTORS

```css
/* Old-school CSS */
.Wrapper { /* properties here */ }

@media (min-width: 40em) {
  .Wrapper { /* properties here */ }
}

@media (min-width: 60em) {
  .Wrapper { /* properties here */ }
}

.Wrapper:hover, .Wrapper:focus { /* properties here */ }
```

NESTED RULESETS

NO MORE DUPLICATE SELECTORS



```css
/* CSS using css-nesting spec */
.Wrapper {
  /* properties here */

  @media (min-width: 40em) { /* properties here */ }

  @media (min-width: 60em) { /* properties here */ }

  @nest &:hover, &:focus { /* properties here */ }
}
```

https://drafts.csswg.org/css-nesting-1

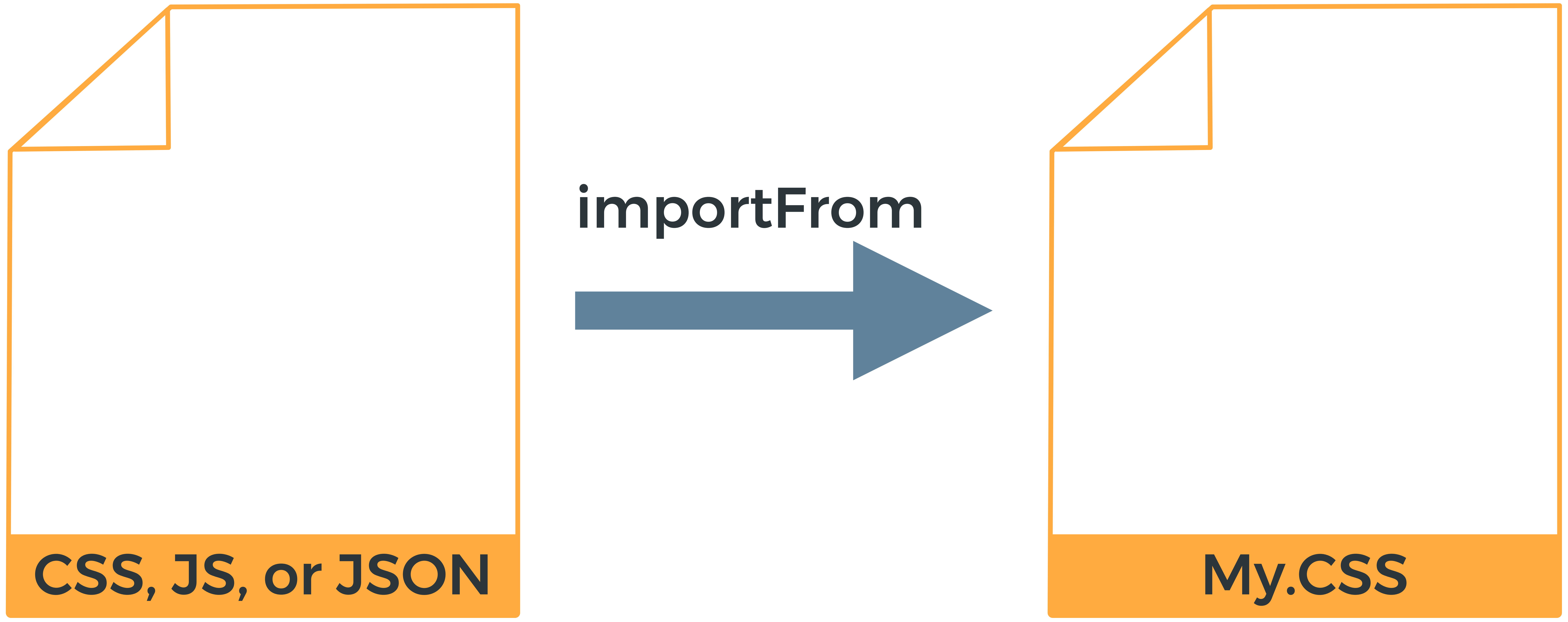
WHY @NEST IS NEEDED

BROWSER PERFORMANCE AND LOOK-AHEAD PARSING

```css
/* What if we tried Sass-style nesting? */
.Wrapper {
  html.js & { /* Won't work! */
    /* properties here */
  }
}

/* The css-nesting spec does allow
   simple Sass-style nesting */
.Wrapper {
  & span { /* Works! */
    /* properties here */
  }
}
```

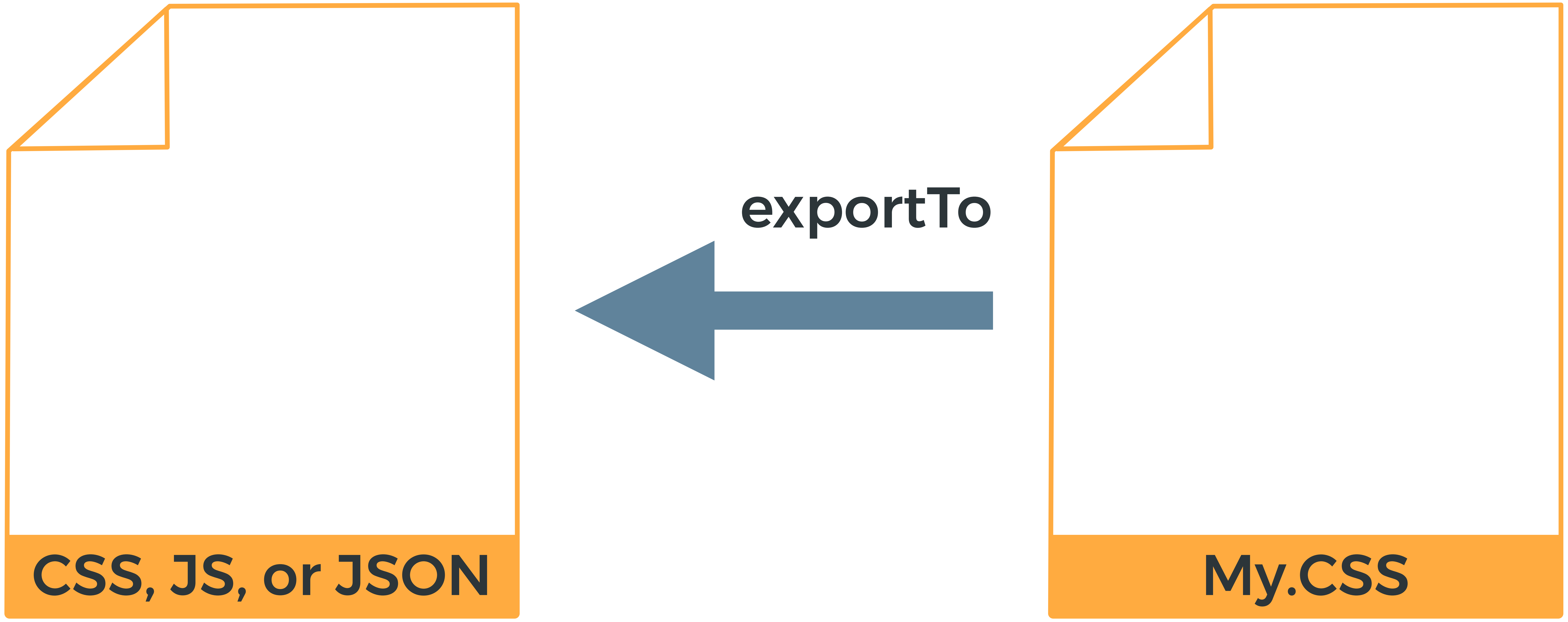
SHARING CSS / JS VARIABLES

WRITE VALUES ONCE, USE THEM EVERYWHERE



SHARING CSS / JS VARIABLES

WRITE VALUES ONCE, USE THEM EVERYWHERE



SHARING CSS / JS VARIABLES

WRITE VALUES ONCE, USE THEM EVERYWHERE

```css
/* These values can be exported to CSS, JS, and JSON files. */

:root {
  --primary-color: red;
}

@custom-media --tablet (min-width: 30em);

@custom-selector :--headings h1, h2, h3, h4, h5, h6;
```

MULTI-PLATFORM SUPPORT

EVERY PLATFORM WORKS WITH CSS FILES

ALL OF THEM

CSS Modules

= webpack loader plugin

+ PostCSS

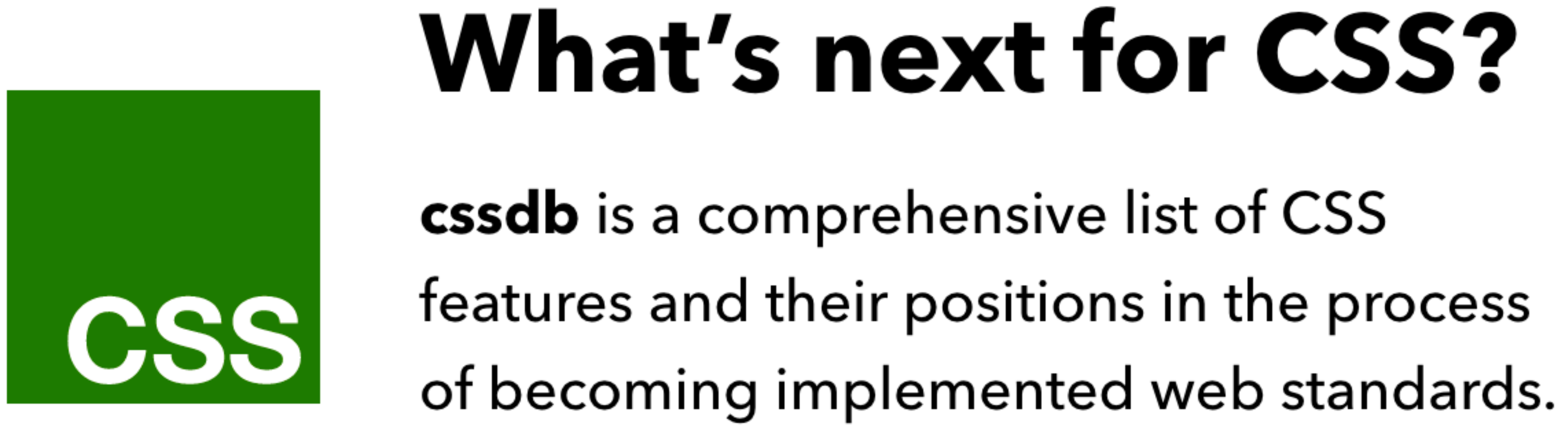
POSTCSS.CONFIG.JS

AMAZEE'S CONFIGURATION (WORK IN PROGRESS)

```
// postcss.config.js
const path = require('path');

module.exports = {
  plugins: {
    'postcss-preset-env': {
      stage: 1,
      browsers: '> 0.5%, last 2 versions, Firefox ESR, not dead',
      exportTo: path.resolve('cssValues.json');
    },
  }
};
```

EXPERIMENTAL CSS, TODAY

postcss-preset-env



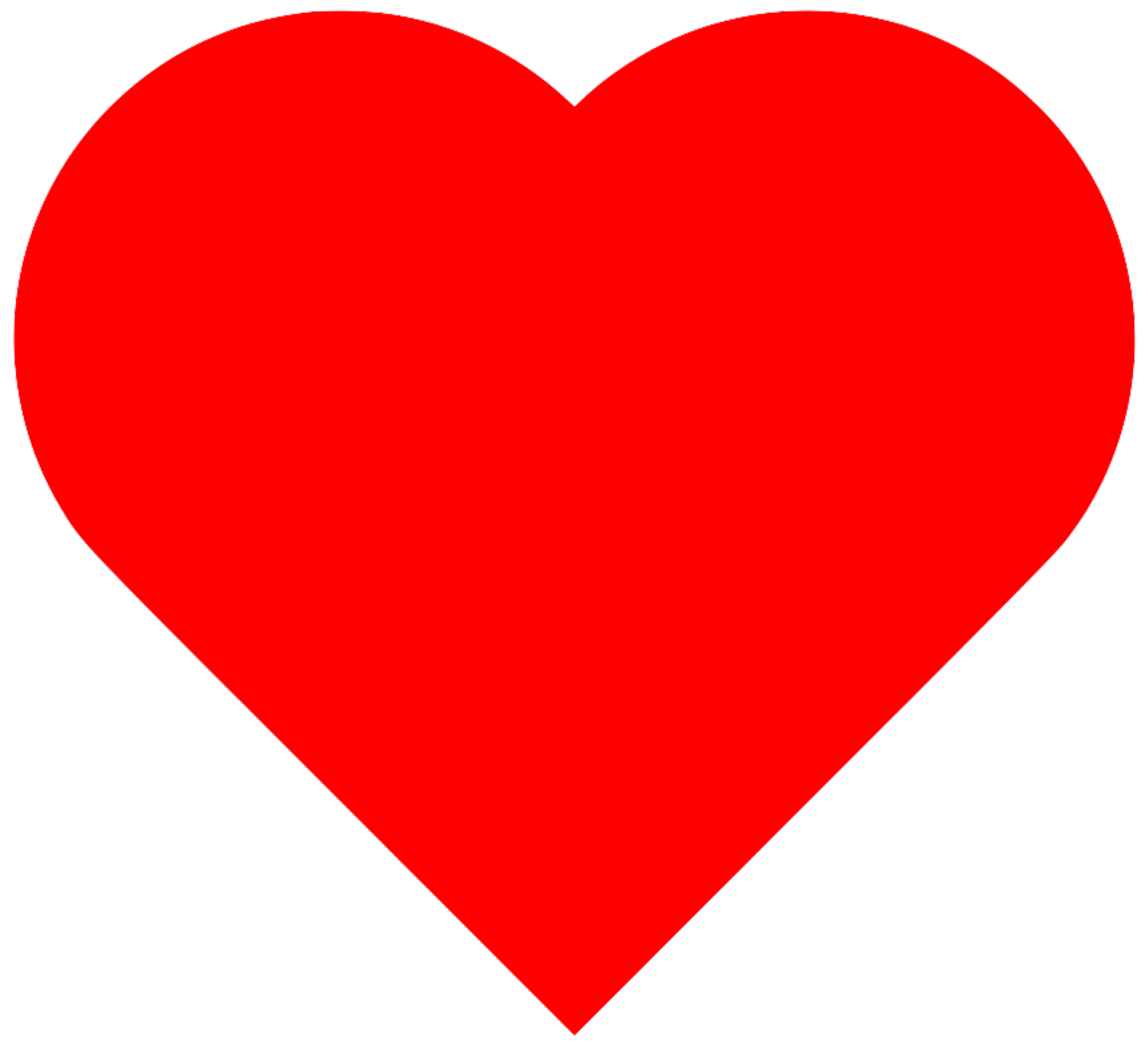
What’s next for CSS?

cssdb is a comprehensive list of CSS features and their positions in the process of becoming implemented web standards.

https://cssdb.org

I ❤ CSS

@JohnAlbin, 2019

I  CSS

```
:root {
  --heart-size: 20vw;
}

.heart {
  display: inline-block;
  margin-top: calc( var(--heart-size) / 5 );
  margin-left: calc( var(--heart-size) / 4 );
  margin-right: calc( var(--heart-size) / 7 );
  background-color: red;
  height: var(--heart-size);
  transform: rotate(-45deg);
  width: var(--heart-size);

  @nest &::before, &::after {
    content: "";
    background-color: red;
    border-radius: 50%;
    height: var(--heart-size);
    position: absolute;
    width: var(--heart-size);
  }
  @nest &::before {
    top: calc( -1 * var(--heart-size) / 2 );
    left: 0;
  }
}
```

https://codepen.io/johnalbin/pen/jJJWmo

THANK YOU

JOHN ALBIN WILKINS

Senior Front-end Developer

e john.albin@amazee.com