sCALABLE

.css -> .module.css -> .js -> .ts

I Am Devloper
@iamdevloper

CSS is easy. It's like riding a bike, which is on fire and the ground is on fire and everything is on fire because it is hell.

1:23 AM · Jul 15, 2016

3.5K 2.3K

CSS ZEN GARDEN

csszengarden.com

PLAIN CSS WORKS GREAT FOR

SMALL WEBSITES / APPS

PROBLEM 1

ZOMBIE CODE

```
// index.html

<p class="tagline">
  Great tagline text!
</p>
```

```
// styles.css

.tagline {
  font-size: 2rem;
}
```

```
// index.html

<p class="tagline">
  Great tagline text!
</p>

<button class="tagline_link">
  Check this out
</button>
```

```
// styles.css

.tagline {
  font-size: 2rem;
}

.tagline_link {
  color: blue;
}
```

```
// index.html

<p class="tagline">
  Great tagline text!
</p>
```

```
// styles.css

.tagline {
  font-size: 2rem;
}

.tagline_link {
  color: blue;
}
```

"nobody" cleans up CSS

PROBLEM 2

CLASS NAMING

```css
// styles.css
.overlay {
  background: transparent;
}

// other.css
.overlay {
  background: rgba(0, 0, 0, .2);
}
```

name collisions
no tooling to prevent it

```
// styles.css
.my-application-overlay {
  background: transparent;
}

// other.css
.overlay {
  background: rgba(0, 0, 0, .2);
}
```

"namespacing"

PROBLEM 3

SPECIFICITY

specificity wars

```
// styles.css


.product .title {}

.product .title.discount {}

#promo .product .title.discount {}

.dark-theme #promo .product .title.discount {}

.special.title { color: blue !important; }
```

```
// products.css
.product .title {
  font-weight: bold;
}

// cart.css
#cart .title {
  font-weight: normal;
}
```

unexpected override

PROBLEM 4

SOURCE ORDER

```html
// index.html

<p class="red blue">
  BLUE TEXT
</p>
```

```css
// styles.css

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

BLUE TEXT

```html
// index.html

<p class="red blue">
  BLUE TEXT
</p>
```

```css
// styles.css

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

⚠️ ***Zombie code***

⚠️ ***Class naming***

⚠️ ***Specificity wars***

⚠️ ***Source order problems***

OOCSS, SMACCS, BEM, ACSS

METHODOLOGIES / ARCHITECTURES

UTILITY FIRST

FRAMEWORKS

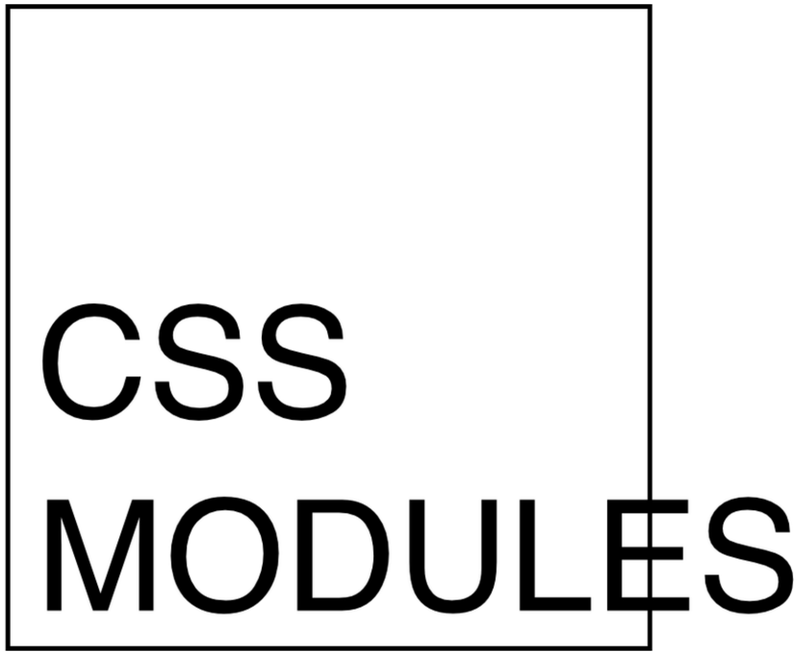
[e-spres-oh] BEM

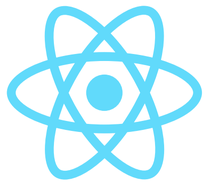
Methodologies are impossible to enforce through tooling

Scoped CSS

```
<style scoped>
  .title {}
</style>
```

Abandoned standard


CSS
MODULES



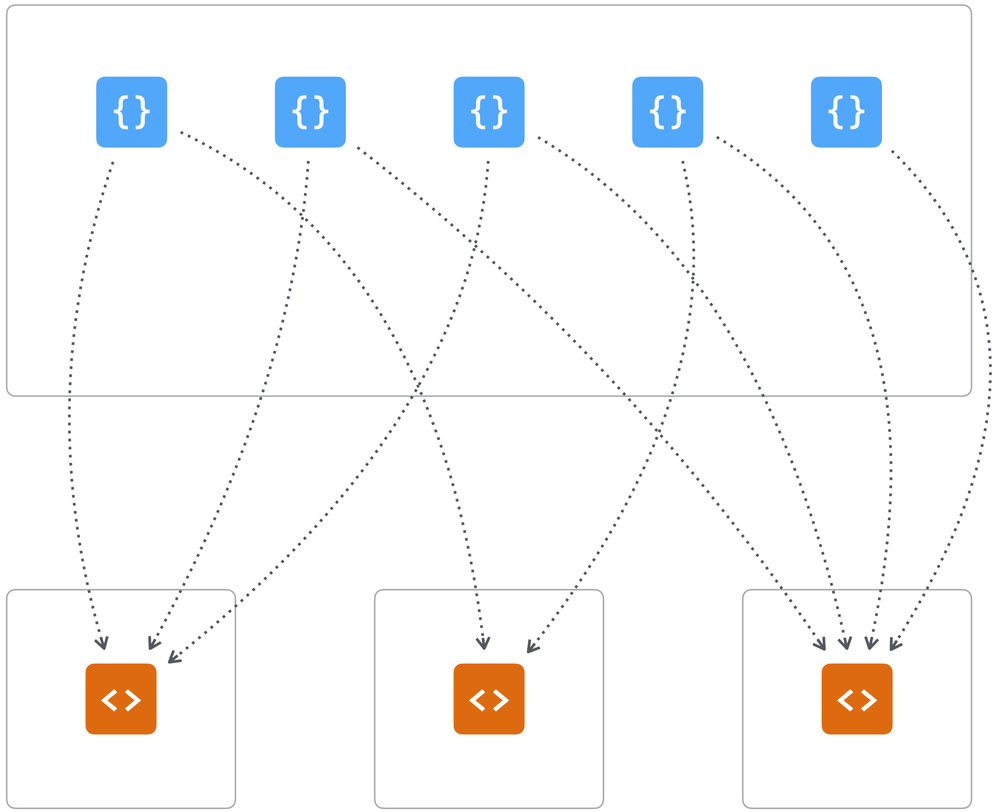
CSS Modules

(optional)

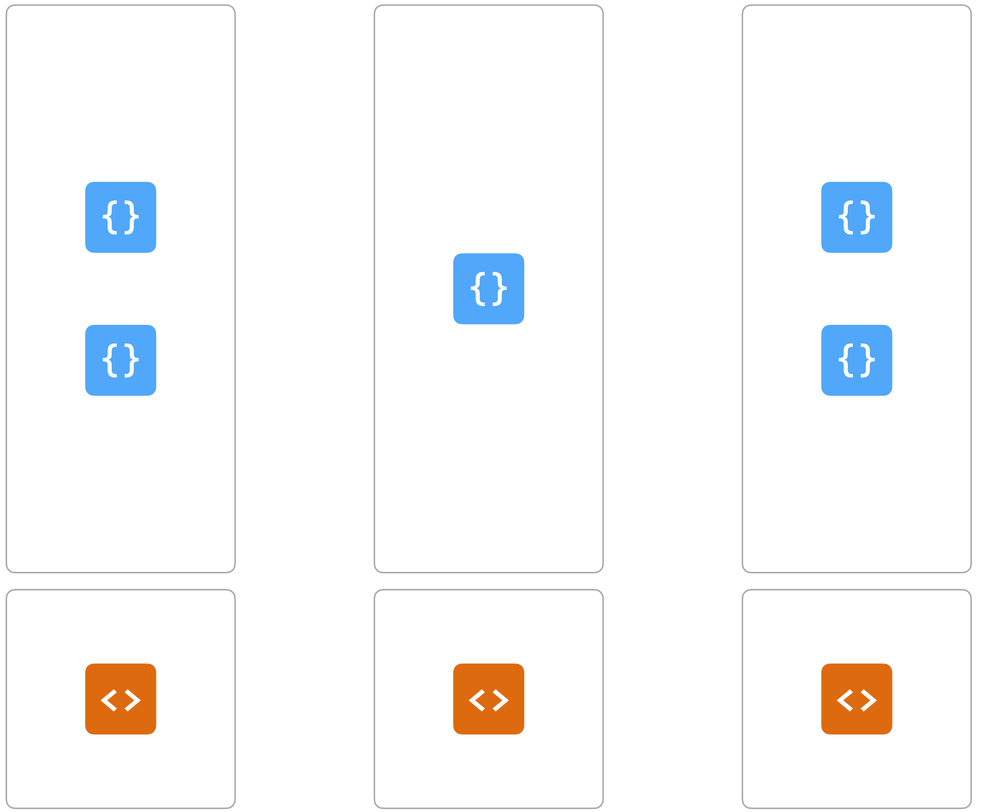
Emulated
Scoped CSS

Emulated
Scoped CSS

SINGLE CSS FILE



ONE CSS FILE / COMPONENT



✅ ***No zombie code***

✅ ***Class names***

✅ ***No source order problems***

✅ ***No specificity wars***

LIMITATIONS

(after working with React Native & TypeScript)

LIMITATION 1

NO TYPE-SAFETY

```
// styles.css
.product_title {}

.....................................................

// products.html
<section>
  <h1 class="product_title"></h1>
</section>
```

Cumbersome refactoring
(no intellisense / go to definition)

LIMITATION 2

CONTEXTUAL STYLES

```
// styles.css

.product_title {}

.product_title:hover {}
.product_title::after {}

@media (min-width: 768px) {
  .product_title {}
}
```

code duplication

```
// styles.css

23|   .product_title {}
...
62|   @media (min-width: 768px) {
63|     .product_title {}
64|   }
...
90|   @media (min-width: 1280px) {
91|     .product_title {}
92|   }
```

disconnected

```scss
// styles.scss

.product_title {
  &:hover {}
  &::after {}
  @media (min-width: 768px) {}
  @media (min-width: 1280px) {}
}
```

DRY & Contextual

LIMITATION 3

SHARED DESIGN TOKENS

Design tokens

are all the values needed

to construct and maintain

a design system

DESIGN TOKENS

```
// color palette
Primary color -> #ffaa00
Accent  color -> #e0c008
...

// breakpoints
// spacing values
// typography
...
```

Why shared?

1. Presentation logic is in JS
2. Type-safety

WHERE TO STORE DESIGN TOKENS ?

1

```
// CSS custom properties
:root { --color-primary: #ffaa00; }
```

2

```
// SCSS/LESS/Stylus variables
$color-primary: #ffaa00;
```

3

```
// JS constants
const color_primary = "#ffaa00";
```

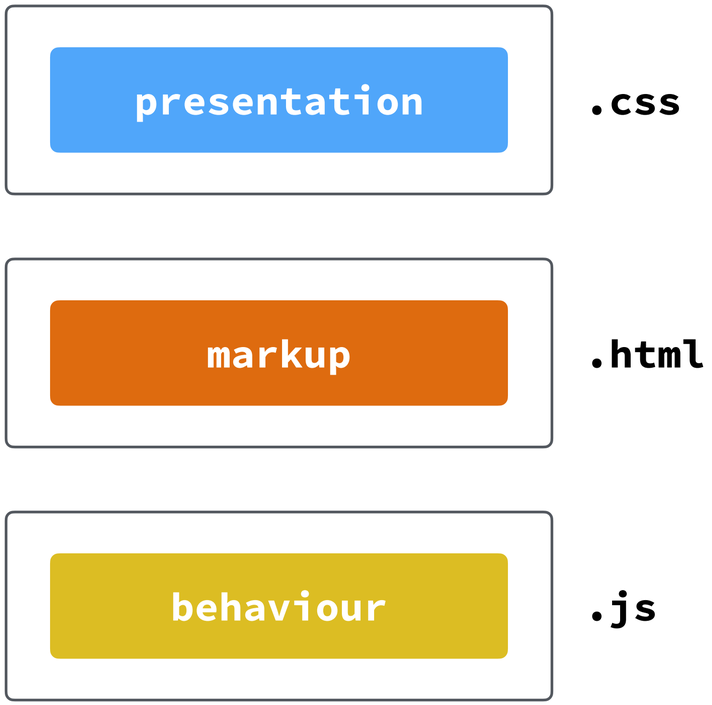
CSS variables

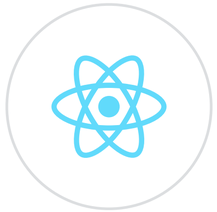
are not type-safe

⚠️ ***No type-safety***

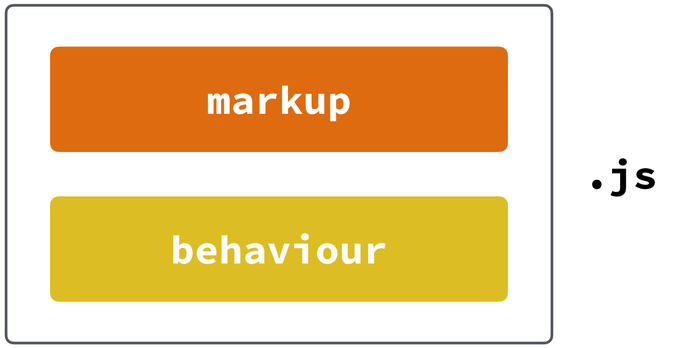
⚠️ ***Contextual styles***

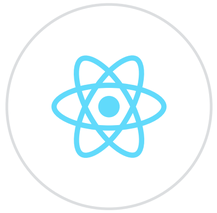
⚠️ ***Cumbersome shared tokens***

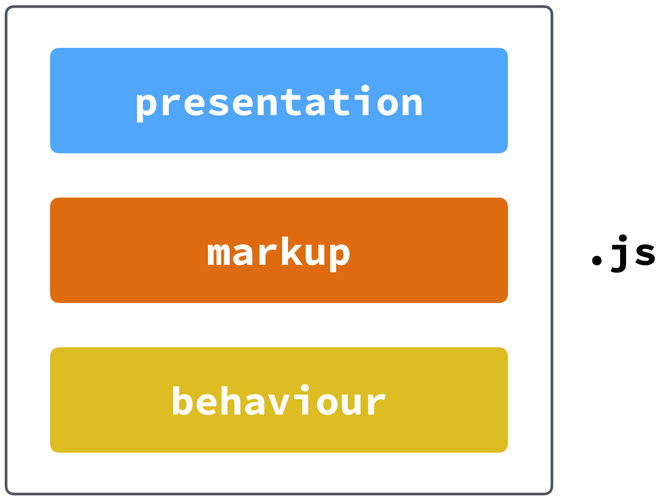

presentation
.css
markup
.html
behaviour
.js




presentation
.css


markup
behaviour
.js





.js

CSS IN JS

SINCE 2014

✅ ***Type-safety***

✅ ***Contextual styles***

✅ ***Trivial shared tokens***

CSS IN JS

A THOROUGH ANALYSIS

CSS IN TS

✅ *Type-safety*

✅ *Contextual styles*

✅ *Trivial shared tokens*

✅ ***Safe refactorings***

Explicitness

Easier to understand

Type-safety

Could be applied to styles, also

CSS-in-JS

is only a tool

CSS-in-JS

helps us scale CSS

How do you scale CSS?

THANK YOU

andreipfeiffer.dev