CSS Preprocessors

- CSS is not compiled
- CSS is understood by the browser
- CSS changes slowly
    - browsers have to support
- CSS can be repetitive, unstructured

What if we changed this, a little bit?

Preprocessing

- Write CSS in a non-CSS language
- Compile ("build") into CSS

Easiest if the language is LIKE CSS

- but still won't work in browser
- needs to be translated to CSS ("built")

Options that come from pre-processing CSS

- Write Multiple files
  - Output fewer files (one?)
  - Can import same file into multiple files
  - Allows organization by selector or purpose
- Variable substitution
  - less klunky than CSS variables
- Group common starting parts of selectors
- Hidden comments

Downsides of CSS Pre-processors

- They have to build
    - more complicated setup
    - slower to have changes viewable
- More effort for other contributors
    - tools required for other contributors
    - They have to understand the syntax
- Code cannot be used in pure-CSS projects

Popular CSS Preprocessors

SASS the most popular CSS preprocessor for a while

https://sass-lang.com/

SASS History

- Actually had two variants
    - `.sass` - python-esque whitespace
    - `.scss` - CSS-superset
        - means it matches CSS + more
        - unchanged CSS is valid `.scss`
        - `.scss` is probably not valid CSS

`.scss` is common

- What most people mean by "SASS"

SASS Variables

https://sass-lang.com/guide

SASS Variables use `$`

```
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

```
body {
  font: 100% Helvetica, sans-serif;
  color: #333;
}
```

Nesting

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }

  li { display: inline-block; }
}
```

```
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
nav li {
  display: inline-block;
}
```

Parent

SASS can fill-in the nesting part using `&`:

```
.alert {
  &:hover {
    font-weight: bold;
  }

  :not(&) {
    opacity: 0.8;
  }
}
```

```
.alert:hover {
  font-weight: bold;
}
:not(.alert) {
  opacity: 0.8;
}
```

Notice: Can't say `span&`

BEM Suffixes via Nested w/Parent

```scss
.accordion {
  background: #f4f4f4;

  &__copy {
    display: none;

    &--open {
      display: block;
    }
  }
}
```

```css
.accordion {
  background: #f4f4f4;
}
.accordion__copy {
  display: none;
}
.accordion__copy--open {
  display: block;
}
```

Partials and Modules

- A `.scss` file that starts with underscore (`_`)
    - Does not generate a matching `.css` file
    - Can be included in another `.scss` file with `@use`

Partials and Modules example

```scss
// _base.scss
$primary-color: #333;

body {
  color: $primary-color;
}

// styles.scss
@use 'base';

.inverse {
  background-color: base.$primary-color;
}
```

```css
/* styles.css */
body {
  color: #333;
}

.inverse {
  background-color: #333;
}
```

Comments

CSS comments are `/* */`

- these remain
- and are "loud" (in resulting CSS)
- silent if in "compressed mode"

SASS adds JS-style `//` comments

- until end of line
- "silent" (not in resulting CSS)

SCSS Comments Example

```
/* loud comment */
// silent comment
body {
  color: #BADA55;
}
```

```
/* loud comment */
body {
  color: #BADA55;
}
```

Extends

- define a CSS rule with a preceding `%`

```
%message-shared {
  color: #333;
}
```

- Have any selectors `@extend` that:

```
.message {
  @extends %message-shared;
}
.error {
  @extends %message-shared;
  border-color: red;
}
```

```
.message, .error {
  color: #333;
}
.error {
  border-color: red;
}
```

Mixin - Like extend, but can take parameters

```
@mixin theme($theme: DarkGray) {
  background: $theme;
  box-shadow 0 0 1px rgba($theme, 0.25);
}

.info {
  @include theme;
}
.alert {
  @include theme($theme: DarkRed);
}
```

```
.info {
  background: DarkGray;
  box-shadow: 0 0 1px rgba(169, 169, 169, 0.25);
}

.alert {
  background: DarkRed;
  box-shadow: 0 0 1px rgba(139, 0, 0, 0.25);
}
```

SASS on the command line

- An involved process
    - part of why we didn't cover this earlier
- Depends on OS
- Multiple implementations exist
    - different languages
    - depends on what is involved in running that language on your system
    - "primary" was ruby, then JS, now Dart

https://sass-lang.com/install

Installation outside the scope of this course

I don't recommend installing a global solution

- Instead work on a per-package basis
- Different projects may use different SASS solutions

`node-sass` no longer primary

- but still up-to-date
- works easily in JS-based projects
- is "noisy" as it has pre-built binaries

Generic statements about command-line use

A common setup is to have

- a "source" or "input" directory with the `.scss` files
- a "target" or "output" directory where the converted `.css` files go

Remember that a `.scss` file might import (@use) partials

- partials don't generate separate `.css` files

SASS has issues in systems like Vite

You CAN use SASS in Vite (and alternatives)

- But the tooling can't easily "talk" to the rest
- An alternate processor called PostCSS is common

PostCSS can automate CSS translation

- Adds "vendor prefixes"
- Converts "newer" CSS to work for older browsers
- Tries to do so without creating a new syntax
    - Unlike SASS

Is SASS winding down?

- New CSS Nesting has most used feature of SASS
- Works with PostCSS and other framework css libs

CSS Nesting DOES NOT support BEM-friendly appends:

```
.accordion {
    background: #f4f4f4;

    &__text {  /* Does NOT work in native CSS */
        display: none;
    }

    &.open { /* DOES work in native CSS (new) */
        display: block;
    }
}
```

SASS still widely used and recognized

- 72% of respondents "regularly used"
    - in 2023 StateOfCSS survey
    - Was 87.5% in 2022, 90% in 2021
- CSS Nesting feature is a result of SASS
    - Like how `.querySelector()` result of jQuery

Summary - CSS Preprocessors

- CSS Preprocessors are programs
    - turn CSS-like syntax into CSS
    - Alternate syntax easier to organize/manage
    - Separate programs mean extra build effort
- SASS is most common CSS preprocessor

Remember: Not allowed on final projects!

Summary - SCSS Syntax for SASS

- Separate from `.sass` syntax for SASS (confusing)
- Variables start with `$`
- Nested selectors
- Can insert parent using `&`
    - Great for BEM
    - Doesn't work with tag-type selectors
- A file that starts with `_` is a partial
    - loaded with `@use 'XXX';` (no `_` here)
- Comments are `/* loud */` or `// silent`
- Rules that start with `%` can (must) be extended
- `@mixin` can be passed params when `@include`ed

Summary - Native support for CSS Nesting

- Browsers quickly adding this new feature
- More limited than SASS version
    - In particular, no string appends for BEM
- Support still early!
    - Will take time