Vue Styling

SFC Styling

- `.vue` files typically end with a `<style>` tag
- By default contains CSS rules

Scoped Styles

- Prevent styles from affecting elements outside the component

- Add **scoped** attribute

```
<style scoped>
   ...
</style>
```

- Adds **data-v-*hash*** attribute to all rendered elements

  - hash is same for all instances of component
    since styling is the same for all

  - attribute has no value, it's just present

- Generated CSS rules include check for this attribute

```
<style scoped>
.danger {
   color: red;
}
</style>
```

```
.danger[data-v-1a2b3c4d] {
   color: red;
}
```

Using Preprocessors

- These add features to CSS
  like variables, nested rules, mixins, and predefined functions

- Components with a small amount of styling
  often don't need these features

- Popular options include

  - Sass - https://sass-lang.com/

  - LESS - http://lesscss.org/

  - Stylus - http://stylus-lang.com/

  - PostCSS - https://postcss.org/

- Specify with `lang` attribute in `<style>`

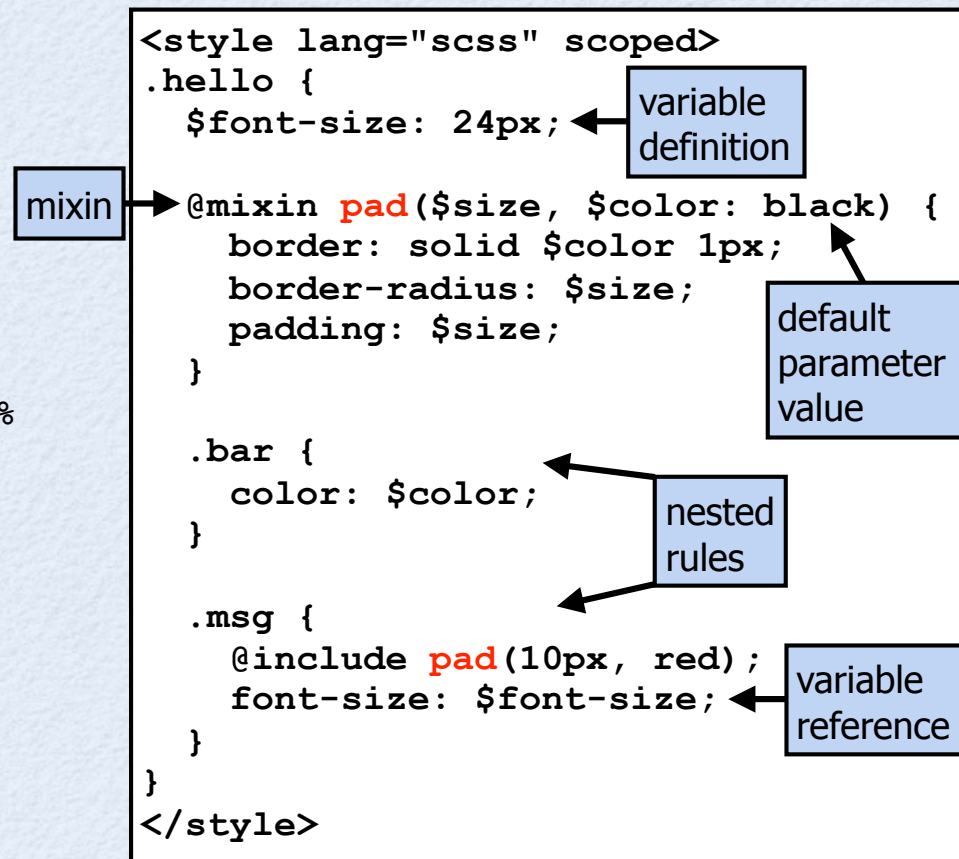 - ex. to use Sass `<style lang="scss">`

Sass

- Advantages
  - variables
  - nested rules
  - mixins
  - parent references with `&`
  - math without `calc()` using `+`, `-`, `*`, `/`, and `%`
  - provided SassScript functions
  - both `/* multi-line */` and `// single-line` comments

- Must install extra development dependencies to use Sass
  - `npm install -D \`
    `  node-sass sass-loader style-loader`

```scss
<style lang="scss" scoped>
.hello {
  $font-size: 24px;          ← variable definition

  @mixin pad($size, $color: black) {      ← mixin / default parameter value
    border: solid $color 1px;
    border-radius: $size;
    padding: $size;
  }

  .bar {
    color: $color;           ← nested rules
  }

  .msg {
    @include pad(10px, red);
    font-size: $font-size;   ← variable reference
  }
}
</style>
```

5

Recommended Sass Usage

- Add `class` attribute to top-most template element whose name matches component name

- Wrap all component CSS in a rule that matches this class name

```
<template>
  <div class="Foo">
    ... content goes here ...
  </div>
</template>

<script>
  export default {
    ... instance definition goes here ...
  };
</script>

<style lang="scss">
  .Foo {
    ... CSS rules for this component go here ...
  }
</style>
```

removes need for `scoped`, resulting in more readable CSS in devtools