




CSS PreProcessors



By: Islam ahmed hassan

Mid-Senior : Software engineer / frontend engineer



Content

- What is css PreProcessors or sass ?
- Benefits
- Installation
- Variables
- Nesting
- Mixins
- Extend / inheritance
- Placeholder
- Advanced tricky

What is Pre-processor ?

SASS (Syntactically Awesome Stylesheet) is a pre-processing language which provides it is own syntax for CSS.

It provides some features and syntax to make stylesheets writing is more efficient.

When you build a stylesheet based on these features you will get a reusable codes of css not a big stylesheet for each project.

Benefits

You can use it cause it easy to learn won't waste your time in something useless. (looks like CSS with few changes no more)

You can create reusable styles can be move from project to another without conflicts

You will get a small size file easy to maintain and your code will be scalable and open for any changes easily

Installation

- Install node .js
<https://nodejs.org/en/download/>
- Run npm --version and node --version
- Create folder for your project and run npm install
- Npm install node-sass
 - Inside package.json file add this line in script section
 - "scss": "node-sass --watch scss -o css"
- Npm run scss

Variables

Think of variables as a way to store information that you want to reuse throughout your stylesheet. You can store things like colors, font stacks, or any css value you think you'll want to reuse.

Sass uses the `$` symbol to make something a variable. Here's an example :

```
$primary-color: #333;  
  
body {  
  color: $primary-color;  
}
```

Nesting

When writing HTML you've probably noticed that it has a clear nested and visual hierarchy. CSS, on the other hand, doesn't.

Sass will let you nest your CSS selectors in a way that follows the same visual hierarchy of your HTML. Be aware that overly nested rules will result in over-qualified CSS that could prove hard to maintain and is generally considered bad practice.

With that in mind, here's an example of some typical styles for a site's navigation:

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

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

Mixins

Some things in css are a bit tedious to write, especially with css3 and the many vendor prefixes that exist.

A mixin lets you make groups of css declarations that you want to reuse throughout your site.

You can even pass in values to make your mixin more flexible. A good use of a mixin is for vendor prefixes. Here's an example for `theme`.

Mixins . cont

Scss

```
@mixin theme($theme: DarkGray) {  
  background: $theme;  
  box-shadow: 0 0 1px rgba($theme, .25);  
  color: #fff;  
}
```

```
.info {  
  @include theme;  
}
```

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

```
.success {  
  @include theme($theme: DarkGreen);  
}
```

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

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

```
.success {  
  background: DarkGreen;  
  box-shadow: 0 0 1px rgba(0, 100, 0,  
0.25);  
  color: #fff;  
}
```

Extend

This is one of the most useful features of Sass.

Using `@extend` lets you share a set of css properties from one selector to another. It helps keep your Sass very DRY

```
.message-shared {  
  border: 1px solid #ccc;  
  padding: 10px;  
  color: #333;  
}  
  
.message {  
  @extend .message-shared;  
}  
  
.success {  
  @extend .message-shared;  
  border-color: green;  
}
```

```
.error {  
  @extend .message-shared;  
  border-color: red;  
}  
  
.warning {  
  @extend .message-shared;  
  border-color: yellow;  
}
```

Placeholder

Sass has a special kind of selector known as a “placeholder”. It looks and acts a lot like a class selector, but it starts with `%` and it's not included in the CSS output. In fact, any complex selector (the ones between the commas) that even *contains* a placeholder selector isn't included in the CSS, nor is any style rule whose selectors all contain placeholders,

```
%message-shared {  
  border: 1px solid #ccc;  
  padding: 10px;  
  color: #333;  
}  
  
.message {  
  @extend %message-shared;  
}  
  
.success {  
  @extend %message-shared;  
  border-color: green;  
}
```

```
.error {  
  @extend %message-shared;  
  border-color: red;  
}  
  
.warning {  
  @extend %message-shared;  
  border-color: yellow;  
}
```

@Each

The `@each` rule makes it easy to emit styles or evaluate code for each element of a list or each pair in a map. It's great for repetitive styles that only have a few variations between them. It's usually written `@each <variable> in <expression> { ... }`, where the expression returns a list. The block is evaluated for each element of the list in turn, which is assigned to the given variable name.

```
$sizes: 40px, 50px,  
80px;
```

```
@each $size in $sizes {  
  .icon-#{ $size } {  
    font-size: $size;  
    height: $size;  
    width: $size;  
  }  
}
```

@Each .cont

Css O / P =>

```
.icon-40px {  
  font-size: 40px;  
  height: 40px;  
  width: 40px;  
}
```

```
.icon-50px {  
  font-size: 50px;  
  height: 50px;  
  width: 50px;  
}
```

```
.icon-80px {  
  font-size: 80px;  
  height: 80px;  
  width: 80px;  
}
```