**CSS 预编译语言概述**

CSS 作为一门样式语言，语法简单，易于上手，但是由于不具备常规编程语言提供的变量、函数、继承等机制，因此很容易写出大量没有逻辑、难以复用和扩展的代码，在日常开发使用中，如果没有完善的编码规范，编写的 CSS 代码会非常冗余且难以维护。

为了解决上述问题，出现了很多 CSS 预编译语言，CSS 预编译语言是基于 CSS 语言的语法扩展，除了能解决上述缺乏语言特性带来的问题之外，还支持嵌套书写，减少重复输入父级选择器（可理解为 CSS 中的继承机制），提高了代码的可读性和编写效率，常见的预编译语言有 [Less](http://lesscss.org/)、[Sass](http://sasscss.org/) 等。

Sass 与 Less 等其它预编译语言不存在优劣之分，一般来说，Sass 的功能更加强大或者说 Sass 的语言层面更接近于一门完整的编程语言，而 Less 则更接近于 CSS 语法，所以我们在 Vue 中通常会使用 Sass 编写样式代码（参考我们在 API 项目教程中的[配置 JavaScript 和 Sass](https://laravelacademy.org/post/9503.html) 这篇教程），Laravel 默认提供的样式代码也是 scss 结尾的，也是使用的 Sass 语言。所以我们来了解下 Sass 的安装和使用，非常简单，即学即用。

**Sass 简介和安装**

Sass 是对 CSS 的扩展，让 CSS 语言更强大、优雅。它允许你使用变量、嵌套规则、混合、导入等众多功能，并且完全兼容 CSS 语法。

Sass 具有两种不同的后缀名分别对应两套语法，最早 Sass 使用的是缩进式语法，使用缩进来区分代码块，并通过分号将具体样式分开，这种语法以 .sass 作为后缀；另一种使用了和 CSS 一样的块语法，这种语法以 .scss 作为后缀。后者更加兼容原生 CSS 语法，所以我们通常使用后者，接下来的教程我们也使用这种语法。

编写好 Sass 文件后，需要将其编译为 CSS 文件才能在项目中使用，为此我们需要安装相应的编译工具，Sass 官方解释器通过 Ruby 编写，同时也有其他语言实现的版本，最常见的就是 C 语言实现的 libSass，NPM 扩展包 node-sass 就封装了对 libSass 的实现，所以我们可以通过安装这个前端扩展包来编译 Sass 文件，不过在 Laravel 项目中，开箱提供了 Laravel Mix 进行前端资源的编译，当我们通过 npm install 安装 laravel-mix 的过程中，系统会自动安装 laravel-mix 声明的依赖，其中就包括了编译 Sass 所需要的 node-sass，我们无需再额外安装这个扩展包，这些事情 Laravel Mix 在底层默默帮我们完成了。

**Sass 使用语法**

Sass 提供了变量、嵌套、混合、导入、循环等功能，不过作为有其他编程语言功底的我们来说，学习起来非常简单，花个一个小时就熟悉了，下面我们逐一来介绍这些功能。

**变量**

和 PHP 一样，Sass 的变量通过 $ 作为标识符，Sass 支持的数据结构包括数字、字符串、数组、颜色、布尔值、null、List、Map、函数引用（如果你不了解 Python 或 Java 这类编程语言，也不熟悉 Redis 中的数据结构，可以将 List 理解为 PHP 中未指定键名的索引数组，将 Map 理解为以字符串作为键名的关联数组）：

// 简单变量

$primary-color: #333;

// 引用变量

body {

color: $primary-color;

}

// List

$font-stack: ('Helvetica', 'Arial', sans-serif);

body {

font: 100% nth($font-stack, 1); // 获取 List 的值，索引从1开始，不是0！

}

// Map

$breakpoints: (

small: 767px,

medium: 992px,

large: 1200px

);

// 变量作为插入变量需要通过 #{} 引入，通过 map\_get 函数从 Map 中获取值

@media (min-width: #{map-get($breakpoints, small)}) {

}

$name: foo;

$attr: border;

p.#{$name} { #{$attr}-color: #44b336; }

有两个需要注意的地方，和一般编程语言数组或列表索引从 0 开始不同，Sass 中的 List 索引从 1 开始；另外，变量作为插入变量，即作为选择器或属性名的时候需要用 #{} 引入，PHP也有类似概念，只不过是通过 {} 引入的。

**嵌套**

Sass 的嵌套语法也很实用，在此之前，我们只能通过多个 CSS 样式定义来解决嵌套问题：

nav {

ul {

margin: 0;

padding: 0;

list-style: none;

}

li { display: inline-block; }

a {

display: block;

padding: 6px 12px;

text-decoration: none;

}

}

相信我不用过多解释，你也能理解这段代码的含义，以及在样式代码编写过程中，带来的效率提升。

**混合（Mixin）**

有的时候，我们可能有一段 CSS 样式代码需要在多个地方使用，这可以通过 Sass 提供的混合（Mixin）功能来实现，定义混合代码的时候需要在选择器前面加上 @mixin 标识，引用混合代码的时候需要通过 @include 来引入：

// 清除浮动

@mixin clearfix {

zoom: 1;

&:before,

&:after { content: ""; display: table; }

&:after {

clear: both;

visibility: hidden;

font-size: 0;

height: 0;

}

}

div { @include clearfix; } // 使用 mixin

**函数**

函数和混合有点类似，不过函数更加强大，可以传入参数并实现运算功能，函数通过 @function 标识声明，函数名允许出现短划线 -，函数体内可以使用在函数声明之前定义的所有变量，同时计算时会带上变量声明时的单位：

$grid-width: 40px;

$gutter-width: 10px;

@function grid-width($n) {

@return $n \* $grid-width + ($n - 1) \* $gutter-width;

}

使用函数时直接调用函数即可：

#sidebar { width: grid-width(5); }

**控制结构**

与 Blade 模板类似，Sass 为控制结构提供了各种指令，如 @if、@else、@for、@each、@while 等：

$type:sass;

// 条件判断，根据不同条件定义不同的样式

p {

@if $type == less {

color: green;

} @else if $type == sass {

color: red;

} @else {

color: black;

}

}

// 循环，定义多个样式

@for $i from 1 through 3 {

.item-#{$i} { width: 2em \* $i; }

}

// 遍历，类似 foreach，也是定义多个样式，用于遍历 Map 结构

@each $header, $size in (h1: 2em, h2: 1.5em, h3: 1.2em) {

#{$header} {

font-size: $size

}

}

需要注意的是 @for $i from start through end 还可以改写成 @for $i from start to end，不同之处在于前者包括 end，后者不包括，另外如果要实现逆序的话，可以将 start 和 end 的值倒过来。

**导入**

Sass 支持通过 @import 指令导入其它 Sass 文件，既可以导入本地开发文件，也可以导入前端依赖库中的文件，还可以导入网络字体文件，以 Laravel 自带的 resources/sass/app.scss 为例：

// Fonts

@import url('https://fonts.googleapis.com/css?family=Nunito');

// Variables

@import 'variables';

// Bootstrap

@import '~bootstrap/scss/bootstrap';

在导入 Sass 文件的时候，可以省略文件后缀。

**继承**

Sass 还支持样式继承，我们通过 % 前缀指定用于继承的样式，然后在需要继承的地方提供 @extend 指令继承相应的父类样式：

// 以%开头的父类不会渲染

%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;

}

**操作符**

和其他常规编程语言一样，Sass 支持对变量进行+、-、\*、/、%操作。

**结语**

好了，通过以上语法的介绍相信你已经具备了编写 Sass 样式文件的能力，在基于 Laravel + Vue.js 驱动的项目中，我们通常会在两个地方编写样式代码，一个是 resources/sass  
目录下独立的 .scss 文件，另一个是 Vue 组件中的 <style lang="scss"></style> 中，我们在属性中设置 lang="scss" 表示这里面是 Sass 代码，需要 Laravel Mix 编译的时候将其编译到指定的 CSS 文件中。

声明：本教程只是 Sass 快速入门教程，不代表 Sass 的全部功能特性，更多语法请参考[官方文档](http://sasscss.org/)。

https://laravelacademy.org/post/9671.html