# SASS基础教程——SASS基本语法与特性

作者：大漠 日期：2013-07-29 点击：2998

[preprocessor](http://www.w3cplus.com/blog/tags/309.html" \t "http://www.w3cplus.com/preprocessor/_blank)

[sass](http://www.w3cplus.com/blog/tags/302.html" \t "http://www.w3cplus.com/preprocessor/_blank)

[scss](http://www.w3cplus.com/blog/tags/368.html" \t "http://www.w3cplus.com/preprocessor/_blank)

[](http://www.miaov.com/2013/download/video_download.html)

前一段时间，一直在聊SASS的环境、安装、调试以及转译等相关问题。但一直未真正的切入SASS是如何使用的。我在想，更多的同学其关注点还是如何使用SASS？如何在项目中运用SASS？那么从这篇文章开始，我们一起来走进SASS。

当你想真正走入SASS的时候，个人建议您能按照前面几篇文章，在你的电脑中构建好SASS的环境，包括如何安装环境、安装SASS、调试SASS以及编译SASS。或许正因为前期有这么多事情要做，很多同学不敢轻意的踏入，其实没有大家想得那么复杂，不管是在Window下还是在Mac OS X下，这些都是非常简单的。如果你感兴趣，不仿在正式阅读下面的内容之前，先根据下面几篇文章完成一些很有必要的工作：

* 《[SASS编译](http://www.w3cplus.com/preprocessor/sass-compile.html" \t "http://www.w3cplus.com/preprocessor/_blank)》
* 《[Nodejs+Grunt配置SASS项目自动编译](http://www.w3cplus.com/preprocessor/nodejs-and-grunt-compile-sass-to-css.html" \t "http://www.w3cplus.com/preprocessor/_blank)》
* 《[SASS调试](http://www.w3cplus.com/preprocessor/sass-debug-with-developer-tool.html" \t "http://www.w3cplus.com/preprocessor/_blank)》
* 《[SASS界面编译工具——Codekit的使用](http://www.w3cplus.com/preprocessor/sass-gui-tool-codekit.html" \t "http://www.w3cplus.com/preprocessor/_blank)》
* 《[SASS界面编译工具——Koala的使用](http://www.w3cplus.com/preprocessor/sass-gui-tool-koala.html" \t "http://www.w3cplus.com/preprocessor/_blank)》

## **简介**

做为一名页面仔来说，对于CSS并不会感到陌生。每天的工作都在为他打交道，或者说很多时候都在做一些复制粘贴性的工作。比如说，文字的排版，颜色的设置，边框设置，元素宽度调整，布局与网格制作等等。这些工作也让很多CSSER变得对自己从事的工作变得机械化，变得无趣，也失去了当初的激情。不知道大家是否有想过，通过另外一种方式来能完成我们的工作，让我们不在是一个复制代码的码农，让我们的工作不在机械，让我们的工作不在无趣。

在这种环境之下，我想使用SASS来改变我的以前的工作方式，让SASS帮我完成以前依靠复制和粘贴的工作，也让SASS来帮我完成一些更具有意义的工作。

## **SASS优点**

要了SASS，我们先从他的优点说起。SASS是一种基于CSS的预处理语言，在CSS的基础上将代码抽象各简单化。SASS让你写CSS时提供更多的自由，它像一个篇程语言一样，可以给你的样式定义变量，嵌套等功能。更强大的是可以让你的CSS具有逻辑功能。类似于其他程序语言中的条件判断、循环等。

举个栗子，或许大家更能理解。SASS可以将Hex,rgb,rgba，hsl和hsla颜色设置为变量，并在整个项目中重复使用他们。复杂一点的，你可以通过SASS的mixins生成一个网格的布局功能，然后在对应的类名通过include来调用，生成所需要的网格布局。另外还可以通过extend来调用你的前面生成的类名。要知道，所有这些都是与函数编程语言很类似。主要目的就是****让你写CSS时不在重复****。

综合上面所述，SASS优点可以简单的规纳为以下几点：

* 增加了一些编程特性
* 简少CSS代码的重复性与代码的冗余
* 维护方便
* 适应性更强，可读性更强

## **SASS语法**

SASS到目前为止有两种语法规则。最新的语法称为“SCSS”，增强了对CSS3语法的支持。这意味着每个CSS3的语法也适用于SCSS。SCSS文件有自己的扩展名.scss。

第二种语为称为SASS的老语法，也称为[缩进语法](http://sass-lang.com/docs/yardoc/file.INDENTED_SYNTAX.html" \t "http://www.w3cplus.com/preprocessor/_blank)。受[HAML](http://ru.wikipedia.org/wiki/Haml" \t "http://www.w3cplus.com/preprocessor/_blank)语法的影响，这个语法主要用来给那些追求CSS简洁的人使用。他不是使用大括号和分号，而是使用缩进来指定块。文件使用扩展名.sass。

简单的理解，SASS分为两种语法，一种是****SCSS****，另一种就是****SASS****。到目前为止，我们所说的SASS其实就是其新语法，也就是——****SCSS****，只不过大家都还是习惯性的称之为SASS。这一点大家需要注意。

这个时候，你肯定又会觉得很蛋疼，不就是一门语言吗？还搞出这么多个花样，还分什么.sass(SASS)和.scss(SCSS)。难道就不能只有一种吗？其实是可以的，如果你不想过多的去了解，那么你只需要记住SASS的新语法，也就是——****SCSS****。特别是对于新手，你只要知道SCSS的语法规则就OK了。

但话说回来，如果你想了解SCSS的来龙去脉，你就一定会知道SASS，也就会碰到上面所说的，两种语法。那么在这些，我们顺便来看看SCSS和SASS语法规则上有何不同。

### SCSS语法

SCSS语法规则和CSS的语法规则可以说是完全一样，由选择器、属性和属性三部分组成，并且和大括号{}与分号;配合在一起使用。

选择器{

属性：属性值；

}

例如有一段CSS代码如下：

.container {

width: 960px;

margin: 0 auto;}

那么在SCSS中，我们使用如下：

.container {

width: 960px;

margin: 0 auto;}

可以说SCSS和CSS的语法规则是一样的，这也是众多前端人员习喜欢使用SCSS的原因，因为他太像CSS了。和平时使用CSS的习惯基本上一致，无须为了使用SCSS而改变以前的书写代码习惯。

### SASS语法

SASS语法也称之为SASS的缩进语法，其目的是担供一个更简洁的语法。对于一些人来说，更多的是基于于CSS的美学吸引力，用SASS来代替SCSS语法。

SASS语法和CSS语法不一样，他不是使用大括号{}和分号;来分隔块的样式，它使用的是类似于[HAML](http://haml.info/" \t "http://www.w3cplus.com/preprocessor/_blank)语法，使用缩进和换行来分块，而不是使用分号来分隔语句。这通常会省去大量的篇幅，也缩小了文件大小。

每个在SASS中语句，属性声明和选择器必须放在自己的线上。换句话说，选择器和样式的声明不用大括号{}区分，但必须分行书写。

选择器

属性：属性值

同样的，我们有一段CSS代码：

.container {

width: 960px;

margin: 0 auto;}

其对应的SASS代码如下：

.container

width: 960px

margin: 0 auto

在SASS语法中，有多个选择器时，每个选择器必须在一个行，而且用逗号,分隔。简单点说，一个选择器占一行，而且相邻两个选择器之间使用,分开：

.user #userTab,.posts #postTab

width: 100px

height:30px

我们简单来看一个SCSS、SASS和CSS三者之间语法的对比示例：

##### SCSS

$blue:#3bbfce;

$margin:16px; .container{

border-color:$blue;

color:darken($blue,9%); } .border {

padding: $margin / 2;

margin: $margin / 2;

border-color: $blue; }

##### SASS

$blue:*#3bbfce* $margin: 16px

.container

border-color:$blue

color:darken($blue,9%)

.border

padding: $margin / 2

margin: $margin / 2

border-color: $blue

编译出来的CSS

.container {

border-color: #3bbfce;

color: #2b9eab;}.border {

padding: 8px;

margin: 8px;

border-color: #3bbfce;}

就我个人而言，我推荐使用SCSS，因为它似乎更具可读性。从外表看，SCSS和CSS几乎是一样的。

## **SASS特征**

仅SCSS的语法外表看，和CSS可以说是基本一致。但事实并不如此，SASS有几个基本特征，那是CSS没有的。也就是说SASS有几大特征是与生俱有的，这几大特征就是：

* 变量(****Variables****)：用来定义变量
* 嵌套(****Nesting****)：样式和属性的嵌套
* ****Mixins****：抽取样式定义为模块
* 选择器继承（****Selector Inheritance****）：继承选择样式

接下来，我们分别来了解SASS这四个基本特性。这几个基本特性就类似于CSS中的一些基本功能，只有掌握了这几个特性，才能更好的理解和掌握SASS。

### 变量（Variables）

对于变量来说，应该是开发人员最好朋友之一。在SASS中你也可以声明变量，并在整个样式表中使用。SASS支持任何变量（例如：颜色、数值、文本等）。然后你可以在任意地方引用变量。

在SASS中定义的变量，除了可以在样式中直接引用之外，还可以对变量进行一些基本的数学运算，而且还可以在一些[有用函数](http://sass-lang.com/docs/yardoc/Sass/Script/Functions.html" \t "http://www.w3cplus.com/preprocessor/_blank)中引用。

SASS声明变量必须用$开头，后面紧变量名和变量值，而且变量名和变量值之间需要使用分号:隔开。就像CSS属性设置一样：

$变量名:变量值;

在SASS和SCSS中定义变量方法是一样。我们来举个例子来看：

|------------SCSS--------------|----------------SASS----------------|/\*声明变量\*/ | /\*声明变明\*/$color: *#333; | $color: #333*$bgcolor:*#f36; | $bgcolor:#f36*

|/\*引用变量\*/ | /\*引用变量\*/

body { | body

color: $color; | color:$color

background-color: *#f36; | background-color:$bgcolor*

} |

编译出来的CSS如下：

body {

color: #333;

background-color:#f36;}

从上面的代码中我们可以看出， SASS中的变量是值级别的重复使用，可以将相同的值定义成变量统一管理起来。

SASS中变量的特性适用于定义主题（也就是我们常说的换肤），我们可以将背景颜色、字体颜色、边框属性等常规样式统一定义，这样不同的主题只需要定义不同的变量文件就可以。

### 嵌套(Nesting)

SASS中的嵌套有两种，一种是选择器的嵌套，另外一种是样式的嵌套。然而这两种嵌套的目的都是一样的，减少代码量，增强代码的可读性。

#### 选择器的嵌套

SASS中选择器的嵌套指的是在一个选择器中嵌套另一个选择器来实现继承。比如说，我们在CSS中多个元素有一个相同的父元素，那么写样式会变得很乏味，我们需要一遍一遍的在每个元素前写这个父元素，除百给特定的元素添加类名class或者ID。

section {

margin: 10px;}section nav {

height: 25px;}section nav a {

color: #0982c1;}section nav a:hover {

text-decoration: underline;}

如果使用SASS中的选择器嵌套特性，我们可以在父元素的大括号{}里写这些元素。同时可以使用&符号来引用元素的父选择器。我们来看看SASS中如何将上面的CSS代码转换成SASS代码：

##### SCSS

section {

margin: 10px;

nav {

height: 25px;

a {

color: *#0982c1;*

&:hover{

text-decoration: underline;

}

}

}

}

##### SASS

section

margin: 10px

nav

height: 25px

a

color: *#0982c1*

&:hover

text-decoration: underline

#### 属性嵌套

在很多时候，我们写CSS常会碰到属性的缩写，比如：font、background、border、margin、padding等，在SASS中我们可以使用属性嵌套。让我们的代码变得简单，明了。

来简单的看一个实例，他会更具有说明力，大家也会更易于明白什么是SASS中的属性嵌套。下面这样的场景，我想大家都常看到：

li {

font-style: italic;

font-family: serif;

font-weight: bold;

font-size:1.2em;}

在SASS中，我们可以使用属性嵌套写成：

|*------------SCSS--------------|--------------SASS----------------|*

li { | li

font: { | font:

style:italic; | style: italic

family:serif; | family: serif

weight:bold; | weight: bold

size:1.2em; | size:1.2em

} |

} |

### Mixins

Mixins是SASS中最强大的特性之一，简单点来说，Mixins可以将一部分样式抽出，作为单独定义的模块，被很多选择器重复使用。平时在写样式时肯定碰到过，某段CSS样式经常要用到多个元素中，这样就需要重复的写多次。在SASS中，可以为这些公用的CSS样式定义一个Mixin，然后在CSS需要使用这些样式的地方直接调用定义好的Mixin。这是一个非常有用的特性，Mixins被作一个公认的选择器，还可以在Mixins中定义变量或者默认参数。

SASS中声明Mixins需要使用@mixin，在后面紧跟Mixins的名：

@mixin Mixins名 {

/\*样式规则\*/

}

同时在Mixins中，还可以定义参数，同时可以给这个参数设置一个默认值，但参数名需要使用$符号开始，而且和参数值之间需要使用:分开：

@mixin Mixins名（$参数名：参数值）{

/\*样式规则\*/

}

来看一个简单的实例：

@mixin error($borderWidth:2px){

border: $borderWidth solid *#f00;*

color: *#f00;*

}

定义好Mixins之后，在需要的使用的选择器中调用定义好的Mixins。在调用定义好的Mixins时，在SCSS和SASS两种语法中调用方式不一样。

在SCSS中调用定义好的Mixins，需要使用@include关键词，然后在其后紧跟需要调用的Mixins。

选择器{

@include(Mixins名)；

}

在上例的基础上，我们来看如何在需要的地方引用定义好的errorMixins：

.generic-error {

@include error();/\*直接调用error Mixins\*/

}

.login-error {

@include error(3px);/\*调用error Mixins，并将$borderWidth参数重定义为3px\*/

}

上面的代码编译成CSS:

.generic-error {

border:2px solid #f00;

color: #f00;}.login-error {

border: 3px solid #f00;

color: #f00;}

除了上面一种调用定义好的Mixins之外，在老的语法中还支持另一种调用Mixins的方法。就是使用+，后面紧跟定义好的Mixins名：

选择器{

+Mixins名

}

回到上面的实例中，我们来演示一下如何调用：

.generic-error

+error()

.login-error

+error(3px)

此时编译出的CSS如下：

.generic-error {

border:2px solid #f00;

color: #f00;}.login-error {

border: 3px solid #f00;

color: #f00;}

### 选择器继承（Selector Inheritance）

在CSS中的属性继承应该很熟悉。平时在写CSS样式也常碰到多个元素应用相同的样式，我们在CSS中常把具有相同样式的选择器并列写在一起：

.error, .badError {

border: 1px #f00;

background: #fdd;}

.error.intrusion,.badError.intrusion {

font-size: 1.3em;

font-weight: bold;}

.badError {

border-width: 3px;}

但往往需给单独的元素添加另外的样式，这个时候我们就需要把其中选择器单独出来写样式，如此一来我们维护样式就相当的麻烦。为了应对这个问题，SASS可以从一个选择器继承另一个选择器下的所有样式。

在SASS的继承是把一个选择器的所有样式继承到另一个选择器上，在继承另一个选择器的样式时需要使用@extend开始，后面紧跟被继承的选择器：

选择器 {

@extend 定义的类

}

例如：

*/\*定义一个类\*/*

.block {

margin: 10px 5px;

padding:2px;

}

p {

@extend .block;*/\*继承.block选择器下所有样式\*/*

border: 1px solid #eee;

}

ul,ol {

@extend .block;*/\*继承.block选择器下所有样式\*/*

color:#333;

text-transform: uppercase;

}

上面的SCSS代码将编译出CSS：

.block,p,ul,ol {

margin: 10px 5px;

padding: 2px;}p {

border: 1px solid #eee;}ul,ol {

color: #333;

text-transform: uppercase;}

正如上面所看到的，上面的代码.block的样式将会被插入到相应的你要继承的选择器中，但需要注意的是优先级的问题。

但这种做法有时候会生成一些没必要的代码，比如说，仅有部分样式相同，但并不想独自创建一个类名，以免增生无用之代码，这个时候可以将类.换成%，在上例的基础上做以调整：

%block {

margin: 10px 5px;

padding: 2px;

}

p {

@extend %block;

border: 1px solid *#eee;*

}

ul,ol {

@extend %block;

color: *#333;*

text-transfomr:uppercase;

}

此时编译出来的CSS如下：

p, ul, ol {

margin: 10px 5px;

padding: 2px; }

p {

border: 1px solid #eee; }

ul, ol {

color: #333;

text-transfomr: uppercase; }

相比这下，使用%代替.定义公用样式，再通过@extend调用，编译出来的CSS要干净很多。

## **结论**

在此文中，主要介绍了SASS的两种语法的规则以及与CSS语法规则的对比。同时分别介绍了SASS中的四个基本特性：****变量****、****嵌套****、****Mixins****和****继承****的基本使用规则。

通过了解SASS的一基本语法和其特性，我们就可以继续深入SASS的下一层次学习。最后希望这篇文章能帮助大家更好的了解SASS的一些基础知识，并且在此基础上能进一步的往下深入。