2015-09-21

什么是 CSS 预处理器？

CSS 预处理器定义了一种新的语言，其基本思想是，用一种专门的编程语言，为 CSS 增加了一些编程的特性，将 CSS 作为目标生成文件，然后开发者就只要使用这种语言进行编码工作。可以让你的 CSS 更加简洁、适应性更强、可读性更佳，更易于代码的维护等诸多好处。

Sass 和 SCSS 有什么区别？

1、 文件扩展名不同，Sass 是以“.sass”后缀为扩展名，而 SCSS 是以“.scss”后缀为扩展名

2、 语法书写方式不同，Sass 是以严格的缩进式语法规则来书写，不带大括号({})和分号(;)，而 SCSS 的语法书写和我们的 CSS 语法书写方式非常类似。

由于 Sass 是基于 Ruby 写出来，所以其延续了 Ruby 的书写规范。在书写 Sass 时不带有大括号和分号，其主要是依靠严格的缩进方式来控制的。

SCSS 是 Sass 的新语法格式，从外形上来判断他和 CSS 长得几乎是一模一样，代码都包裹在一对大括号里，并且末尾结束处都有一个分号。其文件名格式常常以“.scss”为扩展名。

因为 Sass 开发之后，要让 Web 页面能调用 Sass 写好的东西，就得有这么一个过程，这个过程就称之为 **Sass 编译过程**

Sass 的编译有多种方法：**命令编译、GUI工具编译、自动化编译**

**命令编译**

是指使用你电脑中的命令终端，通过输入 Sass 指令来编译 Sass。这种编译方式是最直接也是最简单的一种方式。因为只需要在你的命令终端输入：

sass <要编译的Sass文件路径>/style.scss:<要输出CSS文件路径>/style.css

这是对一个单文件进行编译，如果想对整个项目所有 Sass 文件编译成 CSS 文件，可以这样操作

sass sass/:css/

上面的命令表示将项目中“sass”文件夹中所有“.scss”(“.sass”)文件编译成“.css”文件，并且将这些 CSS 文件都放在项目中“css”文件夹中。

实际编译过程中，你会发现上面的命令，只能一次性编译。每次个性保存“.scss”文件之后，都得重新执行一次这样的命令。如此操作太麻烦，其实还有一种方法，就是在编译 Sass 时，开启“**watch**”功能，这样只要你的代码进行任保修改，都能自动监测到代码的变化，并且给你直接编译出来：

sass --watch <要编译的Sass文件路径>/style.scss:<要输出CSS文件路径>/style.css

对于 **GUI 界面编译工具**，目前较为流行的主要有：

1. Koala (<http://koala-app.com/>)
2. Compass.app（<http://compass.kkbox.com/>）
3. Scout（<http://mhs.github.io/scout-app/>）
4. CodeKit（<https://incident57.com/codekit/index.html>）
5. Prepros（<https://prepros.io/>）

在Sass的编译的过程中，是不是支持“GBK”编码的。所以在创建 Sass 文件时，就需要将文件编码设置为“utf-8”。另外一个错误就是路径中的中文字符引起的。建议在项目中文件命名或者文件目录命名不要使用中文字符。

在 Sass 中编译出来的样式风格也可以按不同的样式风格显示。其主要包括以下几种样式风格：

1. **嵌套输出方式 nested**
2. **展开输出方式 expanded**
3. **紧凑输出方式 compact**
4. **压缩输出方式 compressed**

1、嵌套输出方式 nested

sass --watch test.scss:test.css --style nested

2、展开输出方式 expanded

sass --watch test.scss:test.css --style expanded

这个输出的 CSS 样式风格和 nested 类似，只是大括号在另起一行

3、紧凑输出方式 compact

sass --watch test.scss:test.css --style compact

该方式适合那些喜欢单行 CSS 样式格式的朋友

4、压缩输出方式 compressed

sass --watch test.scss:test.css --style compressed

压缩输出方式会去掉标准的 Sass 和 CSS 注释及空格。也就是压缩好的 CSS 代码样式风格：

**变量**

使用 Sass 的同学都希望能在浏览器中直接调试 Sass 文件，能找到对应的行数。值得庆幸的是，现在实现并不是一件难事，只要你的浏览器支持“sourcemap”功能即可。早一点的版本，需要在编译的时候添加“--sourcemap”  参数

sass --watch --scss --sourcemap style.scss:style.css

普通变量定义之后可以在全局范围内使用。sass 的**默认变量**仅需要在值后面加上 !default 即可。

sass 的默认变量一般是用来设置默认值，然后根据需求来覆盖的，覆盖的方式也很简单，只需要在默认变量之前重新声明下变量即可。

在元素内部定义的变量不会影响其他元素。如此可以简单的理解成，**全局变量**就是定义在元素外面的变量

**什么时候声明变量？**

我的建议，创建变量只适用于感觉确有必要的情况下。不要为了某些骇客行为而声明新变量，这丝毫没有作用。只有满足所有下述标准时方可创建新变量：

1. 该值至少重复出现了两次；
2. 该值至少可能会被更新一次；
3. 该值所有的表现都与变量有关（非巧合）。

<http://sassmeister.com/>

其实伪类嵌套和属性嵌套非常类似，只不过他需要借助`&`符号一起配合使用。

**混合宏**

@mixin border-radius{

-webkit-border-radius: 5px;

border-radius: 5px;

}

中 @mixin 是用来声明混合宏的关键词，有点类似 CSS 中的 @media、@font-face 一样。border-radius 是混合宏的名称。大括号里面是复用的样式代码。

除了声明一个不带参数的混合宏之外，还可以在定义混合宏时带有参数，如：

@mixin border-radius($radius:5px){

-webkit-border-radius: $radius;

border-radius: $radius;

}

复杂的混合宏：

上面是一个简单的定义混合宏的方法，当然， Sass 中的混合宏还提供更为复杂的，你可以在大括号里面写上带有逻辑关系，帮助更好的做你想做的事情,如：

@mixin box-shadow($shadow...) {

@if length($shadow) >= 1 {

@include prefixer(box-shadow, $shadow);

} @else{

$shadow:0 0 4px rgba(0,0,0,.3);

@include prefixer(box-shadow, $shadow);

}

}

这个 box-shadow 的混合宏，带有多个参数，这个时候可以使用“ … ”来替代。简单的解释一下，当 $shadow 的参数数量值大于或等于“ 1 ”时，表示有多个阴影值，反之调用默认的参数值“ 0 0 4px rgba(0,0,0,.3) ”。

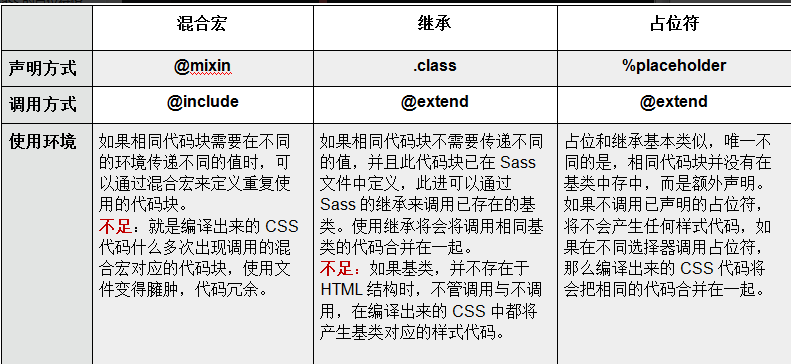
在 Sass 中通过 @mixin 关键词声明了一个混合宏，那么在实际调用中，其匹配了一个关键词“@include”来调用声明好的混合宏

在 Sass 中也具有继承一说，也是继承类中的样式代码块。在 Sass 中是通过关键词 “@extend”来继承已存在的类样式块，从而实现代码的继承。在 Sass 中的继承，可以继承类样式块中所有样式代码，而且编译出来的 CSS 会将选择器合并在一起，形成组合选择器。

Sass 中的占位符 %placeholder 功能是一个很强大，他可以取代以前 CSS 中的基类造成的代码冗余的情形。因为 %placeholder 声明的代码，如果不被 @extend 调用的话，不会产生任何代码。

如果你的代码块中涉及到变量，建议使用混合宏来创建相同的代码块。

如果你的代码块不需要专任何变量参数，而且有一个基类已在文件中存在，那么建议使用 Sass 的继承。



**注释**

在 Sass 中注释有两种方式，我暂且将其命名为：

1、类似 CSS 的注释方式，使用 ”/\* ”开头，结属使用 ”\*/ ”  
2、类似 JavaScript 的注释方式，使用“//”

两者区别，前者会在编译出来的 CSS 显示，后者在编译出来的 CSS 中不会显示

**数据类型**

 Sass 和 JavaScript 语言类似，也具有自己的数据类型，在 Sass 中包含以下几种数据类型：

* 数字: 如，1、 2、 13、 10px；
* 字符串：有引号字符串或无引号字符串，如，"foo"、 'bar'、 baz；
* 颜色：如，blue、 #04a3f9、 rgba(255,0,0,0.5)；
* 布尔型：如，true、 false；
* 空值：如，null；
* 值列表：用空格或者逗号分开，如，1.5em 1em 0 2em 、 Helvetica, Arial, sans-serif。

**字符串**

SassScript 支持 CSS 的两种字符串类型：

* 有引号字符串 (quoted strings)，如 "Lucida Grande" 、'http://sass-lang.com'；
* 无引号字符串 (unquoted strings)，如 sans-serifbold。

在编译 CSS 文件时不会改变其类型。只有一种情况例外，使用 #{ }插值语句 (interpolation) 时，有引号字符串将被编译为无引号字符串，这样方便了在混合指令 (mixin) 中引用选择器名。

所谓**值列表** (lists) 是指 Sass通过空格或者逗号分隔的一系列的值，独立的值也被视为值列表——只包含一个值的值列表。

Sass列表函数（Sass list functions）赋予了值列表更多功能（Sass进级会有讲解）：

1. nth函数（nth function） 可以直接访问值列表中的某一项；
2. join函数（join function） 可以将多个值列表连结在一起；
3. append函数（append function） 可以在值列表中添加值；
4. @each规则（@each rule） 则能够给值列表中的每个项目添加样式。

**运算**

运算时碰到不同类型的单位时，编译也会报错，

”/  ”符号被当作除法运算符时有以下几种情况：

•    如果数值或它的任意部分是存储在一个变量中或是函数的返回值。  
•    如果数值被圆括号包围。  
•    如果数值是另一个数学表达式的一部分。

如下所示：

//SCSS

p {

font: 10px/8px; // 纯 CSS，不是除法运算

$width: 1000px;

width: $width/2; // 使用了变量，是除法运算

width: round(1.5)/2; // 使用了函数，是除法运算

height: (500px/2); // 使用了圆括号，是除法运算

margin-left: 5px + 8px/2px; // 使用了加（+）号，是除法运算

}