Sass学习与分享

徐官龙

2015年8月

内容提要

- 第一部分：Sass介绍
- 第二部分：compass介绍
- 第三部分：总结

第一 部分　Sass介绍

.scss　.sass

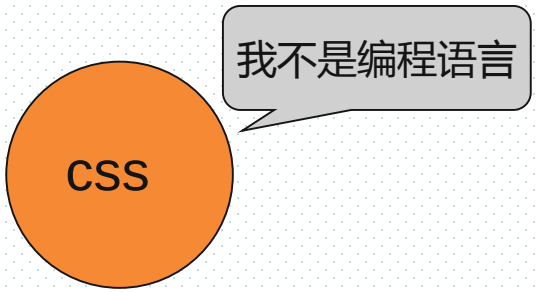
```
$blue: #3bbfce;
$margin: 16px;

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

.border {
  padding: $margin / 2;
  margin: $margin / 2;
  border-color: $blue;
}
```

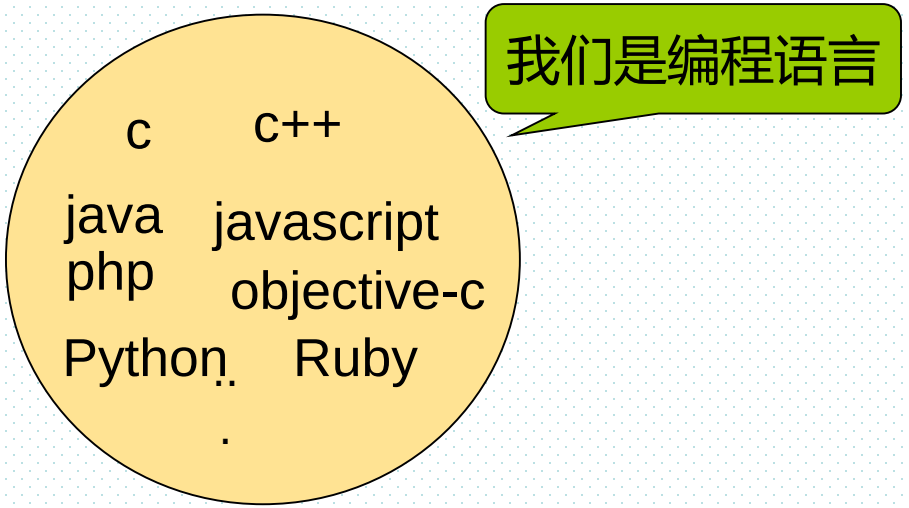
SASS介绍

我们都知道，CSS并不是一种编程语言。它并不像其它程序语言，有自己的变量、常量、逻辑判断语句这些特征，CSS只是一行行单纯的属性描述，写起来相当的费事，而且代码难易组织和维护。

很自然的，有人就开始在想，能不能让CSS像其他程序语言一样，加入编程元素，让CSS能像其他程序语言一样可以做一些预定的处理。

于是“CSS预处器（CSS Preprocessor）”这个玩意。





CSS预处理器

CSS预处理器定义了一种新的语言，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，然后再编译成正常的CSS文件。CSS预处理器为CSS增加一些编程的特性，例如可以在CSS中使用变量、逻辑判断、函数等在编程语言中的一些基本特性，可以让你的CSS更加简洁、可读性更强，更易于代码的维护等诸多好处。常见的预处理器有：Sass,LESS,STYLUS.

特性——变量($)

当样式表里需要换一套颜色，要找到好几处地方，并且替换好几次颜色值。如果我们能够在只改变某一处的属性值，其他地方的属性也会随之相应改变？那么我们可以使用Sass！

```
a {
    color: #08c;
}
.nav {
    background-color: #08c;
}
h1{
    color: #08c;
}
.foo{
    color: #08c;
}
.head{
    color: #08c;
}
```

编译后

```
$linkColor: #08c;
a {
    color: $linkColor;
}
.nav {
    background-color: $linkColor;
}
h1{
    color: $linkColor;
}
.foo{
    color: $linkColor;
}
.head{
    color: $linkColor;
}
```

特性——混合(@mixin)

当样式表里需要换一套颜色，要找到好几处地方，并且替换好几次颜色值。如果我们能够在只改变某一处的属性值，其他地方的属性也会随之相应改变？那么我们可以使用Sass！

```
a {
    color: #08c;
}
.nav {
    background-color: #08c;
}
h1{
    color: #08c;
}
.foo{
    color: #08c;
}
.head{
    color: #08c;
}
```

编译后

```
$linkColor: #08c;
a {
    color: $linkColor;
}
.nav {
    background-color: $linkColor;
}
h1{
    color: $linkColor;
}
.foo{
    color: $linkColor;
}
.head{
    color: $linkColor;
}
```

特性——混合(@mixin)

如果你的整个网站中有几处小小的样式类似（例如一致的颜色和字体），那么使用变量来统一处理这种情况是非常不错的选择。但是当你的样式变得越来越复杂，你需要大段大段的重用样式的代码，独立的变量就没办法应付这种情况了。你可以通过sass的混合器实现大段样式的重用。

```
content {
    font-size: 20px;
    color:black;
    line-height: 1.5;
    margin:20px;
}
header {
    font-size: 25px;
    color:blue;
    line-height: 1.5;
    padding: 10px;
}
footer {
    font-size: 30px;
    color:red;
    line-height: 1.5;
    padding: 15px;
}
```

编译后：

```
@mixin style($size: 15px, $color:#999) {
  font-size: $size;
  color:$color;
  line-height: 1.5;
}
content {
    @include style(20px,black);
    margin:20px;
}
header {
    @include style(25px,blue);
}
footer {
    @include style(30px,red);
}
```

特性——继承(extend)

sass中，选择器继承可以让选择器继承另一个选择器的所有样式，并联合声明。使用选择器的继承，要使用关键词`@extend`，后面紧跟需要继承的选择器

```
#header,
#footer,
.ir{
  text-shadow: none;
  background-color: blue;
  border: 0;
}
#header{
  width:300px;
}
#footer{
  width:300px;
}
```

编译后：

```
%ir{
    text-shadow: none;
    background-color: blue;
    border: 0;
}
#header{
   @extend %ir;
   height:100px;
}
#footer{
   @extend %ir;
   height:150px;
}
.ir{
     @extend %ir;
}
```

继承(extend)——细节

不要在css规则中使用后代选择器（比如.foo .bar）去继承css规则。如果你这么做，同时被继承的css规则有通过后代选择器修饰的样式，生成css中的选择器的数量很快就会失控，有可能有坑！

```
.bip .btt,
 .bip .foo .bar,
 .foo .bip .bar {
    border: 1px solid #fffff;
}
```

编译后：

```
.foo .bar{
    @extend .btt
}
.bip  .btt {
    border:1px solid #fffff;
}
```

特性——函数(@function)

Sass定义了很多函数可供使用，以@fuction开始。比如颜色函数。lighten为减淡，darken为加深。
其调用方法为：
lighten($color,$amount)，
darken($color,$amount)，
它们的第一个参数都是颜色值，第二个参数都是百分比。

```
$baseFontSize: 10px !default;
$gray: #ccc !defualt;

// pixels to rems
@function pxToRem($px) {
    @return $px / $baseFontSize * 1rem;
}

body{
    font-size:$baseFontSize;
    color:lighten($gray,10%);
}
.test{
    font-size:pxToRem(16px);
    color:darken($gray,10%);
}
```

特性——控制语句

Sass 中也有诸如 @if、@while 等常见的控制语句来实现一些简单的流程控制

有@for 循环、@while循环、@each循环

```
@each $animal in puma, sea-slug, egret {
  .#{$animal}-icon {
     background-image: url('/images/#{$animal}.png');
  }
}

//css
.puma-icon {
    background-image: url('/images/puma.png');
}
 .sea-slug-icon {
    background-image: url('/images/sea-slug.png');
 }
 .egret-icon {
      background-image: url('/images/egret.png');
 }
```

SASS使用小结

1. 更有效的使用Sass变量；
2. 不要使用没有设置参数的@mixin，他们应该是.class或者%placeholders；
3. 不要轻意（从不使用）@extend调用.class。会得到你意想不到的结果，特别是定义的.class出现在嵌套或其他的样式表中，你应该使用@extend调用%placeholders；
4. 不要使用太深的选择器嵌套；
5. 不要太相信SASS的自动编译，你应该时时检查生成的CSS。在SASS中纠错能力比较差；
6. %placeholder可以多次使用，而且不会生成重复的代码。这使得输入的CSS更友好，更干净。

第二部分 Compass介绍

什么是compass？

简单说，Compass是Sass的工具库（toolkit）。Compass在Sass的基础上，封装了一系列有用的模块和模板，丰富了Sass的功能。它们之间的关系，有点像Javascript和jQuery的关系。

什么是compass？

简单说，Compass是Sass的工具库（toolkit）。Compass在Sass的基础上，封装了一系列有用的模块和模板，丰富了Sass的功能。它们之间的关系，有点像Javascript和jQuery的关系。

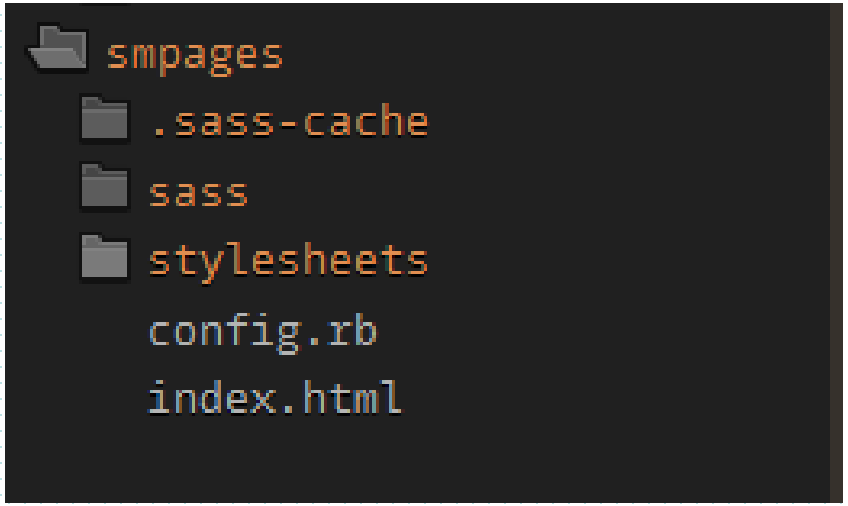
Compass是用Ruby语言开发的，所以安装它之前，必须安装Ruby。假定你的机器已经安装好Ruby，那么在命令行模式下键入：gem install compass
Compass（连同Sass）就安装好了。

创建compass项目

假定项目的名字叫myProject，在命令行键入：

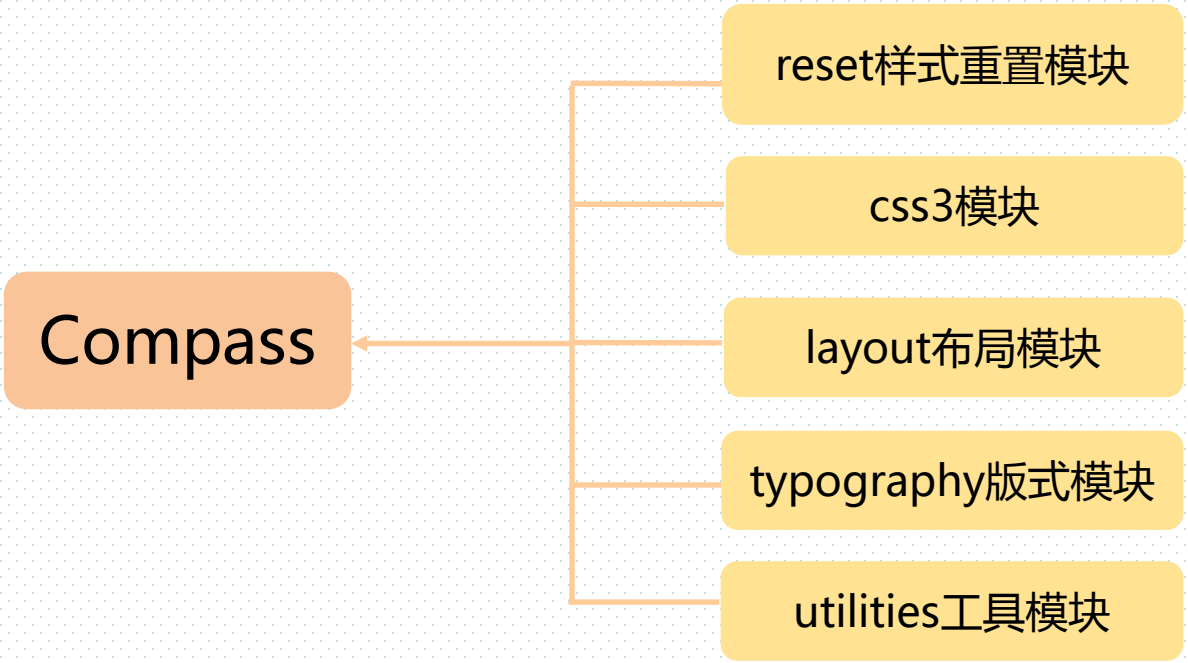
```
compass create smpages
```

当前目录中就会生成一个myproject子目录。进入该目录，里面有一个config.rb文件，这是项目的配置文件。还有两个子目录Sass和stylesheets，前者存放Sass源文件，后者存放编译后的css文件。



Compass的模块

Compass采用模块结构，不同模块提供不同的功能。目前，它内置五个模块：



compass模块的使用方法

使用@import命令，用来指定加载模块， 例如：@import "compass/css3";编译后，会生成相应的css3代码。

css3模块中的圆角（border-radius）的写法：

```
.rounded {
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
    -o-border-radius: 5px;
    -ms-border-radius: 5px;
    -khtml-border-radius: 5px;
    border-radius: 5px;
}
```

编译后：

```
@import "compass/css3";
.rounded {
    @include border-radius(5px);
}
```

第三部分　总结

如何在团队中高效使用Sass

1.设计师需要提供公用的变量值，如颜色，宽度，高度，字体大小等，以便提前定义好所需要的全局变量。

2.提前定义好公用的base，mixin，component等公用文件，根据项目进行微调，由master带领团队成员共同维护。

3.修改重用性高的样式，必须与所有团队成员商议后，有master决定是否可修改。

4.不能直接修改css文件，css文件必须由scss文件编译生成。

Thank you