sass学习教程总结分析

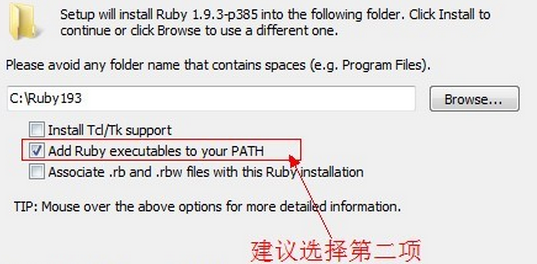
# 一、安装sass

**ruby安装**

<http://rubyinstaller.org/downloads/>

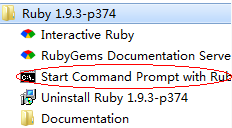
因为sass依赖于ruby环境，所以装sass之前先确认装了ruby。先导官网下载个[ruby](http://rubyinstaller.org/downloads" \t "_blank)

在安装的时候，请勾选Add Ruby executables to your PATH这个选项，添加环境变量，不然以后使用编译软件的时候会提示找不到ruby环境



**sass安装**

安装完ruby之后，在开始菜单中，找到刚才我们安装的ruby，打开Start Command Prompt with Ruby



然后直接在命令行中输入

**第一种方法：**

gem install sass

按回车键确认，等待一段时间就会提示你sass安装成功。最近因为墙的比较厉害，如果你没有安装成功，那么请参考下面的淘宝的RubyGems镜像安装sass，如果成功则忽略。

如果要安装beta版本的，可以在命令行中输入

**第二种方法：**

gem install sass –pre

**第三种方法：**

你还可以从sass的[Git repository](http://www.w3cplus.com/sassguide/github.com/nex3/sass" \t "_blank)来安装，git的命令行为

git clone git://github.com/nex3/sass.git

cd sass

rake install

升级sass版本的命令行为

gem update sass

查看sass版本的命令行为

sass -v

你也可以运行帮助命令行来查看你需要的命令

sass -h

**淘宝RubyGems镜像安装 sass**

由于国内网络原因（你懂的），导致 rubygems.org 存放在 Amazon S3 上面的资源文件间歇性连接失败。这时候我们可以通过gem sources命令来配置源，先移除默认的https://rubygems.org源，然后添加淘宝的源https://ruby.taobao.org/，然后查看下当前使用的源是哪个，如果是淘宝的，则表示可以输入sass安装命令gem install sass了，关于常用gem source命令可参看：[常用的gem source](http://foyo99.zhuidaniu.com/blogs/687" \t "_blank)

**第四种方法：**

## 淘宝RubyGems镜像安装 sass

由于国内网络原因（你懂的），导致 rubygems.org 存放在 Amazon S3 上面的资源文件间歇性连接失败。这时候我们可以通过gem sources命令来配置源，先移除默认的https://rubygems.org源，然后添加淘宝的源https://ruby.taobao.org/，然后查看下当前使用的源是哪个，如果是淘宝的，则表示可以输入sass安装命令gem install sass了，关于常用gem source命令可参看：[常用的gem source](http://foyo99.zhuidaniu.com/blogs/687" \t "_blank)

$ gem sources --remove https://rubygems.org/

$ gem sources -a https://ruby.taobao.org/ 【如果你系统不支持https，请将淘宝源更换成：gem sources -a http://gems.ruby-china.org】

$ gem sources -l

\*\*\* CURRENT SOURCES \*\*\*

https://ruby.taobao.org

# 请确保只有 ruby.taobao.org

$ gem install sass

<http://lijavawei.blog.163.com/blog/static/4565686520129227241445/> gem命令地址

# 二、sass语法

关于sass 3.3.0更新说明——[3.3.0](http://sass-lang.com/documentation/file.SASS_CHANGELOG.html#330_7_march_2014)

sublime相关插件为：[scss语法高亮](https://github.com/MarioRicalde/SCSS.tmbundle" \t "_blank)，[sass语法高亮](https://github.com/nathos/sass-textmate-bundle)，[编译](https://github.com/jaumefontal/SASS-Build-SublimeText2)，[保存即编译](https://github.com/alexnj/SublimeOnSaveBuild)，[格式化](https://github.com/badsyntax/SassBeautify)

## 文件后缀名

sass有两种后缀名文件：一种后缀名为sass，不使用大括号和分号；另一种就是我们这里使用的scss文件，这种和我们平时写的css文件格式差不多，使用大括号和分号。而本教程中所说的所有sass文件都指后缀名为scss的文件。在此也建议使用后缀名为scss的文件，以避免sass后缀名的严格格式要求报错。

//文件后缀名为sass的语法

body

background: #eee

font-size:12px

p

background: #0982c1

//文件后缀名为scss的语法

body {

background: #eee;

font-size:12px;

}

p{

background: #0982c1;

}

## 导入

sass的导入(@import)规则和CSS的有所不同，编译时会将@import的scss文件合并进来只生成一个CSS文件。但是如果你在sass文件中导入css文件如@import 'reset.css'，那效果跟普通CSS导入样式文件一样，导入的css文件不会合并到编译后的文件中，而是以@import方式存在。

所有的sass导入文件都可以忽略后缀名.scss。一般来说基础的文件命名方法以\_开头，如\_mixin.scss。这种文件在导入的时候可以不写下划线，可写成@import "mixin"。

**被导入sass文件a.scss：**

//a.scss

//-------------------------------

body {

background: #eee;

}

**需要导入样式的sass文件b.scss：**

@import "reset.css";

@import "a";

p{

background: #0982c1;

}

**转译出来的b.css样式：**

@import "reset.css";

body {

background: #eee;

}

p{

background: #0982c1;

}

根据上面的代码可以看出，b.scss编译后，reset.css继续保持import的方式，而a.scss则被整合进来了。

**使用方法如下：**

下面的命令，可以在屏幕上显示.scss文件转化的css代码。（假设文件名为test。）

　　sass test.scss

如果要将显示结果保存成文件，后面再跟一个.css文件名。

　　sass test.scss test.css

SASS提供四个[编译风格](http://sass-lang.com/docs/yardoc/file.SASS_REFERENCE.html" \l "output_style" \t "_blank)的选项：

　　\* nested：嵌套缩进的css代码，它是默认值。

　　\* expanded：没有缩进的、扩展的css代码。

　　\* compact：简洁格式的css代码。

\* compressed：压缩后的css代码。

注意：在放置文件的时候不要使用中文命名，否则文件编译出来的会有问题。(编译风格不执行)

生产环境当中，一般使用最后一个选项。

　　sass --style compressed test.sass test.css

你也可以让SASS监听某个文件或目录，一旦源文件有变动，就自动生成编译后的版本。

对于一个项目而言，不太可能只有一个CSS样式文件，如果有多个Sass文件需要监听时，就很有必要的组织一下。默认情况之下，我喜欢把所有的 Sass文件放在“/css/sass”目录中，而生成的CSS文件则直接放在“/css”目录下。现在我们修改一下项目文件目录结构，在“/css”目 录中再创建一个“sass”目录，并找刚才创建的“style.scss”文件移至“/css/sass”目录下。此时监听“css/sass”下的所有 Sass文件时，可以在命令面板中输入如下格式的命令：

$ sass --watch sassFileDirectory:cssFileDirectory（目录不要用中文，否则监听目录会报错）

在本例中，冒号前面的“sassFileDirectory”是指“/css/sass”下的“sass”目录，冒号后面的“cssFileDirectory”是指“css”目录，对应的命令就是：

$ sass --watch css/sass:css

　　// watch a file

　　sass --watch input.scss:output.css

　　// watch a directory

　　sass --watch app/sass:public/stylesheets

SASS的官方网站，提供了一个[在线转换器](http://sass-lang.com/try.html" \t "_blank)。你可以在那里，试运行下面的各种例子。

## 注释

sass有两种注释方式，一种是标准的css注释方式/\* \*/，另一种则是//双斜杆形式的单行注释，不过这种单行注释不会被转译出来。

### 标准的css注释

/\*

\*我是css的标准注释

\*设置body内距

\*/

body{

padding:5px;

}

### 双斜杆单行注释

单行注释跟JavaScript语言中的注释一样，使用又斜杠（//），但单行注释不会输入到CSS中。

//我是双斜杠表示的单行注释

//设置body内距

body{

padding:5px; //5px

}

针对注释容易报错问题：



解决方案如下：

**1.koala可视化编译工具，**

找到安装目录里面sass-3.3.7模块下面的engine.rb文件，例如下面路径：

C:\Program Files (x86)\Koala\rubygems\gems\sass-3.3.7\lib\sass

在这个文件里面engine.rb，添加一行代码

Encoding.default\_external = Encoding.find('utf-8')

放在所有的require XXXX 之后即可。

**2.命令行工具同理**

找到ruby的安装目录，里面也有sass模块，如这个路径：

C:\Ruby\lib\ruby\gems\1.9.1\gems\sass-3.3.14\lib\sass

我自己安装的路径：

D:\Program Files\Ruby200-x64\lib\ruby\gems\2.0.0\gems\sass-3.4.21\lib\sass

在这个文件里面engine.rb，添加一行代码（同方法1）

Encoding.default\_external = Encoding.find('utf-8')

放在所有的require XXXX 之后即可。

**最后在scss文件头部启用编码声明：**

|  |
| --- |
| **@charset "utf-8";//必须设置了这个才能编译有中文的注释** |

## 变量

sass的变量必须是$开头，后面紧跟变量名，而变量值和变量名之间就需要使用冒号(:)分隔开（就像CSS属性设置一样），**如果值后面加上!default则表示默认值**。

### 普通变量

定义之后可以在全局范围内使用。

//sass style

//-------------------------------

$fontSize: 12px;

body{

font-size:$fontSize;

}

//css style

//-------------------------------

body{

font-size:12px;

}

### 默认变量

sass的默认变量仅需要在值后面加上!default即可。

//sass style

//-------------------------------

$baseLineHeight: 1.5 !default;

body{

line-height: $baseLineHeight;

}

//css style

//-------------------------------

body{

line-height:1.5;

}

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

//sass style

//-------------------------------

$baseLineHeight: 2;

$baseLineHeight: 1.5 !default;

body{

line-height: $baseLineHeight;

}

//css style

//-------------------------------

body{

line-height:2;

}

可以看出现在编译后的line-height为2，而不是我们默认的1.5。默认变量的价值在进行组件化开发的时候会非常有用。

### 特殊变量

一般我们定义的变量都为属性值，可直接使用，但是如果变量作为属性或在某些特殊情况下等则必须要以#{$variables}形式使用。

//sass style

//-------------------------------

$borderDirection: top !default;

$baseFontSize: 12px !default;

$baseLineHeight: 1.5 !default;

//应用于class和属性

.border-#{$borderDirection}{

border-#{$borderDirection}:1px solid #ccc;

}

//应用于复杂的属性值

body{

font:#{$baseFontSize}/#{$baseLineHeight};

}

//css style

//-------------------------------

.border-top{

border-top:1px solid #ccc;

}

body {

font: 12px/1.5;

}

### 多值变量

多值变量分为list类型和map类型，简单来说list类型有点像js中的数组，而map类型有点像js中的对象。

#### list

list数据可通过空格，逗号或小括号分隔多个值，可用nth($var,$index)取值。关于list数据操作还有很多其他函数如length($list)，join($list1,$list2,[$separator])，append($list,$value,[$separator])等，具体可参考[sass Functions](http://sass-lang.com/documentation/Sass/Script/Functions.html)（搜索List Functions即可）

**定义**

//一维数据

$px: 5px 10px 20px 30px;

//二维数据，相当于js中的二维数组

$px: 5px 10px, 20px 30px;

$px: (5px 10px) (20px 30px);

使用

//sass style

//-------------------------------

$linkColor:#08c #333 !default;//第一个值为默认值，第二个鼠标滑过值

a{

color:nth($linkColor,1);

&:hover{

color:nth($linkColor,2);

}

}

//css style

//-------------------------------

a{

color:#08c;

}

a:hover{

color:#333;

}

#### map

map数据以key和value成对出现，其中value又可以是list。格式为：$map: (key1: value1, key2: value2, key3: value3);。可通过map-get($map,$key)取值。关于map数据还有很多其他函数如map-merge($map1,$map2)，map-keys($map)，map-values($map)等，具体可参考[sass Functions](http://sass-lang.com/documentation/Sass/Script/Functions.html)（搜索Map Functions即可）

**定义**

$heading: (h1: 2em, h2: 1.5em, h3: 1.2em);

**使用**

//sass style

//-------------------------------

$headings: (h1: 2em, h2: 1.5em, h3: 1.2em);

@each $header, $size in $headings {

#{$header} {

font-size: $size;

}

}

//css style

//-------------------------------

h1 {

font-size: 2em;

}

h2 {

font-size: 1.5em;

}

h3 {

font-size: 1.2em;

}

### 全局变量

在变量值后面加上!global即为全局变量。这个目前还用不上，不过将会在sass 3.4后的版本中正式应用。目前的sass变量范围饱受诟病，所以才有了这个全局变量。

**目前变量机制**

在选择器中声明的变量会覆盖外面全局声明的变量。(这也就人们常说的sass没有局部变量)

//sass style

//-------------------------------

$fontSize: 12px;

body{

$fontSize: 14px;

font-size:$fontSize;

}

p{

font-size:$fontSize;

}

//css style

//-------------------------------

body{

font-size:14px;

}

p{

font-size:14px;

}

**启用global之后的机制**

请注意，这个目前还无法使用，所以样式不是真实解析出来的。

//sass style

//-------------------------------

$fontSize: 12px;

$color: #333;

body{

$fontSize: 14px;

$color： #fff !global;

font-size:$fontSize;

color:$color;

}

p{

font-size:$fontSize;

color:$color;

}

//css style

//-------------------------------

body{

font-size:14px;

color:#fff;

}

p{

font-size:12px;

color:#fff;

}

这里设置了两个变量，然后在body里面重新设置了下，有点不同的是对于$color变量，我们设置了!global。通过编译后的css可以看到font-size取值不同，而color取值相同。与上面的机制对比就会发现默认在选择器里面的变量为局部变量，**而只有设置了!global之后才会成为全局变量**。

关于变量的详细分析请查阅[sass揭秘之变量](http://www.w3cplus.com/preprocessor/sass-basic-variable.html)

## 嵌套(Nesting)

sass的嵌套包括两种：一种是选择器的嵌套；另一种是属性的嵌套。我们一般说起或用到的都是选择器的嵌套。

### 选择器嵌套

所谓选择器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，从而增强了sass文件的结构性和可读性。

**在选择器嵌套中，可以使用&表示父元素选择器**

//sass style

//-------------------------------

#top\_nav{

line-height: 40px;

text-transform: capitalize;

background-color:#333;

li{

float:left;

}

a{

display: block;

padding: 0 10px;

color: #fff;

&:hover{

color:#ddd;

}

}

}

//css style

//-------------------------------

#top\_nav{

line-height: 40px;

text-transform: capitalize;

background-color:#333;

}

#top\_nav li{

float:left;

}

#top\_nav a{

display: block;

padding: 0 10px;

color: #fff;

}

#top\_nav a:hover{

color:#ddd;

}

### 属性嵌套

所谓属性嵌套指的是有些属性拥有同一个开始单词，如border-width，border-color都是以border开头。拿个官网的实例看下：

//sass style

//-------------------------------

.fakeshadow {

border: {

style: solid;

left: {

width: 4px;

color: #888;

}

right: {

width: 2px;

color: #ccc;

}

}

}

//css style

//-------------------------------

.fakeshadow {

border-style: solid;

border-left-width: 4px;

border-left-color: #888;

border-right-width: 2px;

border-right-color: #ccc;

}

当然这只是个属性嵌套的例子，如果实际这样使用，那估计得疯掉。

## @at-root

sass3.3.0中新增的功能，用来跳出选择器嵌套的。默认所有的嵌套，继承所有上级选择器，但有了这个就可以**跳出所有上级选择器**。

**普通跳出嵌套**

//sass style

//-------------------------------

//没有跳出

.parent-1 {

color:#f00;

.child {

width:100px;

}

}

//单个选择器跳出

.parent-2 {

color:#f00;

@at-root .child {

width:200px;

}

}

//多个选择器跳出

.parent-3 {

background:#f00;

@at-root {

.child1 {

width:300px;

}

.child2 {

width:400px;

}

}

}

//css style

//-------------------------------

.parent-1 {

color: #f00;

}

.parent-1 .child {

width: 100px;

}

.parent-2 {

color: #f00;

}

.child {

width: 200px;

}

.parent-3 {

background: #f00;

}

.child1 {

width: 300px;

}

.child2 {

width: 400px;

}

**@at-root (without: ...)和@at-root (with: ...)**

**默认@at-root**只会跳出选择器嵌套，而不能跳出@media或@support，如果要跳出这两种，则需使用@at-root (without: media)，@at-root (without: support)。这个语法的**关键词**有四个：all（表示所有），rule（表示常规css），media（表示media），support（表示support，因为@support目前还无法广泛使用，所以在此不表）。我们默认的@at-root其实就是@at-root (without:rule)。

//sass style

//-------------------------------

//跳出父级元素嵌套

@media print {

.parent1{

color:#f00;

@at-root .child1 {

width:200px;

}

}

}

//跳出media嵌套，父级有效

@media print {

.parent2{

color:#f00;

@at-root (without: media) {

.child2 {

width:200px;

}

}

}

}

//跳出media和父级

@media print {

.parent3{

color:#f00;

@at-root (without: all) {

.child3 {

width:200px;

}

}

}

}

//sass style

//-------------------------------

@media print {

.parent1 {

color: #f00;

}

.child1 {

width: 200px;

}

}

@media print {

.parent2 {

color: #f00;

}

}

.parent2 .child2 {

width: 200px;

}

@media print {

.parent3 {

color: #f00;

}

}

.child3 {

width: 200px;

}

**@at-root与&配合使用**

//sass style

//-------------------------------

.child{

@at-root .parent &{

color:#f00;

}

}

//css style

//-------------------------------

.parent .child {

color: #f00;

}

**应用于@keyframe**

//sass style

//-------------------------------

.demo {

...

animation: motion 3s infinite;

@at-root {

@keyframes motion {

...

}

}

}

//css style

//-------------------------------

.demo {

...

animation: motion 3s infinite;

}

@keyframes motion {

...

}

## 混合(mixin)

**sass中使用@mixin声明混合**，可以**传递参数**，参数名以**$**符号开始，多个参数以**逗号**分开，也可以给参数设置默认值。声明的@mixin通过@include来调用。

**无参数mixin**

//sass style

//-------------------------------

@mixin center-block {

margin-left:auto;

margin-right:auto;

}

.demo{

@include center-block;

}

//css style

//-------------------------------

.demo{

margin-left:auto;

margin-right:auto;

}

**有参数mixin**

//sass style

//-------------------------------

@mixin opacity($opacity:50) {

opacity: $opacity / 100;

filter: alpha(opacity=$opacity);

}

//css style

//-------------------------------

.opacity{

@include opacity; //参数使用默认值

}

.opacity-80{

@include opacity(80); //传递参数

}

**多个参数mixin**

调用时可直接传入值，如**@include传入参数的个数小于@mixin定义参数的个数，则按照顺序表示，后面不足的使用默认值，如不足的没有默认值则报错**。除此之外还可以选择性的传入参数，使用参数名与值同时传入。

//sass style

//-------------------------------

@mixin horizontal-line($border:1px dashed #ccc, $padding:10px){

border-bottom:$border;

padding-top:$padding;

padding-bottom:$padding;

}

.imgtext-h li{

@include horizontal-line(1px solid #ccc);

}

.imgtext-h--product li{

@include horizontal-line($padding:15px);

}

//css style

//-------------------------------

.imgtext-h li {

border-bottom: 1px solid #cccccc;

padding-top: 10px;

padding-bottom: 10px;

}

.imgtext-h--product li {

border-bottom: 1px dashed #cccccc;

padding-top: 15px;

padding-bottom: 15px;

}

**多组值参数mixin**

如果一个参数可以有多组值，如box-shadow、transition等，那么参数则需要在变量后加三个点表示，如$variables...。

//sass style

//-------------------------------

//box-shadow可以有多组值，所以在变量参数后面添加...

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

-webkit-box-shadow:$shadow;

box-shadow:$shadow;

}

.box{

border:1px solid #ccc;

@include box-shadow(0 2px 2px rgba(0,0,0,.3),0 3px 3px rgba(0,0,0,.3),0 4px 4px rgba(0,0,0,.3));

}

//css style

//-------------------------------

.box{

border:1px solid #ccc;

-webkit-box-shadow:0 2px 2px rgba(0,0,0,.3),0 3px 3px rgba(0,0,0,.3),0 4px 4px rgba(0,0,0,.3);

box-shadow:0 2px 2px rgba(0,0,0,.3),0 3px 3px rgba(0,0,0,.3),0 4px 4px rgba(0,0,0,.3);

}

### @content

@content在sass3.2.0中引入，可以用来解决css3的@media等带来的问题。它可以使@mixin接受一整块样式，接受的样式从@content开始。

//sass style

//-------------------------------

@mixin max-screen($res){

@media only screen and ( max-width: $res )

{

@content;

}

}

@include max-screen(480px) {

body { color: red }

}

//css style

//-------------------------------

@media only screen and (max-width: 480px) {

body { color: red }

}

PS：@mixin通过@include调用后解析出来的样式是以拷贝形式存在的，而下面的继承则是以联合声明的方式存在的，所以从3.2.0版本以后，建议传递参数的用@mixin，而非传递参数类的使用下面的继承%。

## 继承

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**。

//sass style

//-------------------------------

h1{

border: 4px solid #ff9aa9;

}

.speaker{

@extend h1;

border-width: 2px;

}

//css style

//-------------------------------

h1,.speaker{

border: 4px solid #ff9aa9;

}

.speaker{

border-width: 2px;

}

### 占位选择器%

从sass 3.2.0以后就可以定义占位选择器%。这种选择器的优势在于：如果不调用则不会有任何多余的css文件，避免了以前在一些基础的文件中预定义了很多基础的样式，然后实际应用中不管是否使用了@extend去继承相应的样式，都会解析出来所有的样式。占位选择器以%标识定义，**通过@extend调用**。

//sass style

//-------------------------------

%ir{

color: transparent;

text-shadow: none;

background-color: transparent;

border: 0;

}

%clearfix{

@if $lte7 {

\*zoom: 1;

}

&:before,

&:after {

content: "";

display: table;

font: 0/0 a;

}

&:after {

clear: both;

}

}

#header{

h1{

@extend %ir;

width:300px;

}

}

.ir{

@extend %ir;

}

//css style

//-------------------------------

#header h1,

.ir{

color: transparent;

text-shadow: none;

background-color: transparent;

border: 0;

}

#header h1{

width:300px;

}

如上代码，定义了两个占位选择器%ir和%clearfix，其中%clearfix这个没有调用，所以解析出来的css样式也就没有clearfix部分。占位选择器的出现，使css文件更加简练可控，没有多余。所以可以用其定义一些基础的样式文件，然后根据需要调用产生相应的css。

**ps**：在@media中暂时不能@extend @media外的代码片段，以后将会可以。

## 函数

sass定义了很多函数可供使用，当然你也可以自己定义函数，以@fuction开始。sass的官方函数链接为：[sass fuction](http://sass-lang.com/documentation/Sass/Script/Functions.html)，实际项目中我们使用最多的应该是颜色函数，而颜色函数中又以lighten减淡和darken加深为最，其调用方法为lighten($color,$amount)和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。

//sass style

//-------------------------------

$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%);

}

//css style

//-------------------------------

body{

font-size:10px;

color:#E6E6E6;

}

.test{

font-size:1.6rem;

color:#B3B3B3;

}

关于@mixin，%，@function更多说明可参阅：

* [sass揭秘之@mixin，%，@function](http://www.w3cplus.com/preprocessor/sass-mixins-function-placeholder.html)
* [Sass基础——颜色函数](http://www.w3cplus.com/preprocessor/sass-color-function.html)
* [Sass基础——Sass函数](http://www.w3cplus.com/preprocessor/sass-other-function.html)

## 运算

sass具有运算的特性，可以对数值型的Value(如：数字、颜色、变量等)进行加减乘除四则运算。请注意运算符前后请留一个空格，不然会出错。

$baseFontSize: 14px !default;

$baseLineHeight: 1.5 !default;

$baseGap: $baseFontSize \* $baseLineHeight !default;

$halfBaseGap: $baseGap / 2 !default;

$samllFontSize: $baseFontSize - 2px !default;

//grid

$\_columns: 12 !default; // Total number of columns

$\_column-width: 60px !default; // Width of a single column

$\_gutter: 20px !default; // Width of the gutter

$\_gridsystem-width: $\_columns \* ($\_column-width + $\_gutter); //grid system width

## 条件判断及循环

### @if判断

**@if可一个条件单独使用，也可以和@else结合多条件使用**

//sass style

//-------------------------------

$lte7: true;

$type: monster;

.ib{

display:inline-block;

@if $lte7 {

\*display:inline;

\*zoom:1;

}

}

p {

@if $type == ocean {

color: blue;

} @else if $type == matador {

color: red;

} @else if $type == monster {

color: green;

} @else {

color: black;

}

}

//css style

//-------------------------------

.ib{

display:inline-block;

\*display:inline;

\*zoom:1;

}

p {

color: green;

}

### 三目判断

**语法为：if($condition, $if\_true, $if\_false) 。三个参数分别表示：条件，条件为真的值，条件为假的值。**

if(true, 1px, 2px) => 1px

if(false, 1px, 2px) => 2px

### for循环

for循环有两种形式，分别为：**@for $var from <start> through <end>**和**@for $var from <start> to <end>**。$i表示变量，start表示起始值，end表示结束值，这两个的区别是**关键字through表示包括end这个数，而to则不包括end这个数**。

//sass style

//-------------------------------

@for $i from 1 through 3 {

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

}

//css style

//-------------------------------

.item-1 {

width: 2em;

}

.item-2 {

width: 4em;

}

.item-3 {

width: 6em;

}

### @each循环

**语法为：@each $var in <list or map>。其中$var表示变量，而list和map表示list类型数据和map类型数据。sass 3.3.0新加入了多字段循环和map数据循环。**

**单个字段list数据循环**

//sass style

//-------------------------------

$animal-list: puma, sea-slug, egret, salamander;

@each $animal in $animal-list {

.#{$animal}-icon {

background-image: url('/images/#{$animal}.png');

}

}

//css style

//-------------------------------

.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');

}

.salamander-icon {

background-image: url('/images/salamander.png');

}

**多个字段list数据循环**

//sass style

//-------------------------------

$animal-data: (puma, black, default),(sea-slug, blue, pointer),(egret, white, move);

@each $animal, $color, $cursor in $animal-data {

.#{$animal}-icon {

background-image: url('/images/#{$animal}.png');

border: 2px solid $color;

cursor: $cursor;

}

}

//css style

//-------------------------------

.puma-icon {

background-image: url('/images/puma.png');

border: 2px solid black;

cursor: default;

}

.sea-slug-icon {

background-image: url('/images/sea-slug.png');

border: 2px solid blue;

cursor: pointer;

}

.egret-icon {

background-image: url('/images/egret.png');

border: 2px solid white;

cursor: move;

}

**多个字段map数据循环**

//sass style

//-------------------------------

$headings: (h1: 2em, h2: 1.5em, h3: 1.2em);

@each $header, $size in $headings {

#{$header} {

font-size: $size;

}

}

//css style

//-------------------------------

h1 {

font-size: 2em;

}

h2 {

font-size: 1.5em;

}

h3 {

font-size: 1.2em;

}

关于循环判断详细分析请查阅：[sass揭秘之@if，@for，@each](http://www.w3cplus.com/preprocessor/sass-advanced-application.html)