## 学习Sass

Sass与less的区别

1. Sass由于是使用Ruby编写的，所以编译的时候是在服务器端处理；而 Less由于是使用JavaScript编写的，所以编译的时候是在浏览器端处 理；
2. Sass拥有更为强大的功能，如循环、函数、混合宏等，而less却没有；
3. Sass拥有成熟稳定的框架来辅助开发，特别是Compass，而less却没 有；
4. Sass在国内外讨论热度最大，并且有一个稳定强大的团队在维护；
5. 相当多的公司更为倾向于使用Sass，而不是less；
6. 安装ruby
7. 下载路径

https://rubyinstaller.org/downloads/

1. 查看版本

ruby -v

1. 安装sass
2. 安装

gem install sass

1. 查看版本

sass -v

1. 更新

gem update sass

1. 卸载

gem uninstall sass

1. 安装compass(可不安装)
2. 安装

gem install compass

1. 查看版本

compress -v

1. 编译sass
2. 单文件转换(index.scss编译成index.css)

sass index.scss index.css

1. 单文件实时监听(index.scss编译成index.css)

sass --watch index.scss:index.css

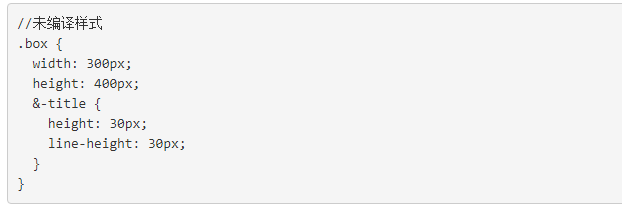
1. 如果你有很多的sass文件的目录，你也可以告诉sass监听整个目录
2. 监听sass文件夹下的index.scss文件编译到public文件夹下的 index.css文件

sass --watch sass/index.scc:public/index.css --style expanded

1. 监听sass下的所有.scss文件编译到public文件夹下

sass --watch sass/:public --style expanded

1. 编译排版格式( 监听可以加上 --watch) 四种编译模式(输出风格)
2. 编辑之前的scss样式



1. 嵌套式 nested 默认格式

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



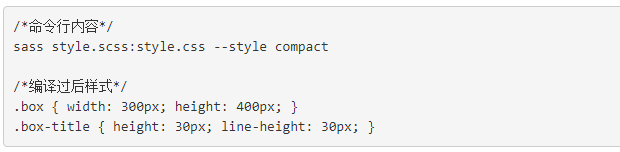
1. 展开式 expanded

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



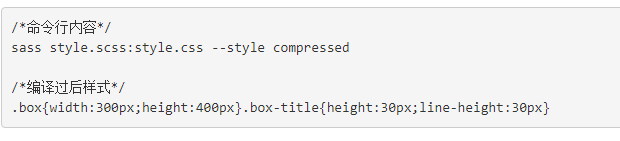
1. 紧凑式 compact

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



1. 压缩式 compressed

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



1. 反编译
2. 将fileName.css文件反编译成fileName.scss

sass-convert index.css index.scss

1. 将fileName.scss文件编译成fileName.sass格式

sass-convert index.scss index.sass

1. 注释
2. 单行注释，只保留在sass源文件中，编译后会被省略掉

//不会出现在编译之后任何格式的CSS文件中。

1. 多行注释，会保留到编译后的文件中

/\* 只会出现在编译之后代码格式为expanded的CSS文件中。 \*/

1. 强制注释(重要注释)，即使压缩编译模式，也会保留注释，通常用于声明 版权信息

/\*! 会出现在任何代码格式的CSS文件中。 \*/

1. 导入：sass中如导入其他sass文件，最后编译为一个css文件， 优于纯css的@import
2. [@import命令，用来插入外部文件，注意默认无后缀名时是引入.sass](mailto:1.@import命令，用来插入外部文件，注意默认无后缀名时是引入.sass) 或.scss文件。

**@import “**path/fileName”;

导入Sass但不输出Css--局部文件：那些专门为 @import 命令而写的 Sass 文件，并不需要生成对应的独立 CSS 文件，这样的 Sass 文件被 称为局部文件。对此，Sass约定局部文件的文件名以下划线 \_ 开头。 这样，Sass 就不会在编译时单独编译这个文件输出 CSS 了，而仅仅是 把这个文件作为导入用途。Sass 更高效的地方在于，导入这样的文件时， 我们还可以省略文件名中开头的下划线 \_，例如，我们需要导入文件 themes/\_night-sky.scss，我们可以仅仅写

**@import** “themes/night-sky”

1. 以下几种情况插入的是.css文件，则等同于css的import命令

**@import** "foo.css"; *//后缀名为.css*

@import "css/style.css"; *//被导入文件的名字是CSS的url()值*

@import "url(.....)"; *//被导入文件是一个URL地址形式给出的*

1. 和css中@import的区别
2. css中用到了再去下载，而sass编译时直接导入到当前文件
3. sass中导入可省略后缀名，默认为.sass格式或者.scss格式
4. Mixin是可以重用的代码块
5. @minxin基础用法

**@mixin left**{

float : left;

margin-left : 10x;

}

*//使用@include命令，调用这个mixin*

**div**{

**@include** left;

}

*//编译后输出*

**div**{

float : left;

margin-left : 10px;

}

1. @mixin传递参数

**@mixin** font($color, $fontSize: 14px) {

color: $color;

font-size: $fontSize;

}

*//使用@include命令，调用这个mixin*

**p** { **@include** font(blue); }

**h1** { **@include** font(blue, 20px); }

**h2** { **@include** font($fontSize: 18px,$color: red); }  
 *//编译后输出*

**p** {

color: blue;

font-size: 14px;

}

**h1** {

color: blue;

font-size: 20px;

}

**h2** {

color: red;

font-size: 18px;

}

1. @mixin 传递多值参数

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

-moz-box-shadow: $shadows;

-webkit-box-shadow: $shadows;

box-shadow: $shadows;

}

*//使用@include命令，调用这个mixin*

.shadows {

**@include** box-shadow(0px 4px 5px #666, 2px 6px 10px #999);

}

*//编译后输出*

.shadows {

-moz-box-shadow: 0px 4px 5px #666, 2px 6px 10px #999;

-webkit-box-shadow: 0px 4px 5px #666, 2px 6px 10px #999;

box-shadow: 0px 4px 5px #666, 2px 6px 10px #999;

}

此外，多值参数还可以用在 @include 传參的时候，分解某个变量值，

例如:

**@mixin** colors($text, $background, $border) {

color: $text;

background-color: $background;

border-color: $border;

}

$values:red,blue,pink;

.primary {

**@include** colors($values...);

}

$value-map: (text: #00ff00, background: #0000ff, border: #ff0000);

.secondary {

**@include** colors($value-map...);

}

*//编译后输出*

.primary {

color: red;

background-color: lime;

border-color: blue;

}

.secondary {

color: lime;

background-color: blue;

border-color: red;

}

1. 向 @mixin 传递内容

定义的一个选择器或者一段代码片段，在其中添加了 @content 这个 命令。当使用 @include 命令使用这条 mixin 的时候，将一段内容包 裹了起来，这段被包裹的内容就会替换掉 @mixin 中的 @content。

**@mixin** apply-to-iphone5 {

**@media** **only** screen **and** (min-device-width : 320px) **and** (max-device-width : 568px) {

**@content**;

}

}

**@include** apply-to-iphone5 {

**#logo** {

background-image: url(/logo@2x.gif);

}

}

*//编译后输出*

**@media** **only** screen **and** (min-device-width: 320px) **and**

(max-device-width: 568px) {

**#logo** {

background-image: url(/logo@2x.gif);

}

}

1. @mixin和@extend的区别：

@mixin 定义的是一个片段，这个片段可以是类似变量的一段文字一条属 性，也可以是一整个选择器和内容，也可以是一个选择器的一部分css 代码。 此外还可以传递参数，通过参数生成不同代码。它需要配合 @inclde 命令 来引用这段代码，类似复制的效果。@mixin 定义的内 容，不会编译输出。

@extend 就是简单的扩展，基于某个选择器，将其他类似需求的选择器 挂靠 上，以提高复用程度

1. 扩展/继承(@extend)

sass允许一个选择器，继承另一个选择器。

如，class2继承class1的属性：

.class1{

border : 1px solid #ddd;

}

.class2{

**@extend** .class1;

font-size : 12px;

}

*//编译后输出*

.class1{

border : 1px solid #ddd;

}

.class2{

border : 1px solid #ddd;

font-size : 12px;

}

十. 自定义函数

1.自定义函数可传参数

//函数名称

@function doubleWidth($n) {

　　　　@return $n \* 2;

　　 }

//引用

#sidebar {

　　　　width: doubleWidth(5px);

}

//编译后输出

#sidebar{

width: 10px;

}

十一. 颜色函数

1. RGB

颜色中的一种表达式，其中 R 是 red 表示红色，G 是 green 表示绿 色而 B 是 blue 表示蓝色。

rgb(red,green,$blue)：根据红、绿、蓝三个值创建一个颜色；

rgba(red,green,blue,alpha)：根据红、绿、蓝和透明度值创建一个颜色；

red($color)：从一个颜色中获取其中红色值；

green($color)：从一个颜色中获取其中绿色值；

blue($color)：从一个颜色中获取其中蓝色值；

mix(color-1,color−1,color-2,[$weight])：把两种颜色混合根据混合比例混合在一 起，默认值50%。

1. lighten()和darken()函数

这两个函数都是围绕颜色的亮度值做调整的；

其中lighten()函数会让颜色变得更亮，与之相反的darken()函数会让 颜色变得更暗。这个亮度值可以是0~1之间，不过常用的一般都在 3%~20%之间。

例如：

$baseColor: #ad141e;

.lighten {

background: lighten($baseColor,10%);

}

.darken{

background: darken($baseColor,10%);

}

1. Ie-hex-str()函数

这个函数常配合IE滤镜使用，主要用来转译出适合IE滤镜的颜色代码

例如：

ie-hex-str(#abc) // "#FFAABBCC"

ie-hex-str(#3322BB) // "#FF3322BB"

ie-hex-str(rgba(0, 255, 0, 0.5)) //"#8000FF00"

1. 流程控制语句
2. if判断

p {

@if 1 + 1 == 2 {

width:30px;

}@else {

width:100px;

}

}

//编译后输出

p { width: 30px; }

1. @for

有两种形式，分别为：@for $var from through 和 @for $var from to

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。

$i表示变量，start表示起始值，end表示结束值，

$nums:5 !default;  
 $width:60px !default;

$diff:20px !default;

%**span**-**base**{

float:left;

margin-left:$diff/ 2;

margin-right:$diff/ 2;

}  
 **@for** $i from 1 through $nums{

.span#{$i} {

**@extend** %span-base;

width:($width+ $diff) \* $i - $diff;

}

}

//编译后输出

.span1, .span2, .span3, .span4, .span5{

float: left;

margin-left: 10px;

margin-right: 10px;

}

.span1 {

width: 60px;

}

.span2 {

width: 140px;

}

.span3 {

width: 220px;

}

.span4 {

width: 300px;

}

.span5 {

width: 380px;

}

1. @while

@while 和 @for 命令是非常接近的，只不过 @while 循环可以自己 定义步长，多次输出，直到该语句的嵌套样式的计算结果为false。

$i: 6;

**@while** $i > 0 {

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

$i: $i - 2;

}

//编译后输出

.item-6 {width: 12px;}

.item-4 {width: 8px;}

.item-2 {width: 4px; }

1. @each

@each $var in

$sprite: puma sea-slug egret salamander !default;

%bg-image{

background: url('/images/animal.png') no-repeat;

}

**@each** $i in $sprite {

.#{$i}-icon {

**@extend** %bg-image;

background-position:0 -(index($sprite,$i)\*30px);

}

}

//编译后输出

.puma-icon, .sea-slug-icon, .egret-icon, .salamander-icon {

background: url("/images/animal.png") no-repeat;

}

.puma-icon {

background-position: -30px;

}

.sea-slug-icon {

background-position: -60px;

}

.egret-icon {

background-position: -90px;

}

.salamander-icon {

background-position: -120px;

}

1. 数据类型

（1）数字值；

（2）字符串；

（3）布尔值；

（4）颜色值；

（5）列表值；

（6）Map值**；**

（7）空值null；