# 理解SASS的嵌套，@extend，%Placeholders和Mixins

作者：大漠 日期：2013-07-31 点击：3384

[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)

在《[SASS基础教程——SASS基本语法与特性](http://www.w3cplus.com/preprocessor/sass-basic-syntax-and-features.html" \t "http://www.w3cplus.com/preprocessor/_blank)》文中主要介绍了[SASS](http://sass-lang.com/" \t "http://www.w3cplus.com/preprocessor/_blank)的基本语法和特性。简单的知道[SASS](http://sass-lang.com/" \t "http://www.w3cplus.com/preprocessor/_blank)具有四个基本特性：****变量Variables****、****嵌套Nesting****、****混合Mixins****和****继承Selector Inheritance****。其实这四种特性中的嵌套、混合和继承是有一种千丝万缕的关系，甚至会让初学者理不清，这也从侧面也说明了这三者在SASS的重要性。那么今天这篇教程，我们将主要介绍和探讨****嵌套****、****混合****、****继承****三者之间的关系，以及各自的优缺点。

## **回顾嵌套、混合和继承特性**

如果您没有阅读过《[SASS基础教程——SASS基本语法与特性](http://www.w3cplus.com/preprocessor/sass-basic-syntax-and-features.html" \t "http://www.w3cplus.com/preprocessor/_blank)》一文，并不要紧。我们一起简单回顾一下SASS中的****嵌套Nesting****、****混合Mixins****和****继承Selector Inheritance****。

#### 嵌套Nesting

仅从字面上理解，嵌套就是一层一层往里套。在DOM，元素与元素之间除了存在兄弟间关系之外，还存有一种父级（祖级）关系。在CSS中是依靠选择器层层深入或者添加额外的类名或ID来控制。那么在SASS中添加了对DOM的嵌套功能。即，元素的所有后代元素都可以放置在父元素之中，如：

<nav>

<ul>

<li><a href="#">Home</a></li>

<li><a href="#">Blog</a></li>

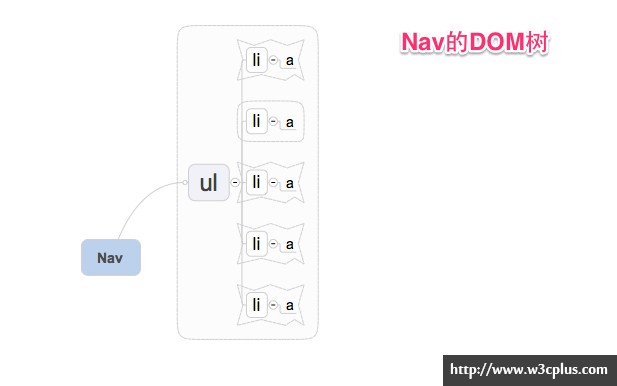
<li><a href="#">Sass</a></li>

<li><a href="#">Less</a></li>

<li><a href="#">Haml</a></li>

</ul></nav>

我们给上面的结构画一个DOM树：



为了实现下图的效果：



我们的样式大致如下：

nav {

display: block;}

nav ul {

margin: 50px auto;

width: 800px;

width: -moz-fit-content;

width: -webkit-fit-content;

width: -o-fit-content;

width: fit-content;

padding: 0;

list-style: none;}nav ul:before,nav ul:after{

content:"";

display: table;}nav ul:after {

clear:both;

overflow: hidden;}nav ul li {

background: #34495e;

float: left;

-webkit-transform: skewX(25deg);

-moz-transform: skewX(25deg);

-o-transform: skewX(25deg);

-ms-transform: skewX(25deg);

transform: skewX(25deg);}nav ul li a {

display: block;

color: #fff;

text-transform: uppercase;

text-decoration: none;

font-family: Arial,Helvetica;

font-size: 14px;

-webkit-transform: skewX(-25deg);

-moz-transform: skewX(-25deg);

-o-transform: skewX(-25deg);

-ms-transform: skewX(-25deg);

transform: skewX(-25deg);

padding: 1em 2em;}nav li:hover {

background: #e74c3c;}

如果在SASS中写，那就要简单得多了，我们可以使用SASS的嵌套来处理：

nav {

display: block;

ul {

margin: 50px auto;

width: 800px;

width: -moz-fit-content;

width: -webkit-fit-content;

width: -o-fit-content;

width: fit-content;

padding: 0;

list-style: none;

&:before,

&:after {

content: "";

display: table;

}

&:after {

clear: both;

overflow: hidden;

}

li {

background: *#34495e;*

float: left;

-webkit-transform: skewX(25deg);

-moz-transform: skewX(25deg);

-o-transform: skewX(25deg);

-ms-transform: skewX(25deg);

transform: skewX(25deg);

&:hover {

background: *#e74c3c;*

}

a {

display: block;

color: *#fff;*

text-transform: uppercase;

text-decoration: none;

font-family: Arial,Helvetica;

font-size: 14px;

padding: 1em 2em;

-webkit-transform: skewX(-25deg);

-moz-transform: skewX(-25deg);

-o-transform: skewX(-25deg);

-ms-transform: skewX(-25deg);

transform: skewX(-25deg);

}

}

}

}

把上面的SASS代码编译完成后，编译出来的CSS和前面展示的CSS一样。

SASS除了能进行结构嵌套之处，还可以对属性进行嵌套，例如上面的示例之中：

a {

…

text-transform: uppercase;

text-decoration: none;

font-family: Arial,Helvetica;

font-size: 14px;

...

}

我们可以将上面的SASS代码进行属性嵌套：

a {

…

text: {

transform: uppercase;

decoration: none;

}

font:{

family: Arial,Helvetica;

size: 14px;

}

...

}

这样的SASS代码并不完美，此处只是通过这样的一个小例，向大家演示SASS中的****嵌套****特性。

#### 混合Mixins

****Mixins****是SASS最出名特色之一。他充许我们通过:

@mixin Mixins名称（参数:参数值）{

/\*公用样式\*/

}

的方式将相同的样式风格定义成一个模块，然后在需要使用的地方通过@include将@mixin定义好的模块调用进来：

selector {

@includ Mixins名称(参数值);

}

Mixins最明显的用例就是用来处理CSS3属性前缀，回到我们上面的示例之中，我们在示例中使用了两个CSS3属性：

nav ul {

…

width: -moz-fit-content;

width: -webkit-fit-content;

width: -o-fit-content;

width: fit-content;

…

}

nav ul li {

…

-webkit-transform: skewX(25deg);

-moz-transform: skewX(25deg);

-o-transform: skewX(25deg);

-ms-transform: skewX(25deg);

transform: skewX(25deg);

...

}

nav ul li a {

…

-webkit-transform: skewX(-25deg);

-moz-transform: skewX(-25deg);

-o-transform: skewX(-25deg);

-ms-transform: skewX(-25deg);

transform: skewX(-25deg);

…

}

针对上面的两个CSS3属性，我们可以定义两个Mixins：

*//define fit-content*

@mixin fit-content() {

width: -webkit-fit-content;

width: -moz-fit-content;

width: -o-fit-content;

width: -ms-fit-content;

width: fit-content;

}

*//define transform skewX()*

@mixin skewX($degrees){

-webkit-transform: skewX($degrees);

-moz-transform: skewX($degrees);

-o-transform: skewX($degrees);

-ms-transform: skewX($degrees);

transform: skewX($degrees);

}

这个时候，我们只需要在对应的地方通过@include调用即可：

nav {

display: block;

ul {

margin: 50px auto;

width: 800px;

@include fit-content();//调用fit-content()

padding: 0;

list-style: none;

&:before,

&:after {

content: "";

display: table;

}

&:after {

clear: both;

overflow: hidden;

}

li {

background: *#34495e;*

float: left;

@include skewX(25deg);//调用skewX()，并传参数值25deg

&:hover {

background: *#e74c3c;*

}

a {

display: block;

color: *#fff;*

text: {

transform: uppercase;

decoration: none;

}

font: {

family: Arial,Helvetica;

size: 14px;

}

padding: 1em 2em;

@include skewX(-25deg);//调用skewX(),并传参数值-25deg

}

}

}

}

这样编译出来的CSS和文章前头显示的CSS一模一样。

当然，上面定义的@mixin skewX()并不是完美的，完美的可以参考一下：[Bourbon](http://bourbon.io/" \t "http://www.w3cplus.com/preprocessor/_blank)或者[Bootstrap SASS](https://github.com/thomas-mcdonald/bootstrap-sass/blob/master/vendor/assets/stylesheets/bootstrap/_mixins.scss" \t "http://www.w3cplus.com/preprocessor/_blank),此处仅做一下演示，详细使用与分析，将会放在CSS3的Mixins中介绍。

*//Example: @include prefixer(border-radius, $radius, webkit spec);//----------------------------------------*

$prefix-for-webkit: true !default;$prefix-for-mozilla: true !default;$prefix-for-microsoft: true !default;$prefix-for-opera: true !default;$prefix-for-spec: true !default; *// required for keyframe mixin*

*//prefixer*

@mixin prefixer ($property, $value, $prefixes) {

@each $prefix in $prefixes {

@if $prefix == webkit and $prefix-for-webkit == true {

-webkit-*#{$property}: $value;*

}

@else if $prefix == moz and $prefix-for-mozilla == true {

-moz-*#{$property}: $value;*

}

@else if $prefix == ms and $prefix-for-microsoft == true {

-ms-*#{$property}: $value;*

}

@else if $prefix == o and $prefix-for-opera == true {

-o-*#{$property}: $value;*

}

@else if $prefix == spec and $prefix-for-spec == true {

*#{$property}: $value;*

}

@else {

@warn "Unrecognized prefix: #{$prefix}";

}

}

}

@mixin skewX($degrees) {

@include prefixer(transform, skewX($degrees), webkit moz o ms spec);

-webkit-backface-visibility: hidden;

}

#### 继承Selector Inheritance

很多时候，我们在写CSS时，多个元素具有同样的样式，我们往往解决的文案是把多个元素相同的样式写在一起，例如：

ul,ol,div{*/\*相同的样式\*/*}

幸运的是，在SASS中，我们可以把这个相同的样式抽取出来，并给他定义为一个类，把相同的样式写在这个类中：

.sameStyle {*/\*相同的样式\*/*}

然后通过@extend来调用。

selector {

@extend sameStyle;

}

同样在上面的实例为例，把ul中的清除浮动单独拿出来，定义一个.clearfix：

.clearfix {

\*zoom: 1;

&:after,

&:before {

content: "";

display: table;

}

&:after {

clear: both;

overflow: hidden;

}

}

并在ul中通过@extend调用刚才定义的.clearfix:

ul {

…

@extend .clearfix;

…

}

这个时候解析出来的CSS就变成：

.clearfix, nav ul {

\*zoom: 1; }.clearfix:after, nav ul:after, .clearfix:before, nav ul:before {

content: "";

display: table; }.clearfix:after, nav ul:after {

clear: both;

overflow: hidden; }

前面通过一个简单的实例，向大家演示了SASSS中****嵌套****、****混合****和****继承****三大特性的优点。好生让人羡慕。此时或许你会问，难道他们就只有优点吗？难道就没有一点缺点吗？因为我对于一个新东西，我看到他的优点，我就会去想，那么他有什么缺点呢？因为万物都是具有两面性的，有圆就有缺。那么SASS中的这三大特性也逃不了这个自然定律。接下来我们一起来看他们之间的不足之处。

## **SASS中嵌套、混合和继承的缺点**

如果你有仔细看上面实例中SASS编译出来的CSS代码，你就不难发现，上面的代码中都存在一些问题，下面我们逐一来看其中的不足之处。

#### 嵌套的缺点

SASS的嵌套让我们在写代码的时候，不再需要考虑如何嵌套，如何添加类或者ID。简单点说，知道了父元素，其后代元素都可以很清晰的控制。但是生成出来的CSS有时候就不尽人意。我们回到上面的实例中，先看一下由SASS编译出来的CSS：

nav {}nav ul {}nav ul li{}nav ul li:hover{}nav ul li a {}

但在我们这个实例中，如果使用CSS来编辑代码的话，元素选择器完全不需要嵌套这么深，我们可以简单的使用：

nav {}nav ul {}nav li {}nav li:hover {}nav a {}

从上面的实例中，我们可以知道，只要你的SASS嵌套的越深，那么编译出来的CSS的选择器就会层级越深。这样一来并不是好事，也不是我们想要的干净代码，换句话来说，直接违背了我们使用SASS的初衷。更重要的是，SASS的嵌套编译出来的CSS，直接会影响页面的性能。别的方面不多，就仅出CSS的深层次的选择器来讲，就是不是最佳的。有关于选择器性有方面的相关知识，大家感兴趣的话，可以阅读下面的文章，这里不做深层次的展开。

##### 选择器性能扩展阅读

* [CSS选择器的优化](http://www.w3cplus.com/css/css-selector-performance" \t "http://www.w3cplus.com/preprocessor/_blank)
* [不同CSS技术及其CSS性能](http://www.zhangxinxu.com/wordpress/?p=1045" \t "http://www.w3cplus.com/preprocessor/_blank)
* [选择器性能](http://zwb.me/smacss/selectors/" \t "http://www.w3cplus.com/preprocessor/_blank)
* [关注CSS性能二](http://peters-playground.com/2013/04/css-performance-2/" \t "http://www.w3cplus.com/preprocessor/_blank)
* [提高 web 应用性能之 CSS 性能调优](http://www.ibm.com/developerworks/cn/web/1109_zhouxiang_optcss/" \t "http://www.w3cplus.com/preprocessor/_blank)
* [编写高效的 CSS 选择器](http://blog.jobbole.com/35339/" \t "http://www.w3cplus.com/preprocessor/_blank)
* [CSS选择器的性能影响](http://www.99css.com/archives/33" \t "http://www.w3cplus.com/preprocessor/_blank)
* [高性能CSS](http://www.alloyteam.com/2012/10/high-performance-css/" \t "http://www.w3cplus.com/preprocessor/_blank)
* [关于css通配符性能问题不完全测试](http://i.wanz.im/2012/01/03/performance_testing_about_css_universal_selector/" \t "http://www.w3cplus.com/preprocessor/_blank)
* [网站CSS选择器性能讨论](http://www.aliued.cn/2013/01/24/%E7%BD%91%E7%AB%99css%E9%80%89%E6%8B%A9%E5%99%A8%E6%80%A7%E8%83%BD%E8%AE%A8%E8%AE%BA.html" \t "http://www.w3cplus.com/preprocessor/_blank)
* [Writing efficient CSS selectors](http://csswizardry.com/2011/09/writing-efficient-css-selectors/" \t "http://www.w3cplus.com/preprocessor/_blank)
* [Writing efficient CSS](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Writing_efficient_CSS" \t "http://www.w3cplus.com/preprocessor/_blank)
* [Efficiently Rendering CSS](http://css-tricks.com/efficiently-rendering-css/" \t "http://www.w3cplus.com/preprocessor/_blank)
* [CSS Selector Performance: Front-End Myths](http://architects.dzone.com/articles/css-selector-performance-front" \t "http://www.w3cplus.com/preprocessor/_blank)
* [CSS Selector Performance has changed! (For the better)](http://calendar.perfplanet.com/2011/css-selector-performance-has-changed-for-the-better/" \t "http://www.w3cplus.com/preprocessor/_blank)
* [Selector Performance](http://smacss.com/book/selectors" \t "http://www.w3cplus.com/preprocessor/_blank)
* [Performance Impact of CSS Selectors](http://www.stevesouders.com/blog/2009/03/10/performance-impact-of-css-selectors/" \t "http://www.w3cplus.com/preprocessor/_blank)

#### 混合的缺点

虽然Mixins能帮助我们把相同的样式通过@mixin来定义成一个模块，在所需之时调用。然而混合在生成的CSS代码中也存在一个潜在的不足，而且很多初学者都将Mixins用在错误的地方。来看一下圆角的使用的案例。

//define @mixin rounded @mixin rouded{

-webkit-border-radius: 5px;

-moz-border-radius: 5px;

-o-border-radius: 5px;

-ms-border-radius: 5px;

border-radius: 5px;

}

button {

@include rounded;

background: *#ccc;*

color: *#222;*

}

//Several lines down, in the same file, or even in another different file

.simple-form input {

@include rounded;

}

.main-nav .item {

color: *#fff;*

a:hover,

a:active {

@include rounded;

}

}

这是一个很普通，也很常见的案例，但是上面的SASS代码编译出来之后，你还会喜欢？

button {

-webkit-border-radius: 5px;

-moz-border-radius: 5px;

-o-border-radius: 5px;

-ms-border-radius: 5px;

border-radius: 5px;

background: #ccc;

color: #222; }

.simple-form input {

-webkit-border-radius: 5px;

-moz-border-radius: 5px;

-o-border-radius: 5px;

-ms-border-radius: 5px;

border-radius: 5px; }

.main-nav .item {

color: #fff; }

.main-nav .item a:hover,.main-nav .item a:active {

-webkit-border-radius: 5px;

-moz-border-radius: 5px;

-o-border-radius: 5px;

-ms-border-radius: 5px;

border-radius: 5px; }

很明显@minxin rounded在三个地方被调用：

button {

@include rounded;

}

.simple-form input {

@include rounded;

}

.main-nav .item a:hover,

.main-nav .item a:active {

@include rounded;

}

这样一来，@mixin rounded中定义的样式就被重复的编译出来：

button {

-webkit-border-radius: 5px;

-moz-border-radius: 5px;

-o-border-radius: 5px;

-ms-border-radius: 5px;

border-radius: 5px;}.simple-form input {

-webkit-border-radius: 5px;

-moz-border-radius: 5px;

-o-border-radius: 5px;

-ms-border-radius: 5px;

border-radius: 5px;}.main-nav .item a:hover,.main-nav .item a:active {

-webkit-border-radius: 5px;

-moz-border-radius: 5px;

-o-border-radius: 5px;

-ms-border-radius: 5px;

border-radius: 5px;}

这样用下来,编译出来的CSS代码不但没有变得简洁化，而且变得更臃肿。

button,.simple-form input,.main-nav .item a:hover,.main-nav .item a:active {

-webkit-border-radius: 5px;

-moz-border-radius: 5px;

-o-border-radius: 5px;

-ms-border-radius: 5px;

border-radius: 5px;}

此时你会反问，那@mixin是不是失去存的意义了。其实不是这样的，当你在使用@mixin时，你要知道他的使用规则。

****Mixins的黄金规则****是将相似的风格定义在一个@mixin中。请注意这里的一个关键词****相似的****，另外Mixins主要是用于重用，而不是用来指定具体的属性值。例如这个实例，我们应该用@mxin来创建不同半径的圆角，而不是用来创建一个具体值的@mixin。换句话来说，如果你创建的Mixins没有传参数，那您就是一种错误的使用方法。基于这点出发，我们可以把上例中的@mixin rounded传入一个$radius参数：

@mixin rounded($radius){

-webkit-border-radius: $radius;

-moz-border-radius: $radius;

-o-border-radius: $radius;

-ms-border-radius: $radius;

border-radius: $radius;

}

在@mixin中，我们除了可以传参之外，还可以给参数设置一个默认值：

@mixin rounded($radius:5px){

-webkit-border-radius: $radius;

-moz-border-radius: $radius;

-o-border-radius: $radius;

-ms-border-radius: $radius;

border-radius: $radius;

}

如此一来，我们就可以在调用的时候传入不同的参数值，当然，要是传入的参数值是一样的，同样会出现上面的现象。这是使用Mixins无法避免的。

#### 继承的缺点

前面说过，SASS的继承，可以将相同样式规则定义在一个类中，然后能过@extend来调用。这样就可以把相同样式合并在一起。按照这个原理，我们可以把上面的@mixin rounded替换成.rounded，然后在需要的地方通过@extend来调用.rounded。这样就可以解决使用@mixin致使样式重复出现多次的问题。

.rounded{

-webkit-border-radius: 5px;

-moz-border-radius: 5px;

-o-border-radius: 5px;

-ms-border-radius: 5px;

border-radius: 5px;

}

button {

@extend .rounded;

background: *#ccc;*

color: *#222;*

}

.simple-form input {

@extend .rounded;

}

.main-nav .item {

color: *#fff;*

a:hover,

a:active {

@extend .rounded;

}

}

将上面的SASS代码编译成CSS：

.rounded, button, .simple-form input, .main-nav .item a:hover,.main-nav .item a:active {

-webkit-border-radius: 5px;

-moz-border-radius: 5px;

-o-border-radius: 5px;

-ms-border-radius: 5px;

border-radius: 5px; }

button {

background: #ccc;

color: #222;}

.main-nav .item {

color: #fff; }

这样的代码相比使用@mixin编译出来的代码干净多了，但是继承在SASS中使用也存在一定的风险。

大家都知道，.rounded样式可能不只运用在一个地方或者一个样式文件之中。另外，@extend是可以读取SASS文件中类名。如此一来就给我们埋下了一个隐形炸弹。我们来看一个简单的实例：

.button {

display: block;

padding: 10px;

background: green;

}

.sidebar .signup .button {

margin-top: 20px;

}

.registrantion,

.remember-password {

.button {

margin-bottom: 33px;

}

}

.edit-account .delete-area .button {

background-color: red;

color: white;

}

.article a {

@extend .button;

}

在上面这段简单的SASS代码中.button一共出现过****四****次，我们来看编译出来的CSS代码：

.button, .article a {

display: block;

padding: 10px;

background: green; }

.sidebar .signup .button, .sidebar .signup .article a, .article .sidebar .signup a {

margin-top: 20px; }

.registrantion .button, .registrantion .article a, .article .registrantion a,.remember-password .button,.remember-password .article a,.article .remember-password a {

margin-bottom: 33px; }

.edit-account .delete-area .button, .edit-account .delete-area .article a, .article .edit-account .delete-area a {

background-color: red;

color: white; }

转译出来的CSS可能出乎你的意外，你原本可能只需要转译出来这样的代码：

.button, .article a {

display: block;

padding: 10px;

background: green; }

可是@extend .button之后，还编译出你不想的：

.sidebar .signup .button, .sidebar .signup .article a, .article .sidebar .signup a {

margin-top: 20px; }

.registrantion .button, .registrantion .article a, .article .registrantion a,.remember-password .button,.remember-password .article a,.article .remember-password a {

margin-bottom: 33px; }

.edit-account .delete-area .button, .edit-account .delete-area .article a, .article .edit-account .delete-area a {

background-color: red;

color: white; }

这可能让你大失所望。.button类名可能用在不同之处，有不同的容器包裹着，然而SASS中的@extend无法判断引用哪个地方的.button。所以他自己做主，将不同地方出现的.button类名都引入了进来，也就造成了上述你不想看到的现象。所以在使用SASS继承时有一个规则:

通过@extend引用的类名，你要有绝对的自信，它从未用在几个地方。

麻烦的是，你不能总是确定他未用在几个地方，就算是你确信引用的类名没用在别的地方。但你不敢保证，你未来你或你的同事不在别的地方引用这个类名，无意之中，你就会踩上这个地雷，把这个事情搞砸。

## **强大的%placeholders**

SASS3.2版本出现的placeholders%是SASS的一个强大的特性。使用%和@extend就可以将继承中埋下的地雷给排了。

%只是一个占位符，他不是正常的选择器，不像.classes或者#ids，只要不通过@extend调用，他是不会产生任何代码量。这个功能对于我们用他来取代.class与@extend是最完美的了。而且其使用方法也非常简单。

首先使用%placeholders定义一个公用样式，类似于.class：

%placeholders {*/\*公用样式\*/*}

在需要使用的地方通过@extend来调用：

selector {

@extend %placeholders;

}

我们来看一个简单的示例，将前面清除浮动的.clearfix换成%clearfix:

%clearfix {

\*zoom: 1;

&:after,

&:before {

content: "";

display: table;

}

&:after {

clear: both;

overflow: hidden;

}

}

并在nav ul中通过@extend调用已定义的%clearfix:

nav {

display: block;

ul {

margin: 50px auto;

width: 800px;

@include fit-content();

padding: 0;

list-style: none;

@extend %clearfix;*//调用%cleafix*

… *//省略后续代码*

}

}

这个时候编译出来的代码：

nav ul {

\*zoom: 1; }nav ul:after, nav ul:before {

content: "";

display: table; }nav ul:after {

clear: both;

overflow: hidden; }

这样，我们就可以把前面使用.button隐藏的地雷给拔了。我们只需要定义一个%button，并用@extend来调用：

.button,

%button {

display: block;

padding: 10px;

background: green;

}

.sidebar .signup .button {

margin-top: 20px;

}

.registrantion,

.remember-password {

.button {

margin-bottom: 33px;

}

}

.edit-account .delete-area .button {

background-color: red;

color: white;

}

.article a {

@extend %button;

}

这样编译出来的CSS，就是你想要的结果了：

.button,.article a {

display: block;

padding: 10px;

background: green; }

.sidebar .signup .button {

margin-top: 20px; }

.registrantion .button,.remember-password .button {

margin-bottom: 33px; }

.edit-account .delete-area .button {

background-color: red;

color: white; }

## **@include vs @extend**

对于初学SASS的同学来说，看到SASS编译出来的CSS都会感到困惑。不是说SASS能让编译出来的CSSS更简洁吗？为什么一到自己手中，反面编译出来的代码出现很多重复的呢？都想有什么方法能让代码避免生成重复的。

随后在SASS中产生了Mixins，我们可以将相似的样式定义成一个函数模块，然后通过@include来调用。但很多时候，我们又不需要这么强大的功能。这个时候出现@extend来调用定义好相同样式的类，可没想到，这个功能是方便了，但无形中为使用者埋下了一个地雷。为了解除这个隐患，在SASS3.2中增加了一个%placeholders功能。让大家能很方便定义一些功能简单的相同样式模块。

通过前面的介绍@mixin需要@include来调用，而.class和%placeholders需要@extend来调用，那么两者有何区别呢？

* @include主要是用来调用@mixin定义的函数模块。在@mixin中可以定义一个相似功能样式，而且可以设置变量、定义参数和默认参数值；
* @extend主要是用来调用.class或者%placeholders定义的属性模块；在.class或者%placeholders中可以定义一个相同样式，但这里面不能定义参数；
* @include每次调用相同的@mixin时，编译出来的CSS相同样式不会进行合并；
* @extend每次调用相同的 .class时，如果.class在样式出现多次，那么编译出来的CSS有可能不是您需要的样式；
* @extend每次调用相同的%placeholders时，编译出来的CSS相同样式会进行合并。

下面我们通过一个清除浮动的案例，分别看看@include和@extend之间的区别：

##### @include与@mixin使用例子

###### SCSS

@mixin clearfix{

& {

\*zoom: 1;

}

&:before,

&:after {

display: table;

content: "";

}

&:after {

clear: both;

overflow: hidden;

}

}

ul{

@include clearfix;

}

.block {

@include clearfix;

}

###### CSS

ul {

\*zoom: 1; }ul:before, ul:after {

display: table;

content: ""; }ul:after {

clear: both;

overflow: hidden; }

.block {

\*zoom: 1; }.block:before, .block:after {

display: table;

content: ""; }.block:after {

clear: both;

overflow: hidden; }

很明显，相同的样式不会进行合并。

##### @include和%placeholders使用例子

###### SCSS

%clearfix{

& {

\*zoom: 1;

}

&:before,

&:after {

display: table;

content: "";

}

&:after {

clear: both;

overflow: hidden;

}

}

ul{

@extend %clearfix;

}

.block {

@extend %clearfix;

}

###### CSS

ul, .block {

\*zoom: 1; }ul:before, .block:before, ul:after, .block:after {

display: table;

content: ""; }ul:after, .block:after {

clear: both;

overflow: hidden; }

很明显相同样式代码已经进行合并。

## **Mixins与%placeholders的结合**

Mixins如果使用不当，就会产生很多重复的代码，但仅用@extend很多时候又无法达到功能上的需求。那么有没有方法能把Mixins与%placeholders结合起来，取他们各自的优势呢？接下来，我们不仿一起探讨一下。

大家都知道,%placeholders就类似于CSS中的.classes或者#ids，只不过使用%代替了.和#。但%placeholders中的代码只有通过 @extend调用之后才会产生代码量，不然他是不会产生任何代码量。

下面我们来看一个Mixins与%placeholders结合在一起制作的一个网格系统。

%grid {

box-sizing: border-box;

display: inline-block;

padding-left: 1em;

padding-right: 1em;}

@mixin grid($width: 1){

@extend %grid;

width: percentage($width);

}

在grid()中通过@extend调用了%grid，不过他并没有产生任何代码，除非你像下面那样调用grid():

.grid-half {

@include grid(1 / 2);

}

.grid-third {

@include grid(1 / 3);

}

输出的CSS如下：

.grid-half, .grid-third {

box-sizing: border-box;

display: inline-block;

padding-left: 1em;

padding-right: 1em; }

.grid-half {

width: 50%; }

.grid-third {

width: 33.33333%; }

按照这样的方法，我们可以制作出一个简单的百分比网格系统：

$columns: 12;$gutter: 2em;

%grid {

box-sizing: border-box;

display: inline-block;

padding: {

left:$gutter / 2;

right:$gutter / 2;

}

}

@mixin grid($width: 1){

@extend %grid;

width: percentage($width);

}

@for $column from 1 through $columns {

.grid-*#{$column} {*

@include grid(1 / $column);

}

}

输出的网格系统代码如下：

.grid-1, .grid-2, .grid-3, .grid-4, .grid-5, .grid-6, .grid-7, .grid-8, .grid-9, .grid-10, .grid-11, .grid-12 {

box-sizing: border-box;

display: inline-block;

padding-left: 1em;

padding-right: 1em; }

.grid-1 {

width: 100%; }

.grid-2 {

width: 50%; }

.grid-3 {

width: 33.33333%; }

.grid-4 {

width: 25%; }

.grid-5 {

width: 20%; }

.grid-6 {

width: 16.66667%; }

.grid-7 {

width: 14.28571%; }

.grid-8 {

width: 12.5%; }

.grid-9 {

width: 11.11111%; }

.grid-10 {

width: 10%; }

.grid-11 {

width: 9.09091%; }

.grid-12 {

width: 8.33333%; }

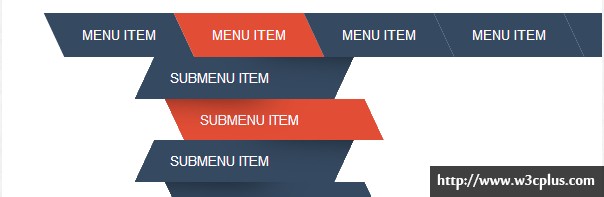
## **使用Mixins和继承的细节**

了解了@include定义的@mixin,@extend定义的.class和@extend定义的%placeholders差异之后，我们在写SASS时，有一些细节大家应该了解：

* 不要使用没有设置参数的@mixin，他们应该是.class或者%placeholders;
* 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;
* 不要使用太深的选择器嵌套。
* 如果你能避免，不要使用标签名。这不是一个taxative规则，但比id或者类名的性能要更低；
* 不要使用子选择器符号>，在SASS中很无效；
* 不要使用同史选择器+，配合你当前的标记他是非常无效。
* 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；

## **案例实战**

说了这么多，还没有实战过。光说不练假把式，下面我们就来做一个效果。使用SASS制作[Red-team](http://www.red-team-design.com/" \t "http://www.w3cplus.com/preprocessor/_blank)新发布的下拉菜单[效果](http://www.red-team-design.com/dropdown-menu-concept" \t "http://www.w3cplus.com/preprocessor/_blank)：

[](http://www.red-team-design.com/dropdown-menu-concept)

别的不多说，先上结构：

<ul class="menu">

<li><a href="">首页</a></li>

<li>

<a href="">博客</a>

<ul class="drop-menu">

<li><a href="">CSS3</a></li>

<li><a href="">SASS</a></li>

<li><a href="">JavaScript</a></li>

<li><a href="">jQuery</a></li>

</ul>

</li>

<li><a href="">案例</a></li>

<li><a href="">资源</a></li>

<li><a href="">前端收藏夹</a></li></ul>

有了结构，我们就要开始动手了，在动手之前对这个效果先简单的分析一下：

* ****定义变量****：我要定义几个变量，方便换成别的风格；
* ****清除浮动****：列表使用了浮动，需要清除浮动
* ****清除列表默认样式****：导航是使用ul制作，所以需要清除其默认样式
* ****定义transfrom****：效果中使用到了CSS3的transform，使用@mixin定义成一个模块
* ****定义transition****：下拉菜单出现的时候有一个transition效果
* ****定义fit-content****：使用CSS3的fit-content
* ****定义box-shadow****：使用CSS3的box-shadow
* ****定义文本****：设置菜单项文本效果

接下来，我们一个一个分析：

#### 1、定义变量

在定义变量中，主要定义了几个常用的变量，比如说文本色、背景色、悬浮的背景色、字号、字体等：

*//1.定义变量*$color: *#fff !default; //设置文本颜色*$bgColor: *#34495e !default;//设置背景色*$sfbgColor: *#e74c3c !default;//设置悬浮背景色* $fontSize: 14px !default;*//设置字号*$fontFamily: Arial, Helvetica !default;*//设置字体*$width: 462px !default; *//设置默认宽度*

#### 2、设置clearfix

因为菜项进行了浮动，需要在父导航上清除浮动，这里使用%clearfix创建了一个清除浮动的属性模块：

*//2.使用%placeholders定义清除浮动*

%clearfix {

&{

\*zoom: 1;

}

&:before,

&:after{

content: "";

display: table;

}

&:after {

clear: both;

overflow: hidden;

}

}

#### 3、清除列表的默认样式

为了让浏览器显示一致，先通过%listStyle定义一个重置列表属性的模块：

*//3.清除列表默认样式*

%listStyle {

margin: 0;

padding: 0;

list-style: none outside none;

}

#### 4、设置浏览器前缀

在样式中需要使用CSS3的部分属性，为了避免添加浏览器的私有属性，在这里，我们先设置一下：

$prefix-for-webkit: true !default;$prefix-for-mozilla: true !default;$prefix-for-microsoft: true !default;$prefix-for-opera: true !default;$prefix-for-spec: true !default;

//浏览器前缀@mixin prefixer ($property, $value, $prefixes) {

@each $prefix in $prefixes {

@if $prefix == webkit and $prefix-for-webkit == true {

-webkit-*#{$property}: $value;*

}

@else if $prefix == moz and $prefix-for-mozilla == true {

-moz-*#{$property}: $value;*

}

@else if $prefix == ms and $prefix-for-microsoft == true {

-ms-*#{$property}: $value;*

}

@else if $prefix == o and $prefix-for-opera == true {

-o-*#{$property}: $value;*

}

@else if $prefix == spec and $prefix-for-spec == true {

*#{$property}: $value;*

}

@else {

@warn "Unrecognized prefix: #{$prefix}";

}

}

}

#### 5、设置transform功能

在效果中我们有使用CSS3的transform功能，为了能方便使用，把这一块也提取出来：

*//5.定义transform*

*//示例: @include prefixer(border-radius, $radius, webkit spec);//Transform, transform-origin, transform-style//----------------------------------------*

@mixin transform($property...) {

@include prefixer(transform, $property, webkit moz o ms spec);

}

@mixin transform-origin($axes: 50%) {*// x-axis - left | center | right | length | %// y-axis - top | center | bottom | length | %// z-axis - length*

@include prefixer(transform-origin, $axes, webkit moz o ms spec);

}

@mixin skewX($degrees) {

@include prefixer(transform, skewX($degrees), webkit moz o ms spec);

-webkit-backface-visibility: hidden;

}

#### 6、设置过渡transition

为了让下接菜单出来的时候，动作平滑些，在效果中使用了transition:

*//6.定义transition*

*// Return vendor-prefixed property names if appropriate// Example: transition-property-names((transform, color, background), moz) -> -moz-transform, color, background//----------------------------------------*

@function transition-property-names($props, $vendor: false) {

$new-props: ();

@each $prop in $props {

$new-props: append($new-props, transition-property-name($prop, $vendor), comma);

}

@return $new-props;

}

@function transition-property-name($prop, $vendor: false) {

*// put other properties that need to be prefixed here aswell*

@if $vendor and $prop == transform {

@return unquote('-'+$vendor+'-'+$prop);

}

@else {

@return $prop;

}

}

*// transition//----------------------------------------*

@mixin transition ($properties...) {

@if length($properties) >= 1 {

@include prefixer(transition, $properties, webkit moz o ms spec);

}

@else {

$properties: all 0.15s ease-out 0;

@include prefixer(transition, $properties, webkit moz o ms spec);

}

}

#### 7、设置fit-content模块

这个CSS3属性，在这里使用@mixin来定义：

*//7.定义fit-content*@mixin fit-content {

width: -webkit-fit-content;

width: -moz-fit-content;

width: -o-fit-content;

width: -ms-fit-content;

width: fit-content;

}

#### 8、设置box-shadow

实现box-shadow比较简单：

*//8.设置box-shadow*

*// box-shadow*

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

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

}

#### 9、定义文本样式块

为了实现导航文本块样式，在需要时才调用，特意将其调用出来：

*//9.设置文本*

%typography {

color: $color;

text: {

decoration: none;

align: center;

}

font: {

family: $fontFamily;

size: $fontSize;

}

}

#### 10、完善导航菜单其他样式

前面1~9可以说都是为第10步所做的准备，那么现在，我们通过SCSS来完善整个导航效果：

.menu {

width: $width;

@extend %clearfix;//调用清除浮动

@extend %listStyle;//调用清除列表样式

@include fit-content;

margin: 50px auto;

}

.drop-menu {

@extend %listStyle;//调用清除列表样式

}

.menu > li {

background: $bgColor;

float: left;

position: relative;

@include skewX(25deg);

}

.menu a {

display: block;

@extend %typography;

}

.menu li:hover {

background: $sfbgColor;

}

.menu > li > a {

padding: 1em 2em;

@include skewX(-25deg);

}

/\*Dropdown menu\*/

.drop-menu {

position: absolute;

width: $width / 4;

left: 50%;

margin-left: -($width / 8);

opacity: 0;

visibility: hidden;

@include skewX(-25deg);

@include transform-origin(left top);

li {

background-color: $bgColor;

position: relative;

overflow: hidden;

opacity: 0;

visibility: hidden;

@include transition(all .2s ease );

a {

padding: 1em 2em;

}

&::after {

content: "";

position: absolute;

top: -125%;

height: 100%;

width: 100%;

@include box-shadow(0 0 50px rgba(0,0,0,.9));

}

&:nth-child(odd) {

@include transform(skewX(-25deg) translateX(0));

a {

@include skewX(25deg);

}

&::after {

right: -50%;

@include transform(skewX(-25deg) rotate(3deg));

}

}

&:nth-child(even){

@include transform(skewX(25deg) translateX(0));

a {

@include skewX(-25deg);

}

&::after {

left: -50%;

@include transform(skewX(25deg) rotate(3deg));

}

}

}

}

.menu > li:hover .drop-menu,

.menu > li:hover .drop-menu li {

opacity: 1;

visibility: visible;

}

.menu > li:hover .drop-menu li:nth-child(even){

@include transform(skewX(25deg) translateX(15px));

}

.menu > li:hover .drop-menu li:nth-child(odd){

@include transform(skewX(-25deg) translateX(-15px));

}

到此，整个SASS制作导航菜单就算完成了，最后我们将SASS代码编译成CSS，并引入文件中：

@charset "UTF-8";*/\**

*\* 1.定义变量*

*\* 2.设置清除浮动*

*\* 3.清除列表默认样式*

*\* 4.定义transform*

*\* 5.定义transition*

*\* 6.定义fit-content*

*\* 7.定义box-shadow*

*\* 8.设置文本*

*\*/*.menu {

\*zoom: 1; }.menu:before, .menu:after {

content: "";

display: table; }.menu:after {

clear: both;

overflow: hidden; }

.menu, .drop-menu {

margin: 0;

padding: 0;

list-style: none outside none; }

.menu a {

color: white;

text-decoration: none;

text-align: center;

font-family: Arial, Helvetica;

font-size: 14px; }

.menu {

width: 462px;

width: -webkit-fit-content;

width: -moz-fit-content;

width: -o-fit-content;

width: -ms-fit-content;

width: fit-content;

margin: 50px auto; }

.menu > li {

background: #34495e;

float: left;

position: relative;

-webkit-transform: skewX(25deg);

-moz-transform: skewX(25deg);

-o-transform: skewX(25deg);

-ms-transform: skewX(25deg);

transform: skewX(25deg);

-webkit-backface-visibility: hidden; }

.menu a {

display: block; }

.menu li:hover {

background: #e74c3c; }

.menu > li > a {

padding: 1em 2em;

-webkit-transform: skewX(-25deg);

-moz-transform: skewX(-25deg);

-o-transform: skewX(-25deg);

-ms-transform: skewX(-25deg);

transform: skewX(-25deg);

-webkit-backface-visibility: hidden; }

*/\*Dropdown menu\*/*.drop-menu {

position: absolute;

width: 115.5px;

left: 50%;

margin-left: -57.75px;

opacity: 0;

visibility: hidden;

-webkit-transform: skewX(-25deg);

-moz-transform: skewX(-25deg);

-o-transform: skewX(-25deg);

-ms-transform: skewX(-25deg);

transform: skewX(-25deg);

-webkit-backface-visibility: hidden;

-webkit-transform-origin: left top;

-moz-transform-origin: left top;

-o-transform-origin: left top;

-ms-transform-origin: left top;

transform-origin: left top; }.drop-menu li {

background-color: #34495e;

position: relative;

overflow: hidden;

opacity: 0;

visibility: hidden;

-webkit-transition: all 0.2s ease;

-moz-transition: all 0.2s ease;

-o-transition: all 0.2s ease;

-ms-transition: all 0.2s ease;

transition: all 0.2s ease; }.drop-menu li a {

padding: 1em 2em; }.drop-menu li::after {

content: "";

position: absolute;

top: -125%;

height: 100%;

width: 100%;

-webkit-box-shadow: 0 0 50px rgba(0, 0, 0, 0.9);

box-shadow: 0 0 50px rgba(0, 0, 0, 0.9); }.drop-menu li:nth-child(odd) {

-webkit-transform: skewX(-25deg) translateX(0);

-moz-transform: skewX(-25deg) translateX(0);

-o-transform: skewX(-25deg) translateX(0);

-ms-transform: skewX(-25deg) translateX(0);

transform: skewX(-25deg) translateX(0); }.drop-menu li:nth-child(odd) a {

-webkit-transform: skewX(25deg);

-moz-transform: skewX(25deg);

-o-transform: skewX(25deg);

-ms-transform: skewX(25deg);

transform: skewX(25deg);

-webkit-backface-visibility: hidden; }.drop-menu li:nth-child(odd)::after {

right: -50%;

-webkit-transform: skewX(-25deg) rotate(3deg);

-moz-transform: skewX(-25deg) rotate(3deg);

-o-transform: skewX(-25deg) rotate(3deg);

-ms-transform: skewX(-25deg) rotate(3deg);

transform: skewX(-25deg) rotate(3deg); }.drop-menu li:nth-child(even) {

-webkit-transform: skewX(25deg) translateX(0);

-moz-transform: skewX(25deg) translateX(0);

-o-transform: skewX(25deg) translateX(0);

-ms-transform: skewX(25deg) translateX(0);

transform: skewX(25deg) translateX(0); }.drop-menu li:nth-child(even) a {

-webkit-transform: skewX(-25deg);

-moz-transform: skewX(-25deg);

-o-transform: skewX(-25deg);

-ms-transform: skewX(-25deg);

transform: skewX(-25deg);

-webkit-backface-visibility: hidden; }.drop-menu li:nth-child(even)::after {

left: -50%;

-webkit-transform: skewX(25deg) rotate(3deg);

-moz-transform: skewX(25deg) rotate(3deg);

-o-transform: skewX(25deg) rotate(3deg);

-ms-transform: skewX(25deg) rotate(3deg);

transform: skewX(25deg) rotate(3deg); }

.menu > li:hover .drop-menu,.menu > li:hover .drop-menu li {

opacity: 1;

visibility: visible; }

.menu > li:hover .drop-menu li:nth-child(even) {

-webkit-transform: skewX(25deg) translateX(15px);

-moz-transform: skewX(25deg) translateX(15px);

-o-transform: skewX(25deg) translateX(15px);

-ms-transform: skewX(25deg) translateX(15px);

transform: skewX(25deg) translateX(15px); }

.menu > li:hover .drop-menu li:nth-child(odd) {

-webkit-transform: skewX(-25deg) translateX(-15px);

-moz-transform: skewX(-25deg) translateX(-15px);

-o-transform: skewX(-25deg) translateX(-15px);

-ms-transform: skewX(-25deg) translateX(-15px);

transform: skewX(-25deg) translateX(-15px); }

最终效果如下面DEMO所示：

[](http://codepen.io/airen/pen/uCHda)

是不是很爽，为了验证一下，我们来调整几个变量，对其进行换肤：

$bgColor: *#3ca803 !default;//设置背景色*$sfbgColor: *#236300 !default;//设置悬浮背景色* $fontSize: 16px !default;*//设置字号*

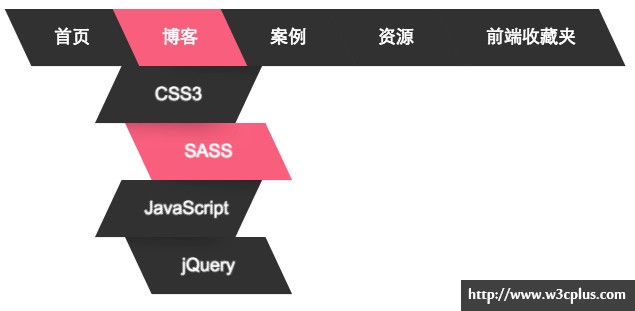
在上面的基础我们修改了三个变量，重新编译一下SCSS文件，这个时候效果就变成：

[](http://codepen.io/airen/pen/wEvxt)

你想怎么换就怎么换，比如说，我想把导航变大一些：

$bgColor: *#333333 !default;//设置背景色*$sfbgColor: *#ff5f7d !default;//设置悬浮背景色* $fontSize: 18px !default;*//设置字号*$width: 560px !default; *//设置默认宽度*

重新编译，刷新后，整个效果又变了：

[](http://codepen.io/airen/pen/mDxgo)

经过实点一翻是不是更有感觉了。当然这个实例还不是最完美的。其实我们还可以设置更多一点的变量，调整变量我们就可以修改整个风格，而不只是修改颜色大小这么简单。另外如果使用对颜色设置多色，通过条件判断来更换颜色，这样也会比目前这个完美。当然还有其他地方也可以完善。如果您对此感兴趣，你不仿动手修改一下。我在这里只是起一个抛砖引玉的效果，最终是要大家自己动手实战。

#### 特别声明：

本教程部分内容和代码引用于下面两篇文章：

* [SASS Placeholders Versus Mixins and Extends](http://miguelcamba.com/blog/2013/07/11/sass-placeholders-versus-mixins-and-extends/" \t "http://www.w3cplus.com/preprocessor/_blank)
* [Combine mixin directives with placeholder selectors](http://www.luyfel.nl/articles/mixins-and-placeholders" \t "http://www.w3cplus.com/preprocessor/_blank)

上面案例效果来自于：

* [Just another dropdown menu concept](http://www.red-team-design.com/dropdown-menu-concept" \t "http://www.w3cplus.com/preprocessor/_blank)

案例中CSS3部分SCSS代码来自于：

* [sassCore](https://github.com/marvin1023/sassCore" \t "http://www.w3cplus.com/preprocessor/_blank)中的[\_css3.scss](https://github.com/marvin1023/sassCore/blob/master/core/_css3.scss" \t "http://www.w3cplus.com/preprocessor/_blank)

## **总结**

本文从SASS的****嵌套****、****混合****、****继承****入手，介绍SASS中最具有特色的三大特性的使用方法。剖析了他们各自的优缺点。并且在此基础上扩展出@include、@extend和%placeholders三者的使用细节，以及相互依赖的关系。最后通过一个下拉导航菜单为例，向大家介绍了如何使用SASS的基本特性制作我们需要的效果。

最后希望这篇文章能给喜欢SASS的同学带来些许的帮助，如果您有更好的建议或者想法可以直接在下面的评论中留言。同时欢迎更多的同学使用SASS，能与大家共同探讨和学习SASS是人生中的一件乐事。(^\_^)