1、嵌套：

0） 父选择器：&

编译后会替换成父级的样式

1. 选择器嵌套：

假设我们有一段这样的结构：

<header>

<nav>

<a href=“##”>Home</a>

<a href=“##”>About</a>

<a href=“##”>Blog</a>

</nav>

<header>

想选中 header 中的 a 标签，在写 CSS 会这样写：

nav a {

color:red;

}

header nav a {

color:green;

}

那么在 Sass 中，就可以使用选择器的嵌套来实现：

nav {

a {

color: red;

header & {

color:green;

}

}

}

1. 属性嵌套：

CSS 有一些属性前缀相同，只是后缀不一样，比如：border-top/border-right，与这个类似的还有 margin、padding、font 等属性。假设你的样式中用到了：

.box {

border-top: 1px solid red;

border-bottom: 1px solid green;

}

在 Sass 中我们可以这样写：

.box {

border: {

top: 1px solid red;

bottom: 1px solid green;

}

}

1. 伪类嵌套：（注意：&和相连的类名之间不能有任何的空格，不然就会变成CSS选择器中的后代选择器）

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。我们就拿经典的“clearfix”为例吧：

.clearfix{

&:before,

&:after {

content:"";

display: table;

}

&:after {

clear:both;

overflow: hidden;

}

}

编译出来的 CSS：

clearfix:before, .clearfix:after {

content: "";

display: table;

}

.clearfix:after {

clear: both;

overflow: hidden;

}

2、继承：@extend

//SCSS

.btn {

border: 1px solid #ccc;

padding: 6px 10px;

font-size: 14px;

}

.btn-primary {

background-color: #f36;

color: #fff;

@extend .btn;

}

.btn-second {

background-color: orange;

color: #fff;

@extend .btn;

}

编译出来之后：

//CSS

.btn, .btn-primary, .btn-second {

border: 1px solid #ccc;

padding: 6px 10px;

font-size: 14px;

}

.btn-primary {

background-color: #f36;

color: #fff;

}

.btn-second {

background-clor: orange;

color: #fff;

}

从示例代码可以看出，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

3、混合宏 VS 继承 VS 占位符

a) Sass 中的混合宏使用：

总结：编译出来的 CSS 清晰告诉了大家，他不会自动合并相同的样式代码，如果在样式文件中调用同一个混合宏，会产生多个对应的样式代码，造成代码的冗余，这也是 CSSer 无法忍受的一件事情。不过他并不是一无事处，他可以传参数。

个人建议：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。

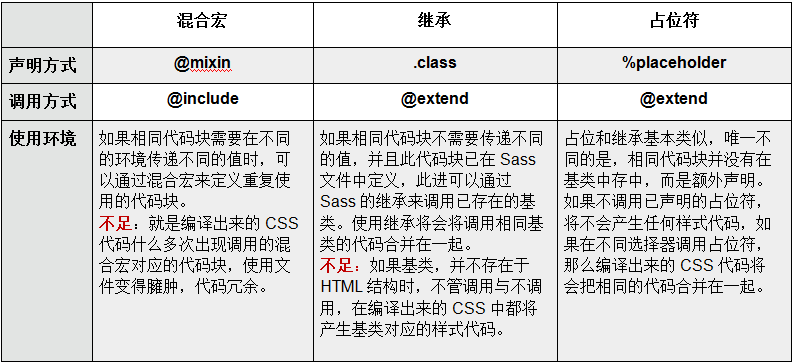
b) Sass 中继承：

总结：使用继承后，编译出来的 CSS 会将使用继承的代码块合并到一起，通过组合选择器的方式向大家展现。但是他不能传变量参数。

个人建议：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。

c) 占位符：

总结：编译出来的 CSS 代码和使用继承基本上是相同，只是不会在代码中生成占位符 mt 的选择器。那么占位符和继承的主要区别的，“占位符是独立定义，不调用的时候是不会在 CSS 中产生任何代码；继承是首先有一个基类存在，不管调用与不调用，基类的样式都将会出现在编译出来的 CSS 代码中。”



4、插值#{}

1. 例子：

$properties: (margin, padding);

@mixin set-value($side, $value) {

@each $prop in $properties {

#{$prop}-#{$side}: $value;

}

}

.login-box {

@include set-value(top, 14px);

}

编译后 CSS：

.login-box {

margin-top: 14px;

padding-top: 14px;

}

1. 构建一个选择器。可以这样使用：

@mixin generate-sizes($class, $small, $medium, $big) {

.#{$class}-small { font-size: $small; }

.#{$class}-medium { font-size: $medium; }

.#{$class}-big { font-size: $big; }

}

@include generate-sizes("header-text", 12px, 20px, 40px);

编译后 CSS：

.header-text-small { font-size: 12px; }

.header-text-medium { font-size: 20px; }

.header-text-big { font-size: 40px; }

1. 使用限制：
2. 可能会无属性值：

$margin-big: 40px;

$margin-medium: 20px;

$margin-small: 12px;

@mixin set-value($size) {

margin-top: $margin-#{$size};

}

.login-box {

@include set-value(big);

}

上面的 Sass 代码编译出来，你会得到下面的信息：error style.scss (Line 5: Undefined variable: “$margin-".)所以，#{}语法并不是随处可用

1. 不能在 mixin 中调用：

@mixin updated-status {

margin-top: 20px;

background: #F00;

}

$flag: "status";

.navigation {

@include updated-#{$flag};

}

上面的代码在编译成 CSS 时同样会报错：

error style.scss (Line 7: Invalid CSS after "...nclude updated-": expected "}", was "#{$flag};")

1. 可以在@extend 中使用插值。例如：

%updated-status {

margin-top: 20px;

background: #F00;

}

.selected-status {

font-weight: bold;

}

$flag: "status";

.navigation {

@extend %updated-#{$flag};

@extend .selected-#{$flag};

}

编译后CSS:

.navigation {

margin-top: 20px;

background: #F00;

}

.selected-status, .navigation {

font-weight: bold;

}

5、注释：

在compressed下，编译后只有/\*! \*/注释会保留；

在expanded下，编译后只有/\*! \*/ 、 /\* \*/注释会保留；

在nested下，编译后只有/\*! \*/ 、 /\* \*/注释会保留；

在compact下，编译后只有/\*! \*/ 、 /\* \*/注释会保留。

即单行注释永不输出。

6、除法：

”/ ”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

}

编译出来的CSS

p {

font: 10px/8px;

width: 500px;

height: 250px;

margin-left: 9px;

}

另，注意在减号运算时，减号两边都要加上空格，不然会识别为连接的字符串。

7、字符串相加+：

如果有引号的字符串被添加了一个没有引号的字符串 （也就是，带引号的字符串在 + 符号左侧）， 结果会是一个有引号的字符串。 同样的，如果一个没有引号的字符串被添加了一个有引号的字符串 （没有引号的字符串在 + 符号左侧）， 结果将是一个没有引号的字符串。 例如：

p:before {

content: "Foo " + Bar;

font-family: sans- + "serif";

}

编译出来的 CSS：

p:before {

content: "Foo Bar";

font-family: sans-serif;

}

8、@for：

在 Sass 的 @for 循环中有两种方式：

@for $i from <start> through <end>

@for $i 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。

through例子：

@for $i from 1 through 3 {

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

}

编译出来的 CSS:

.item-1 {

width: 2em;

}

.item-2 {

width: 4em;

}

.item-3 {

width: 6em;

}

to 关键字的例子：

@for $i from 1 to 3 {

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

}

编译出来的 CSS:

.item-1 {

width: 2em;

}

.item-2 {

width: 4em;

}

进阶用法：

//SCSS

$grid-prefix: span !default;

$grid-width: 60px !default;

$grid-gutter: 20px !default;

%grid {

  float: left;

  margin-left: $grid-gutter / 2;

  margin-right: $grid-gutter / 2;

}

@for $i from 1 through 5 {

  .#{$grid-prefix}#{$i}{

    width: $grid-width \* $i + $grid-gutter \* ($i - 1);

    @extend %grid;

  }

}

//CSS

.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;

}

9、字符串函数：

unquote( ) 函数：

只能删除字符串最前和最后的引号（双引号或单引号），而无法删除字符串中间的引号。如果字符没有带引号，返回的将是字符串本身。

quote()函数：

主要用来给字符串添加引号。如果字符串，自身带有引号会统一换成双引号 ""。字符串中间有单引号或者空格时，需要用单引号或双引号括起，否则编译的时候将会报错。

10、函数：

percentage()：将一个不带单位的数字转换成百分比形式；

round() ：将一个数四舍五入为一个最接近的整数，可带单位；

ceil() ：只做入，不做舍的计算（进1函数），可带单位；

floor()：舍一函数，可带单位；

abs()：绝对值，可带单位；

min(),max()：取最大/小值，可带单位；1%,1 分别用两函数比较，max返回1%，min返回1%；

random() ：随机得到的是0~1之间的数，相当于数学中的[0，1）；

length($list)：返回一个列表的长度值；

nth($list, $n)：返回一个列表中指定的某个标签值，从1开始；

join($list1, $list2, [$separator])：只能将两个列给连接在一起，变成一个列表，$separator取comma或space，默认为auto/space，很多时候不只碰到两个列表连接成一个列表，这个时候就需要将多个 join() 函数合并在一起使用：

>> join((blue red), join((#abc #def),(#dee #eff)))

(#0000ff #ff0000 #aabbcc #ddeeff #ddeeee #eeffff)；

append($list1, $val, [$separator])：将某个值放在列表的最后，

如果没有明确的指定 $separator 参数值，取comma或space，其默认值是 auto/space。

如果列表只有一个列表项时，那么插入进来的值将和原来的值会以空格的方式分隔。

如果列表中列表项是以空格分隔列表项，那么插入进来的列表项也将以空格分隔；

如果列表中列表项是以逗号分隔列表项，那么插入进来的列表项也将以逗号分隔。

zip($lists…)：将几个列表结合成一个多维的列表，

>> zip(1px 2px 3px,solid dashed dotted,green blue red)

((1px "solid" #008000), (2px "dashed" #0000ff), (3px "dotted" #ff0000))；

index($list, $value)：返回一个值在列表中的位置值，返回值从1开始。

11、Introspection 函数包括了几个判断型函数：

type-of($value)：返回一个值的类型：

返回值：

number 为数值型。

string 为字符串型，包括有/无引号。

bool 为布尔型。

color 为颜色型。；

unit($number)：返回一个值的单位；

unitless($number)：判断一个值是否带有单位；

comparable($number-1, $number-2)：判断两个值是否可以做加、减和合并。

12、Miscellaneous 函数：

if($condition,$if-true,$if-false) 同其他语言的 xxx ? ture : false；

13、Map：

数据结构：

$map: (

$key1: value1,

$key2: value2,

$key3: value3

)

函数：

map-get($map,$key)：根据给定的 key 值，返回 map 中相关的值，如果 $key 不存在 $map中，将返回 null 值，且表现在编译后的css文件上，不存在选择器，就像没编译一样，可以使用下面的函数进行判断，最后可以封装一个函数

@function colors($color){

$names: map-keys($social-colors);

@if not index($names,$color){

@warn "Waring: `#{$color} is not a valid color name.`";

}

@return map-get($social-colors,$color);

} 帮助判断

；

map-merge($map1,$map2)：将两个 map 合并成一个新的 map，如果 $map1 和 $map2 中有相同的 $key 名，那么将 $map2 中的 $key 会取代 $map1 中的值。

map-remove($map,$key)：从 map 中删除一个 key，返回一个新 map。

map-keys($map)：返回 map 中所有的 key，传入的map不能有嵌套map，具体应用

@each $name in map-keys($social-colors){

.btn-#{$name}{

color: colors($name);

}

} 。

map-values($map)：返回 map 中所有的 value，传入的map不能有嵌套map。

map-has-key($map,$key)：根据给定的 key 值判断 map 是否有对应的 value 值，如果有返回 true，否则返回 false。

keywords($args)：返回一个函数的参数，这个参数可以动态的设置 key 和 value，

可以通过混合宏或函数的参数变创建 map。参数也是成对出现，其中 $args 变成 key(会自动去掉$符号)，而 $args 对应的值就是value。

@mixin map($args...){

@debug keywords($args);

}

@include map(

$dribble: #ea4c89,

$facebook: #3b5998,

$github: #171515,

$google: #db4437,

$twitter: #55acee

);

在命令终端可以看到一个输入的 @debug 信息：

DEBUG: (dribble: #ea4c89, facebook: #3b5998, github: #171515, google: #db4437, twitter: #55acee)

14、颜色函数：

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

rgba($red,$green,$blue,$alpha)/ rgba($color,$alpha)：根据红、绿、蓝和透明度值创建一个颜色，后面一种更常用；

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-2,[$weight])：把两种颜色混合在一起，$color-1 和 $color-2可以是任何表达式，也可以是颜色变量，$weight 为 合并的比例（选择权重），默认值为 50%，其取值范围是 0~1 之间。它是每个 RGB 的百分比来衡量，当然透明度也会有一定的权重。默认的比例是 50%，这意味着两个颜色各占一半，如果指定的比例是 25%，这意味着第一个颜色所占比例为 25%，第二个颜色所占比例为75%。

15、HSL函数简介：

hsl($hue,$saturation,$lightness)：通过色相（hue）、饱和度(saturation)和亮度（lightness）的值创建一个颜色；

hsla($hue,$saturation,$lightness,$alpha)：通过色相（hue）、饱和度(saturation)、亮度（lightness）和透明（alpha）的值创建一个颜色；

hue($color)：从一个颜色中获取色相（hue）值；

saturation($color)：从一个颜色中获取饱和度（saturation）值；

lightness($color)：从一个颜色中获取亮度（lightness）值；

adjust-hue($color,$degrees)：通过改变一个颜色的色相值，创建一个新的颜色，-360deg 至 360deg 之间，也可以是百分数；

lighten($color,$amount)：通过改变颜色的亮度值，让颜色变亮，创建一个新的颜色；

darken($color,$amount)：通过改变颜色的亮度值，让颜色变暗，创建一个新的颜色；

saturate($color,$amount)：通过改变颜色的饱和度值，让颜色更饱和，从而创建一个新的颜色

desaturate($color,$amount)：通过改变颜色的饱和度值，让颜色更少的饱和，从而创建出一个新的颜色；

grayscale($color)：将一个颜色变成灰色，相当于desaturate($color,100%);

complement($color)：返回一个补充色，相当于adjust-hue($color,180deg);

invert($color)：反回一个反相色，红、绿、蓝色值倒过来，而透明度不变。

16、Opacity函数：

alpha($color) /opacity($color)：获取颜色透明度值；

rgba($color, $alpha)：改变颜色的透明度值；

opacify($color, $amount) / fade-in($color, $amount)：使颜色更不透明，其取值范围主要是在 0~1 之间；

transparentize($color, $amount) / fade-out($color, $amount)：使颜色更加透明。

17、@import ：根据文件名引入。 默认情况下，它会寻找 Sass 文件并直接引入，在少数几种情况下，它会被编译成 CSS 的 @import 规则：

如果文件的扩展名是 .css。

如果文件名以 http:// 开头。

如果文件名是 url()。

如果 @import 包含了任何媒体查询（media queries）。

如果上述情况都没有出现，并且扩展名是 .scss 或 .sass， 该名称的 Sass 或 SCSS 文件就会被引入。如果没有扩展名，Sass 将试着找出具有 .scss 或 .sass 扩展名的同名文件并将其引入。例如：

@import "foo.scss"; 或 @import "foo";

两者都将引入 foo.scss 文件，

而

@import "foo.css";

@import "foo" screen;

@import "http://foo.com/bar";

@import url(foo);

将被编译为：

@import "foo.css";

@import "foo" screen;

@import "http://foo.com/bar";

@import url(foo);

也可以通过一个 @import 引入多个文件。例如：

@import "rounded-corners", "text-shadow";

将引入 rounded-corners 和 text-shadow 两个文件。

引入SCSS 或 Sass 文件，在文件名前面加一个下划线，就能避免被编译。然后，就可以像往常一样引入这个文件了，而且还可以省略掉文件名前面的下划线。

例如，你有一个文件叫做 \_colors.scss。 这样就不会生成 \_colors.css 文件了， 而且你还可以这样做：

@import "colors";//不用加下划线

来引入 \_colors.scss 文件。

注意，在同一个目录不能同时存在带下划线和不带下划线的同名文件。 例如， \_colors.scss 不能与 colors.scss 并存。

嵌套 @import

虽然大部分时间只需在顶层文件使用 @import 就行了， 但是，你还可以把他们包含在 CSS 规则 和 @media 规则中。

示例：

假设要引入的样式文件`example.scss`文件中包含这样的代码：

.example {

color: red;

}

然后这样引用：

#main {

@import "example";

}

编译出来的 CSS：

#main .example {

color: red;

}

18、@media：和CSS 的使用规则一样的简单，但它有另外一个功能，可以嵌套在 CSS 规则中。有点类似 JS 的冒泡功能一样，如果在样式中使用 @media 指令，它将冒泡到外面。示例：

.sidebar {

width: 300px;

@media screen and (orientation: landscape) {

width: 500px;

}

}

编译出来：

.sidebar {

width: 300px;

}

@media screen and (orientation: landscape) {

.sidebar {

width: 500px;

}

}

@media 也可以嵌套 @media：

@media screen {

.sidebar {

@media (orientation: landscape) {

width: 500px;

}

}

}

此时编译出来：

@media screen and (orientation: landscape) {

.sidebar {

width: 500px;

}

}

19、@extend除继承外的另一种用法：

%placeholder 不使用@extend显示调用是不会生成任何样式代码。那么在选择器中使用占位符一样。比如下面的代码：

#context a%extreme {

color: blue;

font-weight: bold;

font-size: 2em;

}

这段代码在不调用之前不产生任何代码，只有能过@extend调用之后才生成代码：

.notice {

@extend %extreme;

}

编译出来的CSS

#context a.notice {

color: blue;

font-weight: bold;

font-size: 2em;

}

20、@error 和 @warn、@debug 功能是如出一辙，抛出自定义错误信息。