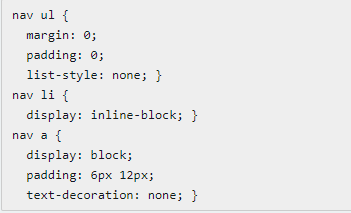
Sass不同样式风格的输出方法

1. 嵌套输出方式 nested
2. 展开输出方式 expanded
3. 紧凑输出方式 compact
4. 压缩输出方式 compressed

只有紧凑输出方式 compact 没有ed。

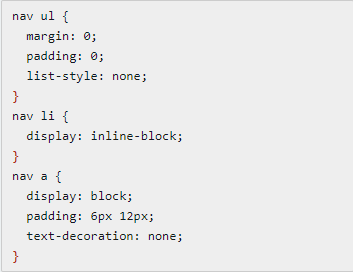
sass --watch test.scss:test.css --style nested

嵌套输出方式 nested:

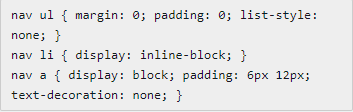


扩展输出方式 expanded:

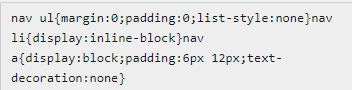
sass --watch test.scss:test.css --style expanded



紧凑输出方式 compact：



压缩输出方式 compressed:



在编辑器中都挤在一行。

声明变量：

$width: 100px !default; $color: #fff !default;

其中!default是指默认值。

Sass 中变量的作用域在过去几年已经发生了一些改变。直到最近，规则集和其他范围内声明变量的作用域才默认为本地。如果已经存在同名的全局变量，从 3.4 版本开始，Sass 已经可以正确处理作用域的概念，并通过创建一个新的局部变量来代替。

$color: orange !default; // 定义全局变量

em{

$color: red; // 定义局部变量

a{

Color: $color; //调用局部变量red

}

}

span{

Color:$color //调用全局变量orange

}

我们没有必要声明一个从不会变更的变量！ 最好是对那些被频繁使用或者会被动态更新的内容声明变量。

我们应该尽可能避免使用选择器嵌套。

混合宏，可以带参数，内部包含需要复用的代码段。也可以包含复杂的逻辑。

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

@if length($shadow) >= 1{ // 如果$shadow的参数数量大于等一1.

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

} @else {

$shadow: 0 0 4px rgba(0,0,0,.3);

@include prefixer(box-shadow,$shadow)

}

}

然后使用@include mixinname(); 来调用混合宏。

混合宏可以带参数

@mixin border-radius($radius:3px){

-webkit-border-radius: $radius;

border-radius: $radius;

}

如果有很多地方都是圆角，而且都是3px的圆角，就完全可以不带入参数。

@include border-radius;

Tip: box-shadow并没有想象的那么简单：

/\* x偏移量 | y偏移量 | 阴影颜色 \*/box-shadow: 60px -16px teal;

/\* x偏移量 | y偏移量 | 阴影模糊半径 | 阴影颜色 \*/box-shadow: 10px 5px 5px black;

/\* x偏移量 | y偏移量 | 阴影模糊半径 | 阴影扩散半径 | 阴影颜色 \*/box-shadow: 2px 2px 2px 1px rgba(0, 0, 0, 0.2);

/\* 插页(阴影向内) | x偏移量 | y偏移量 | 阴影颜色 \*/box-shadow: inset 5em 1em gold;

/\* 任意数量的阴影，以逗号分隔 \*/box-shadow: 3px 3px red, -1em 0 0.4em olive;

/\* 全局关键字 \*/box-shadow: inherit;box-shadow: initial;box-shadow: unset;

第四个数字参数是决定的阴影大小，第三个参数是模糊距离。

混合宏的最大缺点是，不能智能的合并代码

@mixin border-radius{

-webkit-border-radius: 3px;

border-radius: 3px;

}

.box {

@include border-radius;

margin-bottom: 5px;

}

.btn {

@include border-radius;

}

.box {

-webkit-border-radius: 3px;

border-radius: 3px;

margin-bottom: 5px;

}

.btn {

-webkit-border-radius: 3px;

border-radius: 3px;

}

继承 @extend 就不谈了

占位符 %placeholder ，多了就不谈了，好处是如果不使用@extend继承占位符的样式，就不会有任何代码产生。但是缺点也很明显，就是不能带参数。而且编译出来的代码会将相同的代码合并在一起。

//SCSS%mt5 {

margin-top: 5px;

}%pt5{

padding-top: 5px;

}

.btn {

@extend %mt5;

@extend %pt5;

}

.block {

@extend %mt5;

span {

@extend %pt5;

}

}

//CSS

.btn, .block {

margin-top: 5px;

}

.btn, .block span {

padding-top: 5px;

}

