Sass (.scss)

sass-lang

Sass拥有许多perks(过滤器)，使我们能够编写简洁的，可读的代码。在这一课中，我们将探讨三个概念：

- Variables
- Mixins
- Nests

Nesting

选择器
属性：

```
.banner {
    font-family: 'Pacifico', cursive;
    height: 400px;
    background-image: url("lemonade.jpg");
    border : {
        top: $standard-border;
        bottom: $standard-border;
    }
    .slogan {
        position: absolute;
        border: $standard-border;
        top: 200px;
        left: 25%;
        width: 50%;
        height: 200px;
        background-color: $translucent-white;
        span {
            position: absolute;
            text-align: center;
            line-height: 200px;
        }
    }
}
```

Variables

> 在SCSS变量让你自己选择的标识符分配给特定的值。
> 可以分配给一个变量在CSS中不同的数据类型。
> 颜色, 数字, 文本字符串, 布尔值, null(空值), 列表(list), 映射(map)
> 1.列表可以通过空格或逗号分隔。
> 注意：您也可以用括号包围一个列表，并创建由列表组成的列表
> 2.映射与列表非常相似，但是每一个对象是一个键 - 值对。

> 注意：在映射中，一个键的值可以是列表或另一个映射。

```
$translucent-white: rgba(255,255,255,0.3);
$icon-square-length: 300px;
$standard-border: 4px solid black;
.div{
    font-family: 'Pacifico', cursive;
    height: 400px;
    background-image: url("lemonade.jpg");
    border : {
        top: $standard-border;
        bottom: $standard-border;
    }
}
```

```
1.5em Helvetica bold;
/* or */
Helvetica, Arial, sans-serif;

(key1: value1, key2: value2);
```

伪类(:hover), 伪元素(::before，::after)

&

```
.notecard{
    &:hover{
        @include transform (rotatey(-180deg));
    }
}
```

Mixins （@mixin， @include）

> 在Sass中，mixin让你做CSS声明组,你想要在整个网站中重用的。
> 注意：Mixin的名字和所有其他Sass标识使用的连字符和下划线可以互换？。

```
@mixin backface-visibility {
    backface-visibility: hidden;
    -webkit-backface-visibility: hidden;
    -moz-backface-visibility: hidden;
    -ms-backface-visibility: hidden;
    -o-backface-visibility: hidden;
}
/* usage */
.notecard {
    .front, .back {
        width: 100%;
        height: 100%;
        position: absolute;
        @include backface_visibility;
        /* backface-visibility === backface_visibility ??? */
    }
}
```

Mixins 也有接收一个参数的能力.

```
/* backface-visibility: visible|hidden|initial|inherit; */
$visibility: hidden;

@mixin backface-visibility($visibility) {
    backface-visibility: $visibility;
    -webkit-backface-visibility: $visibility;
    -moz-backface-visibility: $visibility;
    -ms-backface-visibility: $visibility;
    -o-backface-visibility: $visibility;
}
/* 为其参数,传递一个值的语法如下： */
@include backface-visibility(hidden);
```

Mixin 参数可以被分配一个默认值,在**Mixin**定义，通过使用一个特殊的符号。

> 当Mixin被包括,如果没有值传递,缺省值被分配给参数。

```
@mixin backface-visibility($visibility: hidden) {
    backface-visibility: $visibility;
    -webkit-backface-visibility: $visibility;
    -moz-backface-visibility: $visibility;
    -ms-backface-visibility: $visibility;
    -o-backface-visibility: $visibility;
}
```

一般来说，以下是有关参数和混入**5**个重要的事实：

> 1. Mixins可以采取多个参数。
> 2. Sass允许你明确你的@include声明定义每个参数

> 3. 当明确指定的值，你可以给他们发送乱序.
> 4. 如果一个Mixin定义参数的组合,没有默认值，你应该首先定义那些没有默认值的。
> 5. Mixins 可以嵌套

```
@mixin dashed-border($width, $color: #FFF) {
    /*two argument only one has default value*/
    border: {
        color: $color;
        width: $width;
        style: dashed;
    }
}
span { //only passes non-default argument
    @include dashed-border(3px);
}
p { //passes both arguments
    @include dashed-border(3px, green);
}
div { //passes out of order but explicitly defined
    @include dashed-border(color: purple, width: 5px);
}
```

Sass允许你传递多个参数, 以列表或映射格式。

```
/* list */
@mixin stripes($direction, $width-percent, $stripe-color, $stripe-background: #FFF) {
    background: repeating-linear-gradient(
        $direction,
        $stripe-background,
        $stripe-background ($width-percent - 1),
        $stripe-color 1%,
        $stripe-background $width-percent
    );
}
/* map */
$college-ruled-style: (
    direction: to bottom,
    width-percent: 15%,
    stripe-color: blue,
    stripe-background: white
);
/* 然后,我们可以传递这些参数,在一个映射中,用跟随的...符号: */
.definition {
    width: 100%;
    height: 100%;
    @include stripes($college-ruled-style...);
 }
```

优先考虑可读性

> 注意：记住总是优先考虑可读性,, 在写更少的代码之上。
> 如果你发现它增加了你代码库的清晰度,这种方法是唯一有用。

字符串插值

> 在 Sass，字符串插值是放置一个字符串变量在其它两个字符串的中间的过程

```
@mixin photo-content($file) {
    content: url(#{$file}.jpg); //string interpolation
    object-fit: cover;
}
//....
.photo {
    @include photo-content('titanosaur');
    width: 60%;
    margin: 0px auto;
}
```

Sass 允许mixins内部使用 & 选择器。

> 1. & 选择器获取在mixin被包括的点,被分配的父的值。
> 2. 如果没有父选择器，那么该值为null且Sass将抛出一个错误。

```
@mixin text-hover($color){
    &:hover {
        color: $color;
    }
}
/* uasge */
.word { //SCSS:
    display: block;
    text-align: center;
    position: relative;
    top: 40%;
    @include text-hover(red);
}
```

> 他们具有能力，接收参数，给这些参数分配默认值，并接收上述的任何的格式参数，是最可读，方便你使得mixin Sass's最受欢迎的指令。

> &选择器 *是一个 Sass结构，它允许更灵活的表现通过引用父选择器,,when用CSS伪元素和类时。

> 字符串插值是胶水，可以让你在另一个字符串中间插入一个字符串，当它是在一个可变格式。
> 它的应用有所不同，但插值能力特别有用，用于传递文件名。

functions, arithmetic, and color operations

https://github.com/xgqfrms-GitHub/Sass/tree/master/Tutorial

在Sass 功能和操作可实现对样式的计算和迭代。

> 操作颜色值
> 在列表和映射上迭代
> 根据条件应用样式
> 赋值, 数学运算的结果值

opacity 不透明度

(alpha parameter) RGBA , HSLA

> transparent 透明

```
/* fade-out */
$color: rgba(39, 39, 39, 0.5);
$amount: 0.1;
$color2: fade-out($color, $amount);//rgba(39, 39, 39, 0.4)
/* fade-in */
$color: rgba(55,7,56, 0.5);
$amount: 0.1;
$color2: fade-in($color, $amount); //rgba(55,7,56, 0.6)

/* adjust-hue($color, $degrees) */
// 通过采取颜色和若干度（通常为-360度和360度之间）改变颜色的色调，并且通过量旋转色轮。（通常-360度和360度之间），和旋转由该量的色轮。
adjust-hue($color, $degrees)
```

执行数学函数, 计算测量-包括颜色。

> 计算的颜色

1. 该操作是在红，绿，和蓝分量执行的。
2. 它通过操作每两个数字,计算答案。
   Sass算术也可以计算HSLA和字符串的颜色，如红色和蓝色。

```
/* 分段计算 */
$color: #010203 + #040506;

color: red + blue;
```

Sass 算术运算

> 加 + 减 - 乘 * 除 / 模%(是在给定的除法的余数, “9%2”将是“1”)
> SCSS算术要求的单位是兼容; 例如，你无法通过ems乘像素(pixels)。

```
/* Error: 就像常规数学，两个单位相乘的结果平方单位(px^2)*/
10px * 10px = 100px * px
/* 为了获得100像素 */
10px * 10
```

```
.math {
    width: $width;
    text-align: center;
    background-color: $lagoon-blue;
    color: red + blue;
    height: $width/6;
    line-height: $width/6;
    border-radius: $width*1/30
}
```

在**CSS /**字符可以被用作分隔符。在**Sass**，该字符也被用于除。

> 下面是 / 字符用于除, 具体的实例 ：
>
> 1. 如果该值，或它的任何部分，被存储在一个变量或由一个函数返回。
> 2. 如果该值是由括号包围，除非这些括号在列表外面，值是在里边。
> 3. 如果该值被用作另一个算术表达式的一部分。

```
width: $variable/6; //division
line-height: (600px)/9; //division
margin-left: 20-10 px/ 2; //division
font-size: 10px/8px; //not division

font-size: $width/6/2;
```

Each loop (Each 迭代循环)

```
$list: (orange, purple, teal);
//Add your each-loop here
@each $item in $list {
    .#{$item} {
        background: $item;
    }
}
```

For loops (For 循环)

> 1. $i 只是一个变量, 列表中的索引，或列表中元素的位置
> 2. ***begin***和 end是循环的起点和终点的占位符
> 3. through 和 to关键字,在Sass是互换

```
@for $i from $begin through $end {
     //some rules and or conditions
}

/* 颜色轮 */
$total: 10; //Number of .ray divs in our html
$step: 360deg / $total; //Used to compute the hue based on color-wheel
@for $i from 1 through $total {
    .ray:nth-child(#{$i}) {
        background: adjust-hue(blue, $i * $step);
    }
}
.ray {
    height: 30px;
}
```

if

```
width: if( $condition, $value-if-true, $value-if-false);
/* https://www.codecademy.com/courses/learn-sass/ */
@mixin deck($suit) {
    @if($suit == hearts || $suit == spades){
        color: blue;
    }
    @else-if($suit == clovers || $suit == diamonds){
        color: red;
    }
    @else{
        //some rule
    }
}
```

对于越来越多的 **Sass**代码库的最佳实践。

```
http://sass-lang.com/guide
```

我们将开始最佳实践,组织文件。

@import

Partials(部分的)

> 在 Sass中,Partials 是你代码库中分开组织的特定功能的文件。

他们在文件名中使用一个"_"前缀符号,告诉**Sass** 推迟编译单独的文件，而是将其导入。

```
_filename.scss
# 要导入此部分到主文件,或该文件封装了重要的规则和大部分项目的样式文件-忽略下划线.
```

```
/* 例如，要导入名为_variables.scss文件，添加以下代码行 */
@import "variables";
//全局文件导入所有组件和集中逻辑。
@import "helper/variables";
@import url(https://fonts.googleapis.com/css?family=Pacifico);
//CSS import
@import "helper/placeholders";
@import "helper/mixins";
// helper 提醒者,是指象变量，mixins，函数等资源.
//这些元素是“帮助”弥补你的代码库的基础，with它的增长。
```

@extend

```
.lemonade {
    border: 1px yellow;
    background-color: #fdd;
}
.strawberry {
    @extend .lemonade;
    border-color: pink;
}
```

占位符

> 其行为就像一个类或ID选择器，但是使用 % 符号,而不是 # 或 . 符号

> 占位符防止自己规则被渲染到自己的CSS,
> 只有,被任何地方可以被扩展ID或类,一旦可以被扩展变得活跃

```
 a%drink {
    font-size: 2em;
    background-color: $lemon-yellow;
 }
 .lemonade {
    @extend %drink;
    //more rules
 }
 /* usage */
 a.lemonade {
    font-size: 2em;
    background-color: $lemon-yellow;
 }
.lemonade {
    //more rules
}
```

> 占位符巩固规则的好方法, 这实际上在HTML中,从未获得对自己使用。

```
// helper/_placeholders.scss:
%absolute{
    position: absolute;
}
// main.scss
@extend %absolute;
```

我们可以清楚地看到扩展方式结果,更清洁，更高效的输出结果,用越少可能的重复越好。

```
@mixin no-variable {
    font-size: 12px;
    color: #FFF;
    opacity: .9;
}
%placeholder {
    font-size: 12px;
    color: #FFF;
    opacity: .9;
}
span {
    @extend %placeholder;
}
div {
    @extend %placeholder;
}
p {
    @include no-variable;
}
h1 {
    @include no-variable;
}
```

> 将编译：

```
span, div{
    font-size: 12px;
    color: #FFF;
    opacity: .9;
}
p {
    font-size: 12px;
    color: #FFF;
    opacity: .9;
    /*rules specific to ps*/
}
h1 {
    font-size: 12px;
    color: #FFF;
    opacity: .9;
    /*rules specific to ps*/
}
```

> 1. 尽量只创建是采取一个参数入mixins，否则你应该扩展。
> 2. 总是看你的CSS的输出，以确保按预期的 extend.

read more ...

Sass (Syntactically Awesome StyleSheets)