2018-07-27 SASS使用笔记

一、为什么要使用sass

Sass 是一个 CSS 的扩展，它在 CSS 语法的基础上，允许您使用变量 (variables)，嵌套规则 (nested rules)，混合 (mixins)，导入 (inline imports) 等功能。sass 比less诞生的早，只是因为它是 Ruby 写的，因此受众面比较少。但是前端现在有 webpack 和 gulp 这样的构建工具之后，我们就不需要下载编译器或者是使用命令行了。（或者是借用 kaola 这样的工具）；需要注意的是，sass 可以使用 .sass 为后缀名，也可以使用 .scss 后缀名，前者比较恶心的是，语法像 python 一样没有花括号，也不能写分号，因此选择sass；

二、sass 的安装和使用

```
node npm 下安装
npm install node-sass sass-loader --save-dev
```

一、sass 的语法

```
1 . 注释
    @charset 'utf-8';                          //必须设置这个才能编译有中文的注释
    //单行注释                                 //单行注释不会被转译出来
    /*********/多行注释
2 . 导入
    @import '../static/sass/reset.scss'      //@import 是它的关键字，但是做了一些处理，如果导入的后缀名是 .scss .sass 或者是不写，那么它就会对文件进行编译，如果后缀名是 .css 就不会被编译
3 . 变量的定义
    $fontColor      :    #fff;
    $bgColor        :   #f5f6f7;
    $fnClass          :    top !default;
    .Class{
        border-#{$fnClass} : 1px solid #ddd;
        color : $fontColor;
        background : $bgColor;
    }
    $ 符是 sass 定义变量的关键字 ， #{} 拼接变量
 4 . 嵌套
    嵌套是 sass 语法里面非常常用的一个，嵌套的好处是可以有效的预防 css 类名重复，导致覆盖之前的样式，缺点是不宜嵌套过多层，这样会导致编译之后的代码体积变大
 5.继承
    设计一个页面时常遇到这样的问题，当一个样式类（class）包含另一个类的所有样式，并且它有自己的特定的样式，处理这种最常见的方法是在 html里面同时使用一个通用样式类的类名，不幸的是，这意味着我们必须时刻记住使用公用类名的时候还需要搭配使用特定样式的类名，这对于维护来说，是一个负担； @extend 指令可以避免这些问题
    eg : .error{
        border : 1px solid red;
        background : red;
```

```
}
.seriseError{
    @extend .error;
    border-width : 3px;
}
```

二、Sass 数组

1. 一维数组

```
    $array : 5px 10px 20px 30px;
```

2. 二维数组

```
    $array : 5px 10px , 20px 30px;
    $array : (5px 10px) (20px 30px);
```

3. 键值对数组

```
    $array : (
        key : val,
        key : val
    )
eg : $array : 5px 10px 15px 20px;
    a{
        color : nth($array, 1);
        &:hover{
            color : nth($array, 2);
        }
    }
```

注 ： 这里的数组下标是从 1 开始，包括后面的循环操作，下标都是从 1 开始

4. 遍历数组 ： @each $key , $val in $array{}

```
eg : $array : (
    1  :  #ff00d2,
    2  :  #f00f00,
    3  :  #1d5235
)
@each $key,$val in $array{
    .header-#{$key}{
        background : $val;
    }
}
```

5.合并数组 ： join($list , $list2 , $separator:auto)

```
join(10px 20px , 30px 40px)                                  // 10px 20px 30px 40px;
join((blue , red) , (#abc , #def))                           // blue, red, #abc, #def;
join(10px , 20px )                                           // 10px 20px
join(10px , 20px , coma)                                     // 10px , 20px
```

6.取得目标在数组的位置 ： index($list , value) // 下标以1开始

```
index(1px solid red , solid)                                 // 2
index((width : 10px , height : 20px),(height : 20px))        // 2
```

```
7.取得数组的长度 ： length($list)
    length(10px)                                                    // 1
    length(10px 20px 30px)                                          // 3
    length((width : 10px) , (height : 20px))                        // 2
    --> 往数组里面插值 ： append($list , $val , $separator:auto)
    append(10px 20px , 30px)                                        // 10px 20px 30px
    append((blue, red) , green)                                     // blue, red, green
    append(10px 20px , 30px 40px)                                   // 10px 20px (30px 40px)
    append(10px, 20px, comma)                                       // 10px, 20px
    append((blue , red) , green , space)                            // blue red green
```

三、Sass 对象

Sass 的对象与 js 的对象很相似，唯一不同的是，它是用小括号括起来，因为花括号用作各种嵌套边界。同时，为了操作sass对象，它提供了比JS丰富多的函数，它们基本是以map-开头（全部小写并且用“-”连起来是纯正的ruby风格）。

```
eg : (blue : #fof , red : #ff0 , yellow : #f00);
1.获取对象的某一个属性的值                                          // map-get
    map-get(('foo' : 1 , 'bar' : 2) , 'foo')                        // 1
2.删掉某一个键值对                                                  // map-remove($map , $key)
    map-remove(('foo' : 1 , 'bar' : 2) , 'foo')                     // ('bar' : 1)
3.取得所有对象的属性名，以数组形式返回                              // map-keys($map)
    map-keys(('foo' : 1 , 'bar' : 2))                               // 'foo' , 'bar'
4.取得所有对象的值，以数组形式返回                                  // map-values($map)
    map-values(('foo' : 1 , 'bar' : 2))                             // 1 , 2
5.判断它是否拥有某一个属性                                          // map-has-key($map , $key)
    map-has-key(('foo' : 1 , 'bar' : 2) , 'foo')                    // true
6.合并两个对象                                                      // map-merge($map1 , $map2)
    map-merge(('foo' : 1) , ('bar' : 2))                            // ('foo' : 1 , 'bar' : 2)
```

四、流程控制

```
@if  @else  @for  @each  @while
1.@if
    $boolean : true !default;
    @mixin simple-mixin {
        @if $boolean{
            display : block;
        } @else {
            display : none;
        }
    }
    .Class-Select{
        @include simple-mixin;
    }
```

2.三元运算符

```
    @if $width != auto {
        $width : if (unitless($width) , $width + px , $width);
    }
```

3.@for @for 循环有两种方式，@for xxx form yyy through zzz 或 @for xxx form yyy to zzz ，需要指定开始值和结束值，它们应该都是数字；

```
    $name : color !default;
    @for $i from 1 through 4 {                              // 相当于 js 的 for(var i = 1; i <= 4 ; i ++){}
        .#{name}-#{$i}{
            width : 60px + $i;
        }
    }
```

    @for 循环指令除了可以从小数值到大数值循环外，还可以从大数值到小数值循环，两种形式都支持

```
    @for $i from 5 through 1{}                              // 相当于 js 的 for(var $e = 5; $e >= 1; $e--){}
```

4.@each @each 是遍历数组与对象的，如果是遍历数组，@each 后面跟着的是元素的变量名，随便起，接着是in，最后是数组名

```
    $list : adam john box content main;
    @mixin author-images{
        @each $author in $list {
            .photo-#{$author}{
                background: url("/images/avatars/#{$author}.png") no-repeat;
            }
        }
    }
```

    @each 循环二维数组

```
    $animal-data : (puma, black, default),(sea-slug, blue, pointer),(egret, white, move);
    @each $animal , $color , $cursor in $animal-data{
        .#{$animal}-icon{
            background-image: url('/images/#{$animal}.png');
            border: 2px solid $color;
            cursor: $cursor;
        }
    }
```

    @each 遍历对象，后面跟两个变量，分别是键名与键值，逗号隔开，接着是 in ，最后是对象名

```
    @each $key , $val in (h : 2em , m : 5em , n : 1.2em){
        .#{$key}-icon{
            font-size : $val;
        }
    }
```

5.@while @while 与 js 的while非常相似

```
    $types : 4;
    $type-width : 20px;
    @while $type > 0 {
        .while-#{types}{
```

```
            width : $type-width + $types;
        }
        $types : $types -1;
    }
```

五、混合样式 (@mixin)

通常用来定义在整个样式表中重复使用的样式，而避免了无语义的类；大大减少了代码的重复率；

```
eg : @mixin clearfix{
        display : inline-block;
        & : after{
            content : '';
            display : block;
            height : 0;
            clear : both;
            visibility : hidden;
        }
    }
    .class-select{
        @include clearfix;
    }
```

@mixin 支持参数和默认参数

```
eg : @mixin border($bc : #000 , $side : all , $br : 0 , $bs : solid){
         position: relative;
         &: after{
            position: absolute;
            content: "";
            top: 0;
            left: 0;
            box-sizing: border-box;
            width: 100%;
            height: 100%;
            @if $side==all {
                border: 1px $bs $bc;
            }
            @else {
                $size: length($side);
                @for $i from 1 through $size {
                    border-#{nth($side, $i)}: 1px $bs $bc;
                }
            }
         }
    }
    .class-select{
        @include   border(#ccc , top);
```

```
    }
```

六、函数指令（@function）

Sass 支持自定义函数，并且能在任何值或者是脚本上下文中使用，函数可以访问任何全局内定义的变量，以及接受参数，就像一个混入（mixin），函数可以包含语句，并且必须调用 @return 来设置函数的返回值；

```
eg : $gridwidth : 40px;
    @function grid-width($n){
        @return $n * $gridwidth + ($n - 1);
    }
    .class-select{
        width : grid-width(5);
    }
```

七、Sass 的扩展和自定义 Sass 函数

对于独特的用户需求，Sass 为用户提供了多项高级的定制功能，使用这些功能需要对 Ruby 有深刻的理解；