## **前言：**

为了提高开发效率，前去学习sass。学完后发现非常简单，官网上前文太多，学习起来十分不效率。因此出一片这样文章，给那些想要把sass快速上手的人。

ps:阅读本文章前首先需要熟悉css          ps2：第一次写文章可能不算太好请见谅，文章末尾有介绍编译sass的编译器

**变量的引入**

用$符号声明变量

$color-red:red

div{color:$color-red}

编译后：

div{color:red;}

其中$color-red==$color\_red

在sass中 不区分 下划线和中划线

**css嵌套规则**

此处借用sass官网的代码

原css代码：

#content article h1 { color: #333 }

#content article p { margin-bottom: 1.4em }

#content aside { background-color: #EEE }

利用sass嵌套可以这么写：

#content {

        article {

            h1{color:#333}

            P{margin-bottom:1.4em}

        }

    aside{background-color:#EEE}

}

应对类似:hover的伪类时，如：

#content a{color:red}

#content a:hover{color:blue}

用sass写

#content a{

        color:red;

        &:hover{color:blue}

}

如果不是很清楚结构的话，再来一个例子

sass代码：

#content{

     a  {   color: red;

                &:hover{color: blue}

         }

    &:hover{ color: #000; }

}

编译后的css：

#content a {color: red; }

 #content a:hover { color: blue; }

#content:hover { color: #000; }

理解：&符号指向当前父级选择器，改变上文sass代码

#content{

  a{ color: red;

    body &:hover{color: blue}

   }

   body &:hover{        color: #000;    }

}

然后编译

#content a {color: red; }

 body #content a:hover { color: blue; }

body #content:hover { color: #000; }

**群组选择器嵌套**

sass能够减少重复的代码量，如下运用官网的例子，css：

.container h1, .container h2, .container h3 { margin-bottom: .8em }

Sass：

.container{

h1,h2,h3{margin-bottom:.8em}

}

一下子减去许多负担，有或者，sass：

nav, aside {

   a {color: blue}

    p {color: red;}

}

编译后的css：

nav a, aside a {  color: blue; }

nav p, aside p {  color: red; }

**子组合选择器和同层组合选择器：>、+和~**

此处用官网的例子：

article {

  ~ article { border-top: 1px dashed #ccc }

  > section { background: #eee }

  dl > {

    dt { color: #333 }

    dd { color: #555 }

  }

  nav + & { margin-top: 0 }

}

编译后：

article ~ article { border-top: 1px dashed #ccc }

article > footer { background: #eee }

article dl > dt { color: #333 }

article dl > dd { color: #555 }

nav + article { margin-top: 0 }

不禁让人想到了&符号

&指向父级，然而在 >+~选择器的前面加上&并不影响使用，但是面对伪类选择器&符号必须有，至于为什么，可以自己去尝试在伪类 : 符号连接的代码随便哪一边多出一个space，试试效果就明白了

**嵌套属性**

利用官网的例子

Sass：

nav {

  border: {

  style: solid;

  width: 1px;

  color: #ccc;

  }

}

编译后的Css：

nav {

  border-style: solid;

  border-width: 1px;

  border-color: #ccc;

}

例子二：

Sass：

nav {

      border: 1px solid #ccc {

          left: 0px;

          right: 0px;

  }

}

编译后的css：

nav {

  border: 1px solid #ccc;

  border-left: 0px;

  border-right: 0px;

}

**导入sass文件**

@import “sidebar”

可以省去默认的sass和scss的文件后缀

如果要导入原生css，请将css的后缀改为scss

**默认变量值**

$font-size-lg:40px !default;

如果$font-size-lg变量值没有重新声明，就使用默认值40px

**混合器（个人理解为可移动复制的代码块）**

使用@mixin做标识，并用@include引用，官方例子如下，sass：

@mixin rounded-corners {

  -moz-border-radius: 5px;

  -webkit-border-radius: 5px;

  border-radius: 5px;

}

notice {

  background-color: green;

  border: 2px solid #00aa00;

  @include rounded-corners;

}

编译成css：

.notice {

  background-color: green;

  border: 2px solid #00aa00;

  -moz-border-radius: 5px;

  -webkit-border-radius: 5px;

  border-radius: 5px;

}

当然混合器中还能使用css，如：

@mixin no-bullets {

  list-style: none;

  li {

    list-style-image: none;

    list-style-type: none;

    margin-left: 0px;

  }

}

使用时注意父级选择器

#### **给混合器传参**

@mixin tag-color($a-color,$b-color:$a-color,$c-color:$a-color){

color:$a-color;

background-color:$b-color;

border-color:$c-color;

}

div{

@include tag-color(red);或者@include tag-color(red,blue,yellow);

}

编译后的css：

div {

color:red;

background-color:red;

border-color:red; }

#### **选择器继承就不讲解了，不是那么容易理解的，需要的请去官网第六节查看[https://www.sass.hk/guide/](https://www.sass.hk/guide/" \t "https://www.jianshu.com/writer" \l "/notebooks/23191847/notes/_blank)**

最后介绍一下编辑器，在做好sass环境设置的前提下  
第一种：可以使用webstorm，这里贴上如何设置的[帖子](http://blog.csdn.net/ernijie/article/details/51672346" \t "https://www.jianshu.com/writer" \l "/notebooks/23191847/notes/_blank)  
第二种：使用webpack，添加sass环境，最后打包编译（webpack很好用，但是学习起来有一定的难度这里就不做详细介绍）  
第三种