

2_4

2_4	1
1. Sass预处理器	2
1.1. 上节回顾	2
1.1.1. Bootstrap 警告框、列表组、面版	2
1.1.2. Bootstrap 面包屑、输入框组、缩略图	2
1.1.3. Glyphicons 字体图标的使用	2
1.1.4. Bootstrap插件button、Bootstrap插件-标签页	2
1.2. 学习目标	2
1.2.1. sass的介绍及用法	2
1.2.2. sass中级用法——重用	2
1.2.3. 高级用法——循环语句、条件语句	2
1.2.4. 高级用法——循环语句、自定义函数	2
1.3. 课程内容	2
1.3.1. sass的介绍及用法	2
1.3.2. sass中级用法——重用	10
1.3.3. 高级用法——循环语句、条件语句	16
1.3.4. 高级用法——循环语句、自定义函数	19
1.4. 课后作业	21
1.4.1. 作业1	21
1.4.2. 作业2	21
1.4.3. 作业3	21
1.4.4. 作业4	22
1.5. 当天小结	22
1.5.1. sass的介绍及用法	22
1.5.2. sass中级用法——重用	22
1.5.3. 高级用法——循环语句、条件语句	22
1.5.4. 高级用法——循环语句、自定义函数	22

1. Sass预处理器

1.1. 上节回顾

1.1.1. Bootstrap 警告框、列表组、面版

1.1.2. Bootstrap 面包屑、输入框组、缩略图

1.1.3. Glyphicons 字体图标的使用

1.1.4. Bootstrap插件button、Bootstrap插件-标签页

1.2. 学习目标

1.2.1. sass的介绍及用法

1.2.2. sass中级用法——重用

1.2.3. 高级用法——循环语句、条件语句

1.2.4. 高级用法——循环语句、自定义函数

1.3. 课程内容

1.3.1. sass的介绍及用法

1

什么是Sass

Sass是一种CSS预处理语言，提供了许多便利的写法，大大节省了设计者的时间，使得CSS的开发，变得简单和可维护。

当你使用Sass这门语言时，你使用编译程序来转换Sass文件，通常以SCSS文件格式书写然后转换成CSS文件。Sass通过添加方便的函

数, 变量以及其他类似脚本的助手使CSS能更加快速得书写和更加简单的控制。

SASS文件就是普通的文本文件, 里面可以直接使用CSS语法。文件后缀名是.scss, 意思为Sassy CSS。

LESS与Sass的区别

LESS和Sass的主要不同就是他们的实现方式, LESS是基于JavaScript, 所以, 是在客户端处理的。

Sass是基于Ruby的, 然后是在服务器端处理的。

很多开发者不会选择LESS因为JavaScript引擎需要额外的时间来处理代码然后输出修改过的CSS到浏览器。

SCSS与 Sass 有什么差别

Sass

原先使用的缩排, 对于网页设计师来说相当不直观。而且实务上也不能直接将旧有的 CSS 直接贴进 Sass 中。因此 Sass 进行了进化, 改良了 syntax, 所以到了Sass 3 后来就被称为 SCSS (Sassy CSS)。

它的 syntax 与 CSS 原有的 syntax 完全 compatible, 使用了 { } 去取代原先的缩排方式。

并且

Bootstrap4将使用的Sass。

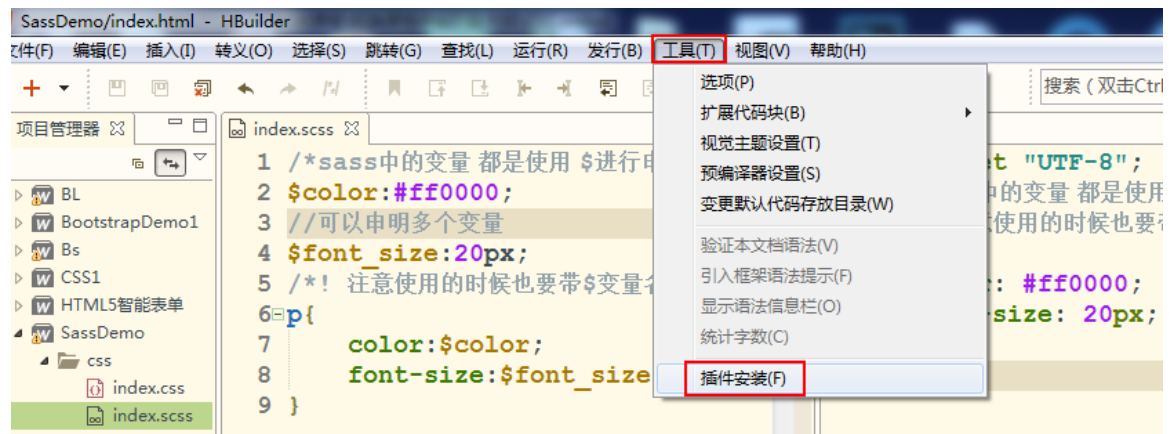
安装配置

安装方式有多种：

1, 使用Ruby方式

SASS是Ruby语言写的，但是两者的语法没有关系。不懂Ruby，照样使用。只是必须先安装Ruby，然后再安装SASS。

2, 使用HBuilder(7.2版本才有)工具 自带的编译工具





Sass的四个编译风格

nested：嵌套缩进的css代码，它是默认值

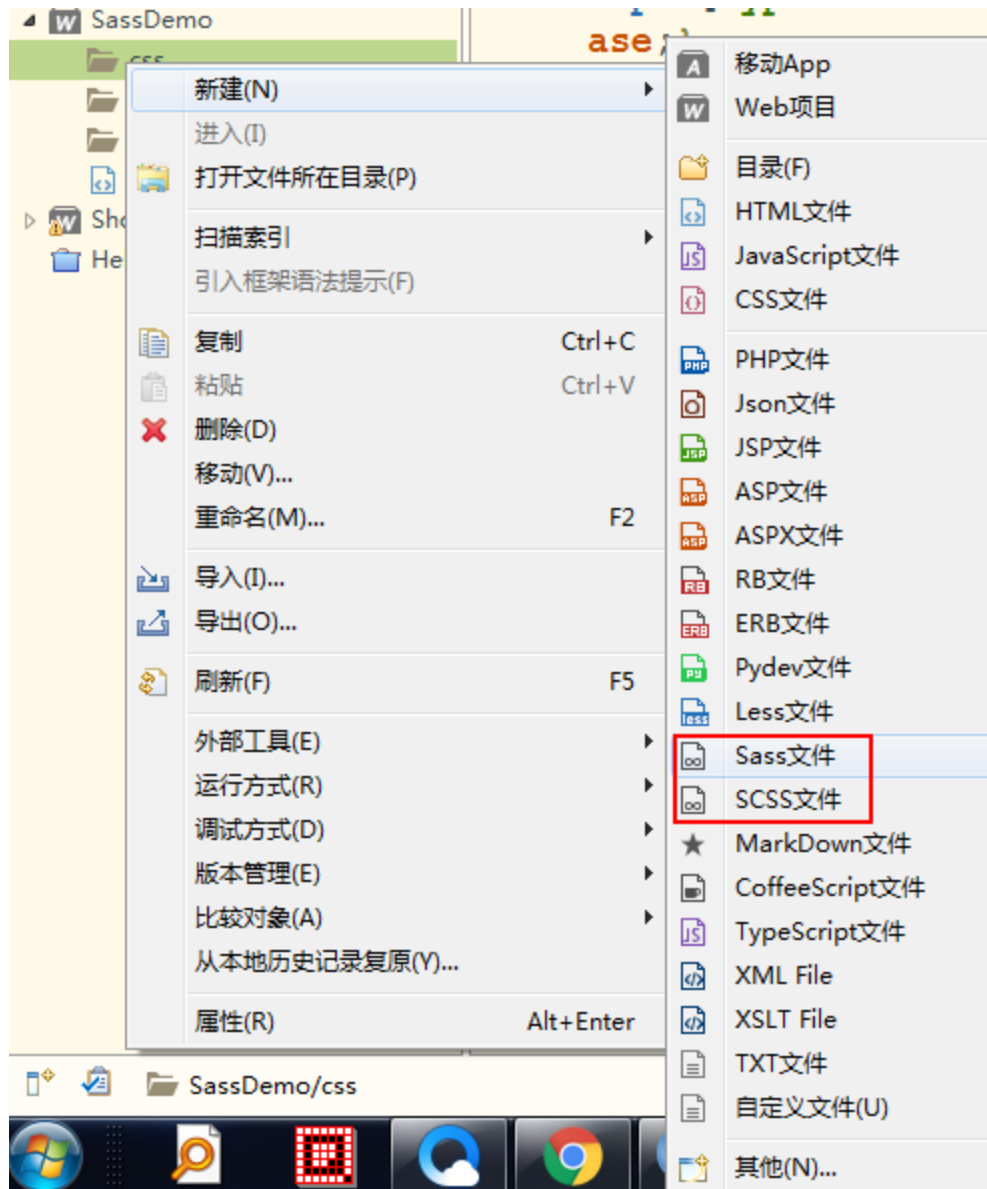
expanded：没有缩进的、扩展的css代码

compact：简洁格式的css代码

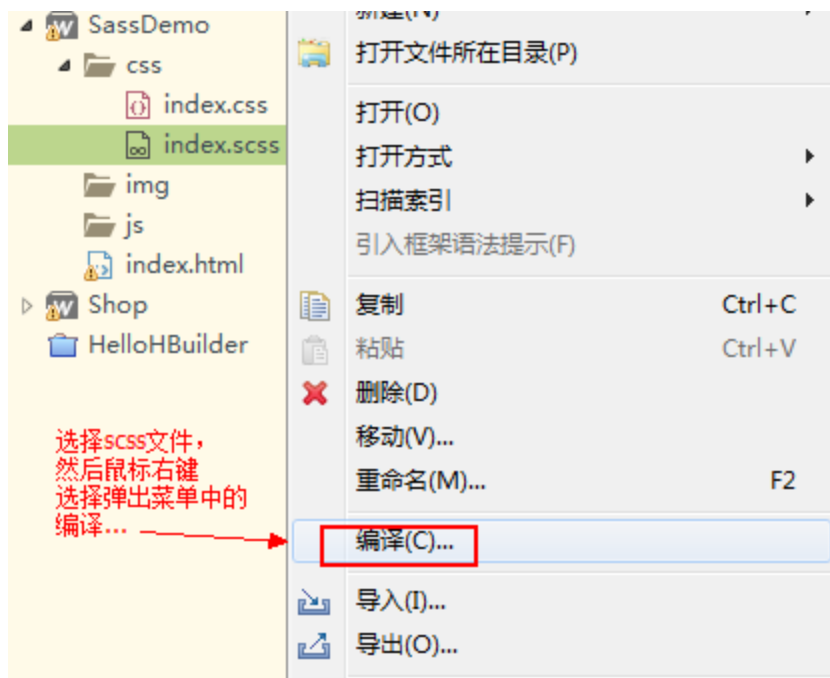
compressed : 压缩后的css代码（生产环境用）

基础用法

1, 新建(建议新建scss后缀的文件)

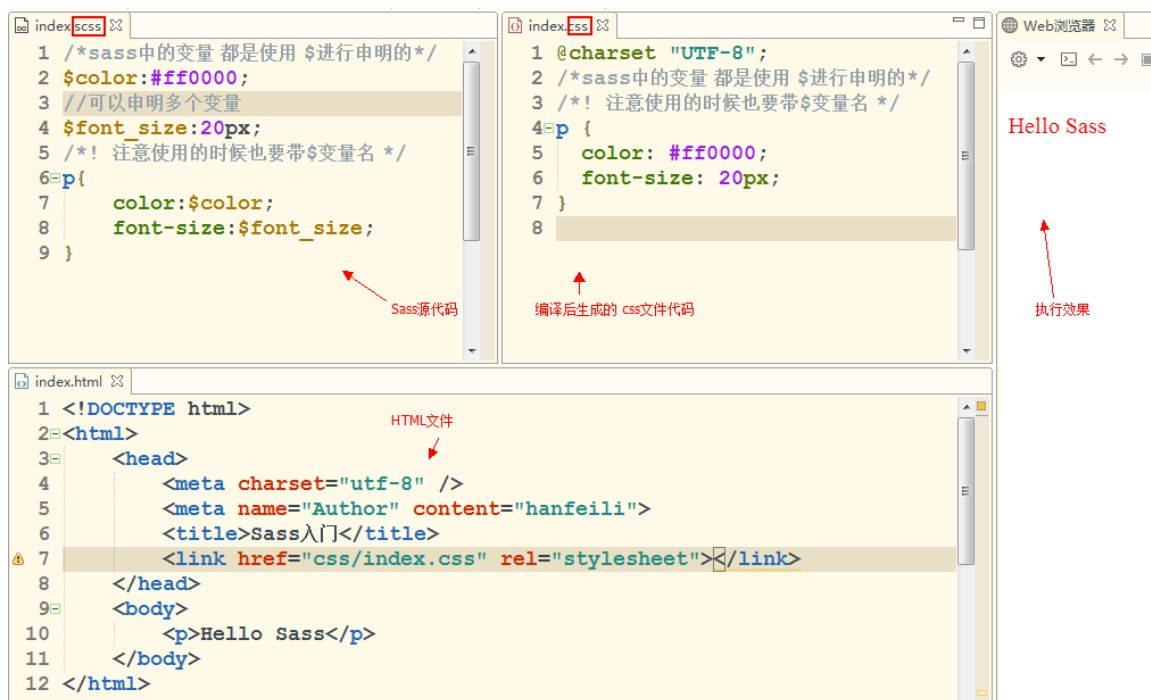


2, 写好了scss之后编译生成css文件。



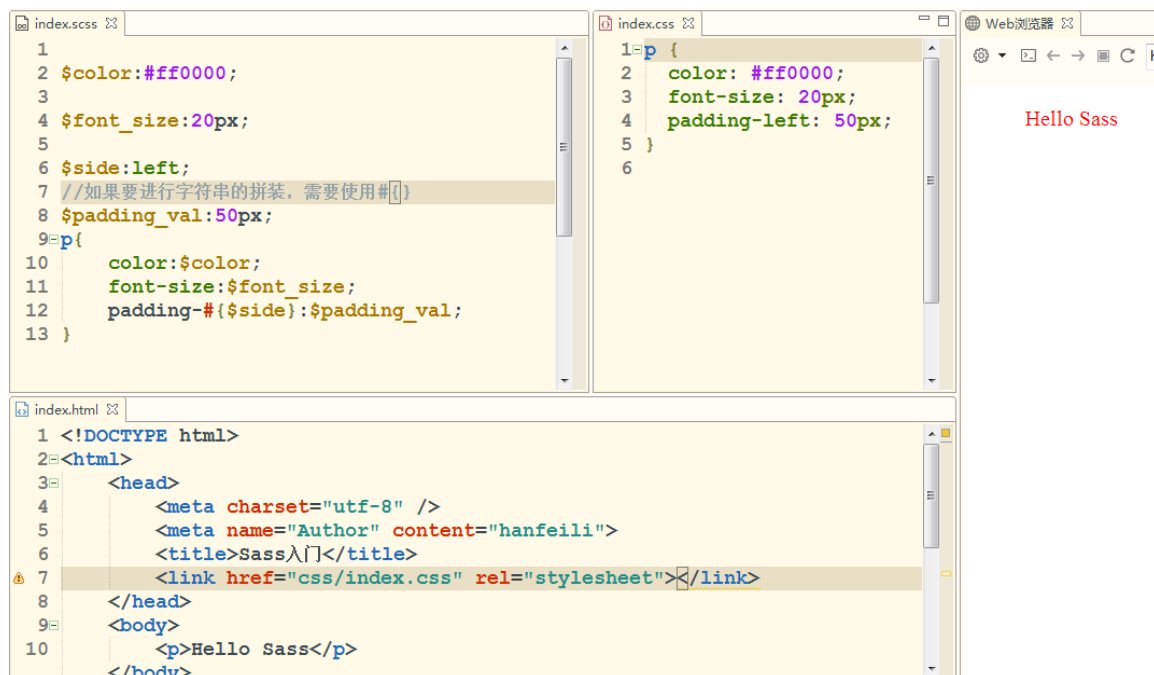
变量

SASS允许使用变量，所有变量以\$开头。



字符串拼装的使用：

/*如果变量需要镶嵌在字符串之中,就必须需要写在#{ }之中*/



注释

SASS共有两种注释风格。

标准的CSS注释,会保留到编译后的文件。

/* 内容 */

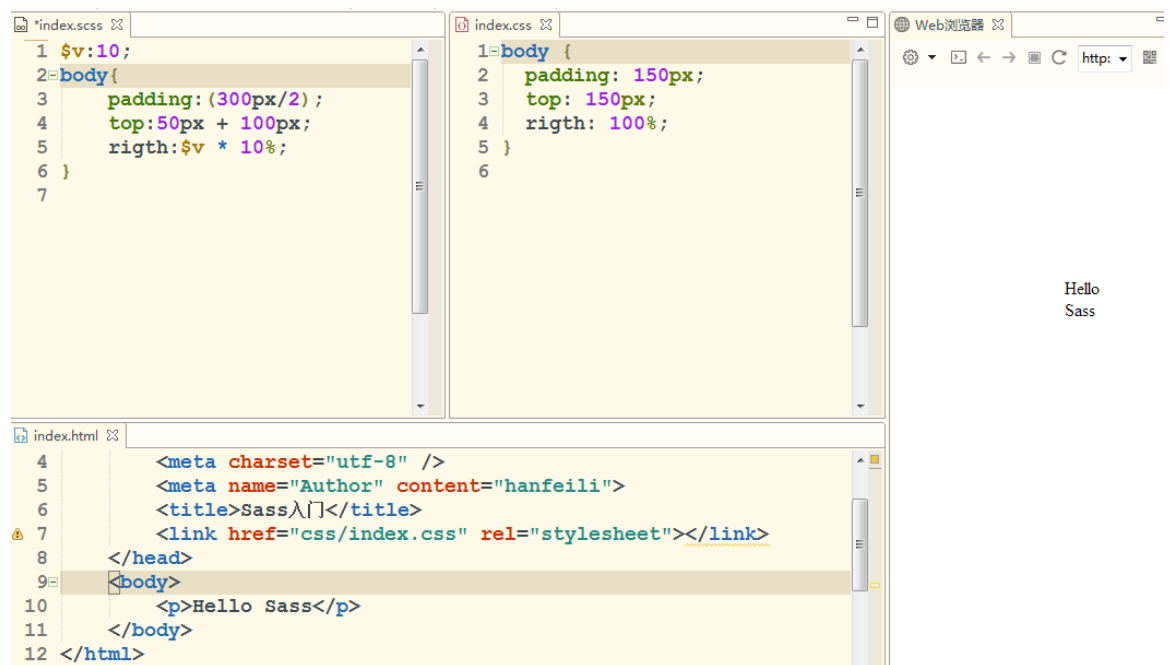
单行注释,只保留在SASS源文件中,编译后被省略。

// 内容

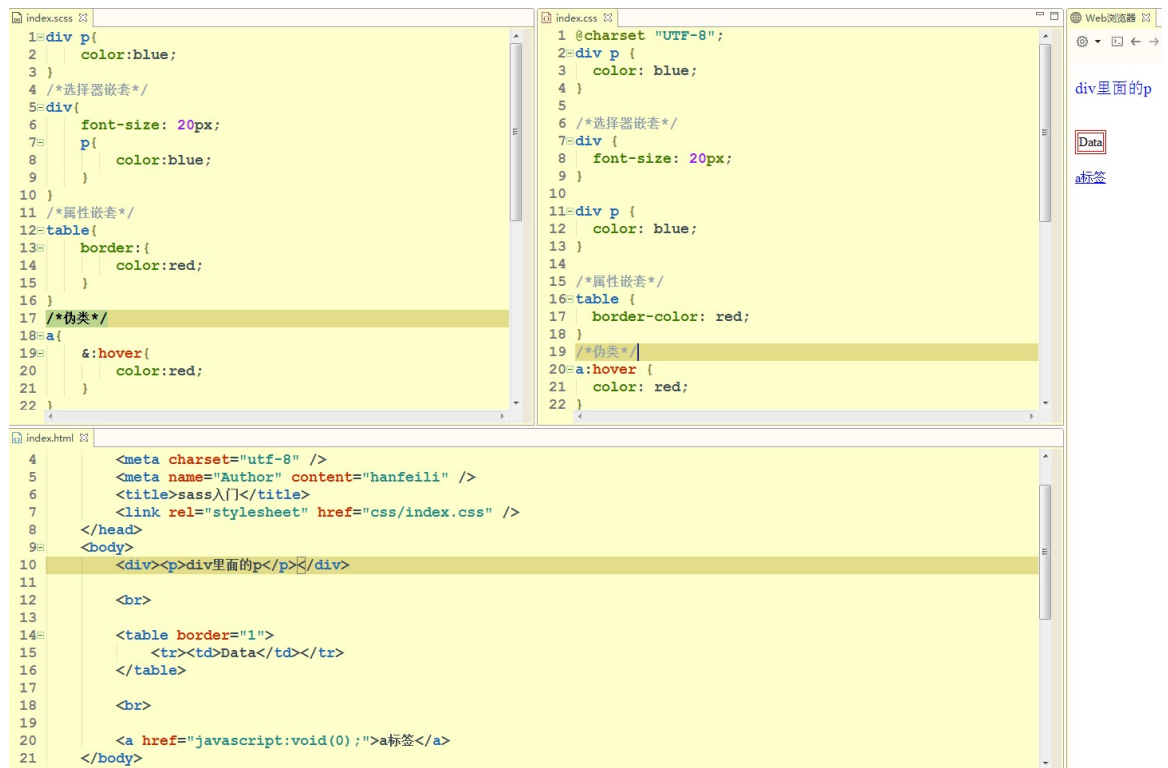
在/*后面加一个感叹号,表示这是"重要注释"。即使是压缩模式编译,也会保留这行注释,通常可以用于声明版权信息。

```
/*! 重要注释 */
```

计算功能



嵌套



1.3.2. sass中级用法——重用

2

@extend继承

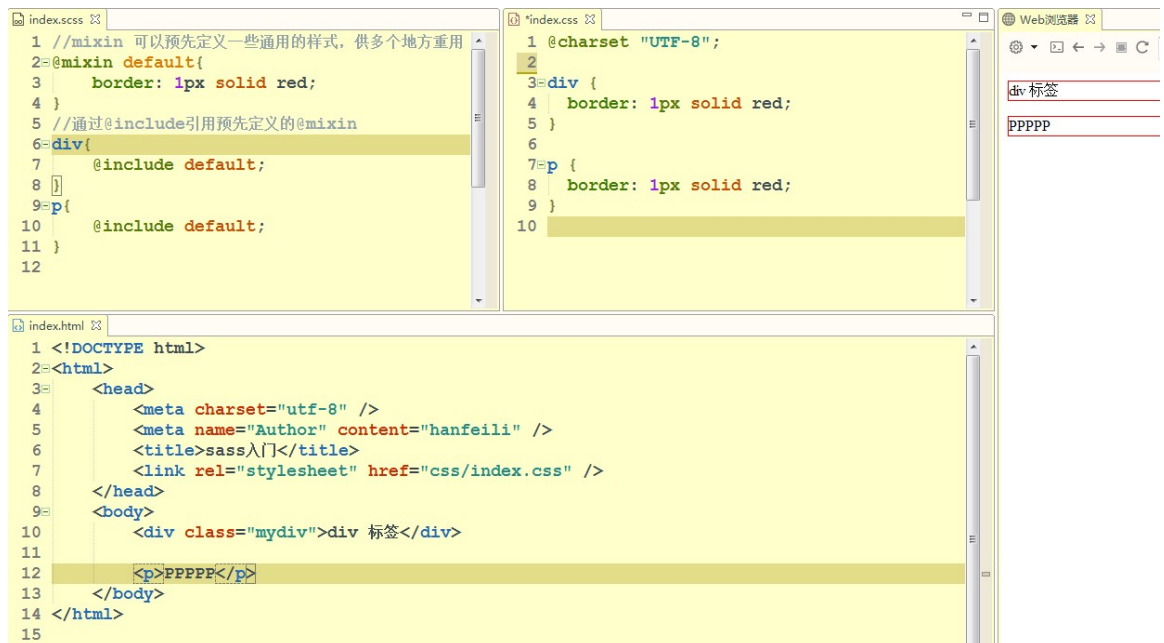
SASS允许一个选择器, 继承另一个选择器通过@extend



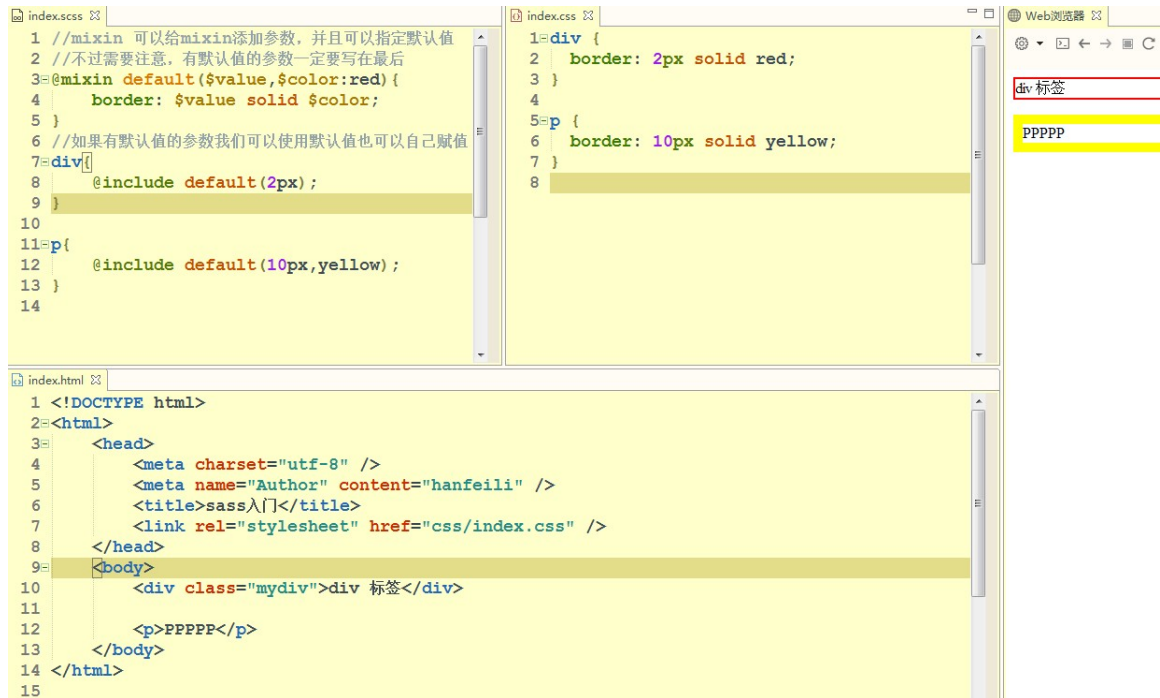
@mixin和@include

@mixin有点像C语言的宏(macro)，是可以重用的代码块

1基本使用：



2带参数使用:



颜色函数

Sass提供了多种内置颜色函数, 方便快捷, 准确生成系列颜色和获取相应的颜色。

<http://sass-lang.com/documentation/Sass/Script/Functions.html>

在Sass的官方文档中, 列出了Sass的颜色函数清单, 从大的方面主要分为RGB、HSL和Opacity三大类函数, 当然其还包括一些其他颜色函数, 比如说adjust-color、change-color等等。

1:RGB颜色函数

rgb颜色只是颜色中的一种表达方式, 其中R是“red”表示红色, 而G是“green”绿色, B是“blue”蓝色。在Sass中为RGB颜色提供六个函数:

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

`rgba($red,$green,$blue,$alpha)`: 根据红、绿、蓝和透明度值创建一个颜色;

`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;

2: HSL颜色函数

在Sass中提供了一系列有关于HSL的颜色函数, 其中常用的有saturation、lightness、adjust-hue、lighten、darken等等。

HSL给我们带来了一个更直观的颜色控制, 我们时常需要让一个颜色比另一个颜色更暗一点或者说更亮一点。比如说“a:hover”状态下我们需要把一个颜色变暗一点, 那么使用“HSL”是非常方便的, 反

而我们使用十六进制那就需要更多的时间调试。而这个“HSL”他只是一个简单的数量变化。

“HSL”所表示的是“H:色相”，“S:饱和度”，“L:亮度”。色相是在色盘上的颜色，颜色的选择是使用饱和度：“0度是红色”，“120度为绿色”和“240度为蓝色”。



`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;

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

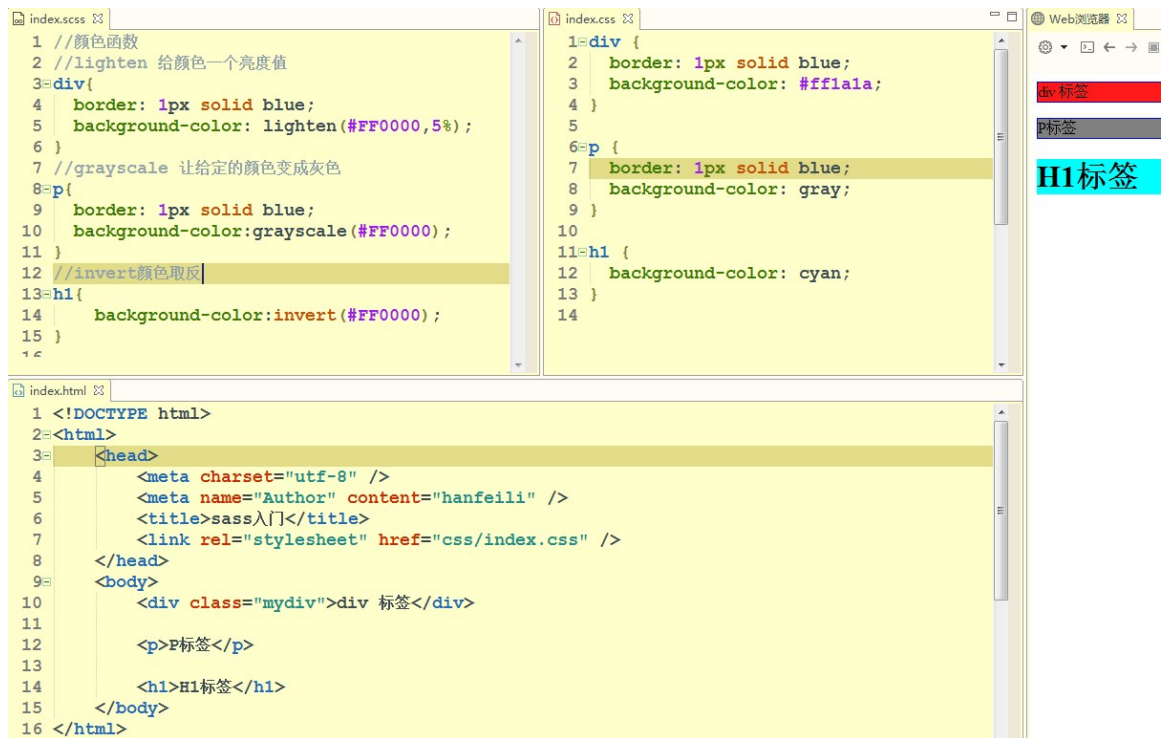
desaturate(\$color,\$amount): 通过改变颜色的饱和度值, 让颜色更少的饱和, 从而创建出一个新的颜色;

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);

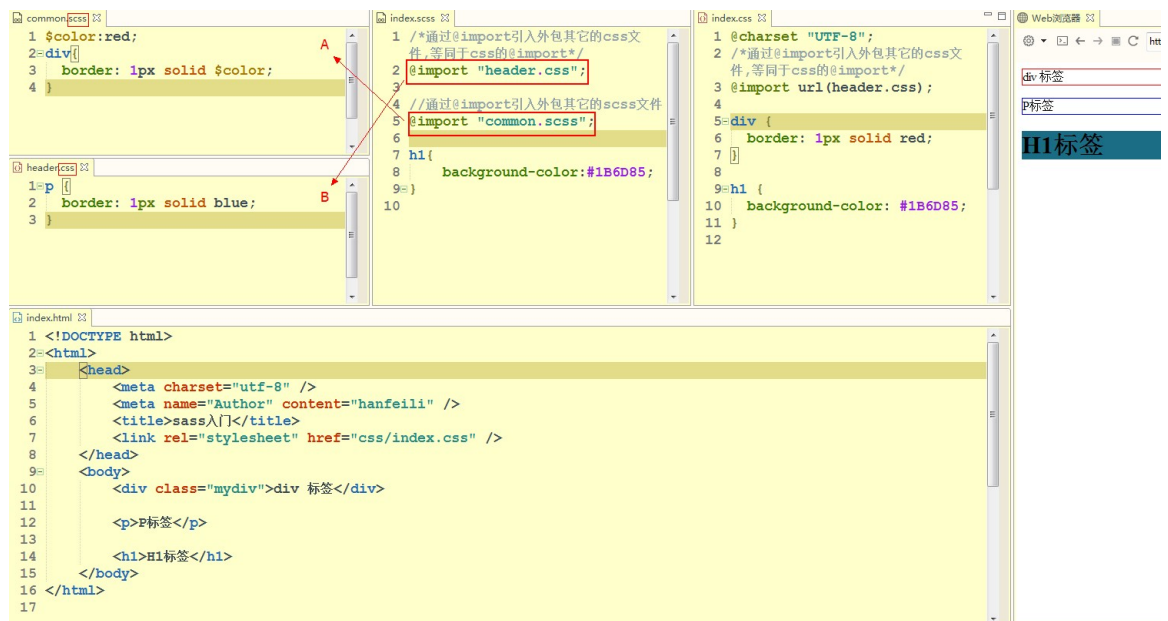
invert(\$color): 反回一个反相色, 红、绿、蓝色值倒过来, 而透明度不变。

实例



@import引入外部文件

@import 引入外部文件，方便进行外部资源重用；



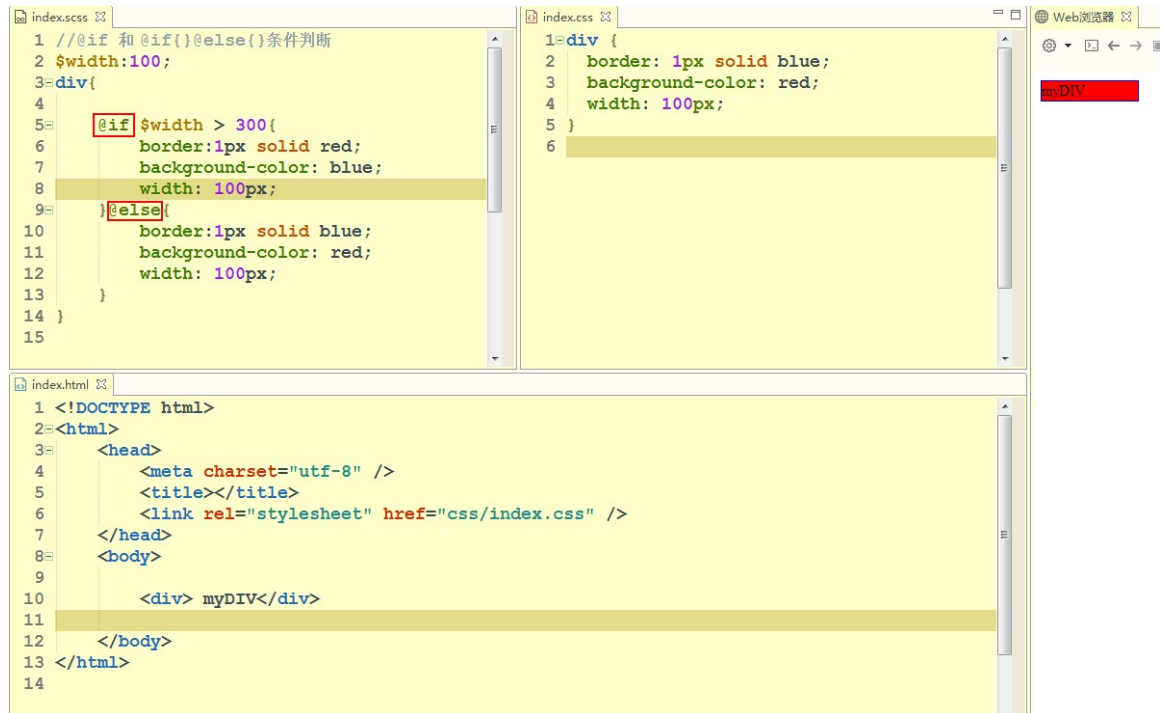
1.3.3. 高级用法——循环语句、条件语句

条件语句

通过@if

和

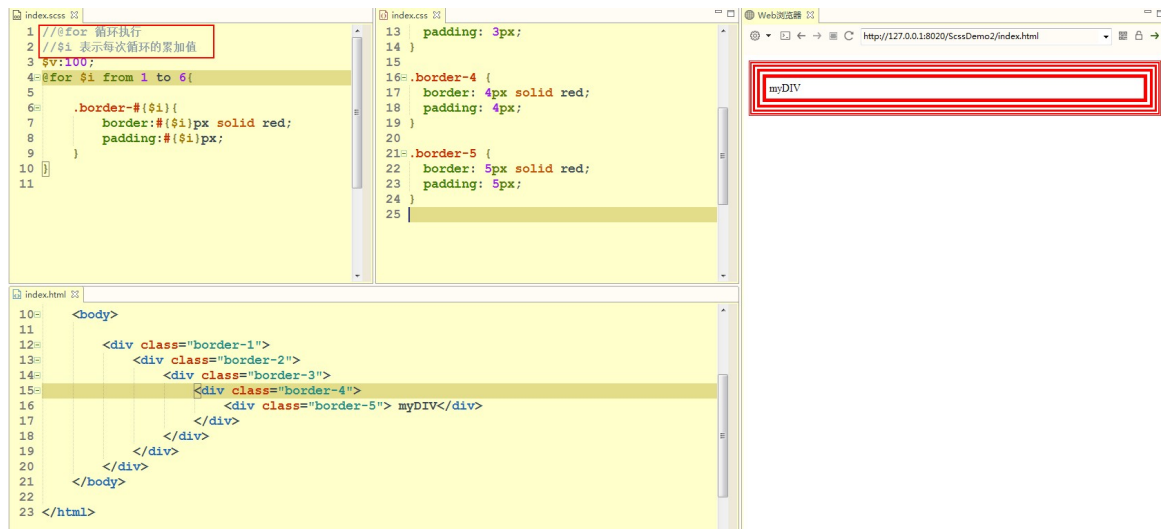
@if{}@else{} 条件判断，根据条件有选择的执行某些样式效果。



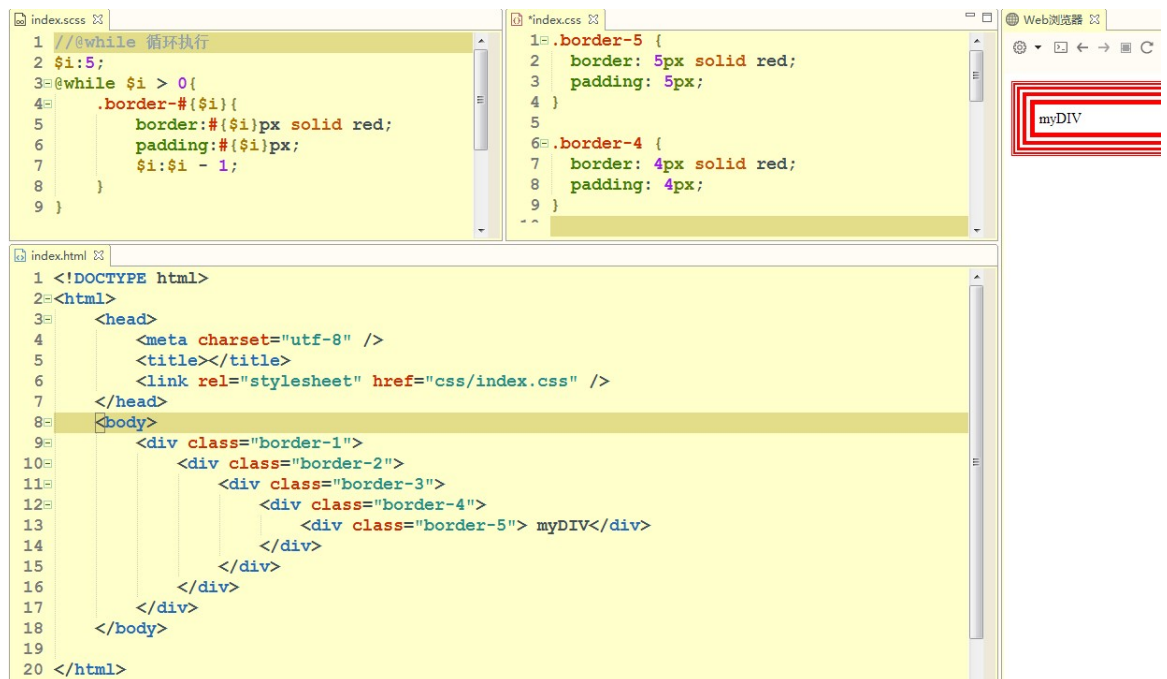
循环语句

@for 循环执行需要重复执行的操作

@for循环

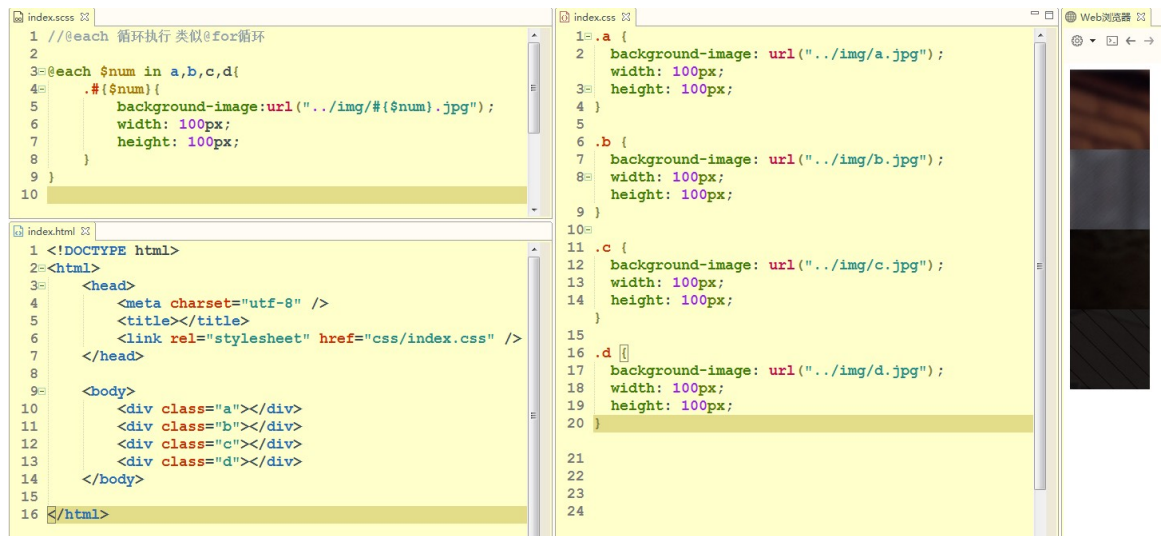


@while循环



@each循环

@each 循环执行 类似@for循环



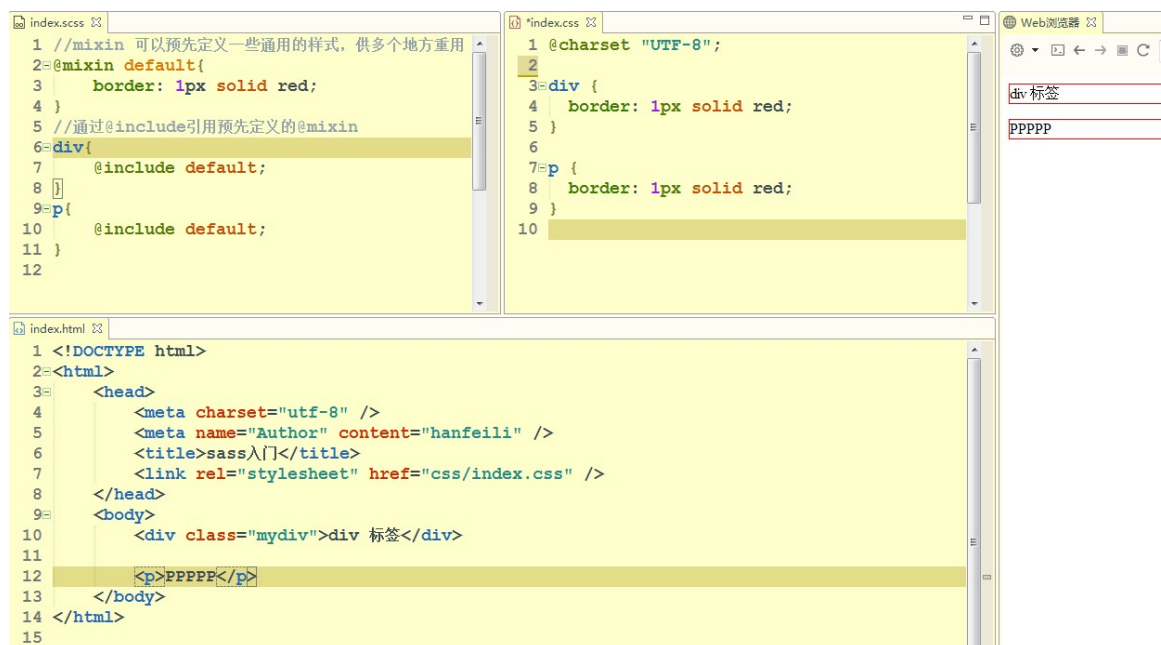
1.3.4. 高级用法——循环语句、自定义函数

4

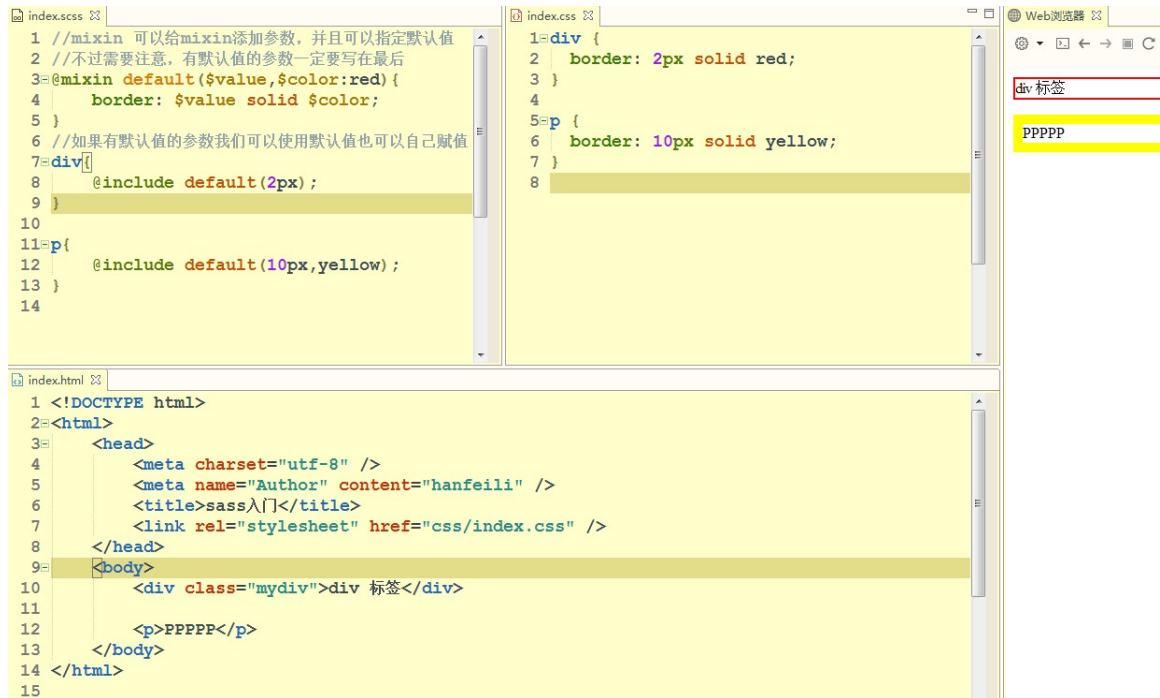
@mixin和@include

@mixin有点像C语言的宏(macro)，是可以重用的代码块

1基本使用：



2带参数使用:



```
index.scss
1 //mixin 可以给mixin添加参数, 并且可以指定默认值
2 //不过需要注意, 有默认值的参数一定要写在最后
3 @mixin default($value,$color:red){
4   border: $value solid $color;
5 }
6 //如果有默认值的参数我们可以使用默认值也可以自己赋值
7
8 div{
9   @include default(2px);
10 }
11
12 p{
13   @include default(10px,yellow);
14 }
15
```

```
index.css
1 div {
2   border: 2px solid red;
3 }
4
5 p {
6   border: 10px solid yellow;
7 }
8
```

```
index.html
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8" />
5     <meta name="Author" content="hanfeili" />
6     <title>sass入门</title>
7     <link rel="stylesheet" href="css/index.css" />
8   </head>
9   <body>
10    <div class="mydiv">div 标签</div>
11
12    <p>PPPPP</p>
13  </body>
14 </html>
15
```

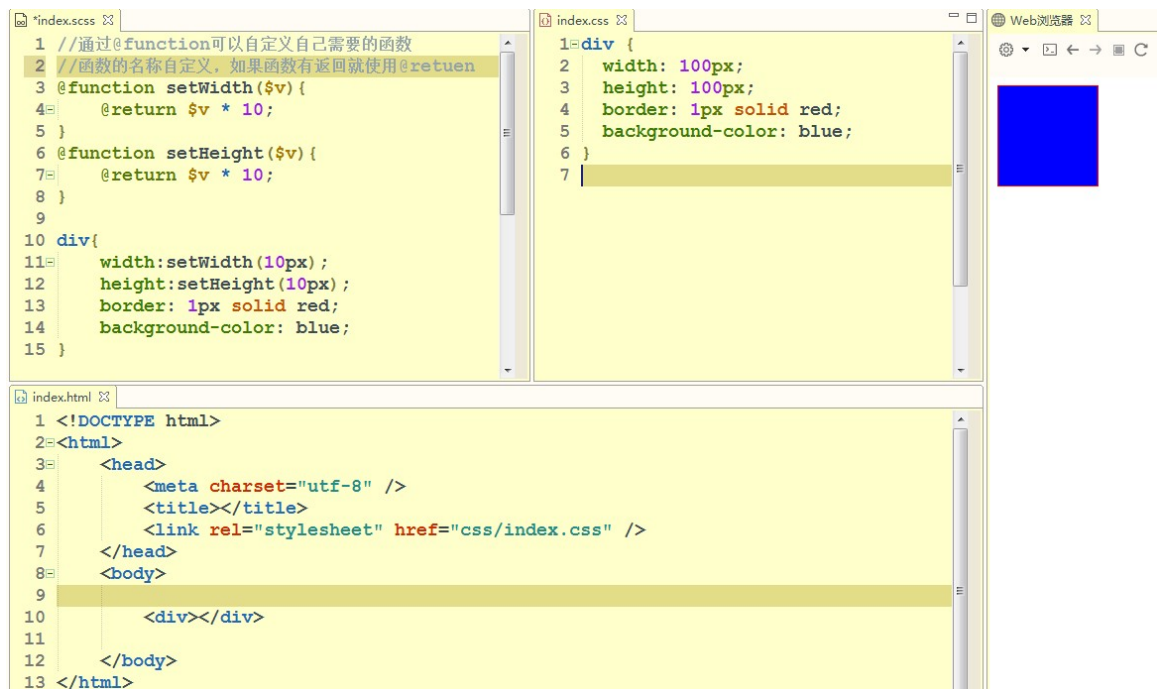
Web浏览器

div 标签

PPPPP

自定义函数

当系统提供的函数不能满足我们的需求的时候, 可以通过@function进行自定义函数的定义。



1.4. 课后作业

1.4.1. 作业1

作业1:

- 1、了解什么是Sass
- 2、了解Sass与Less的区别
- 3、练习Sass的基础语法

1.4.2. 作业2

作业2:

- 1、练习课堂上的实例
- 2、使用@import引入外部文件

1.4.3. 作业3

作业3:

- 1、练习条件语句, 熟悉条件语句的用法
- 2、使用循环语句制作实例

1.4.4. 作业4

作业4:

- 1、动手定义一个属于自己的函数

1.5. 当天小结

1.5.1. sass的介绍及用法

1.5.2. sass中级用法——重用

1.5.3. 高级用法——循环语句、条件语句

1.5.4. 高级用法——循环语句、自定义函数