



css预处理

CSS 预处理器定义了一种新的语言，其基本思想是，用一种专门的编程语言，为 CSS 增加了一些编程的特性，将 CSS 作为目标生成文件，然后开发者就只要使用这种语言进行编码工作。

通俗的说，“CSS 预处理器用一种专门的编程语言，进行 Web 页面样式设计，然后再编译成正常的 CSS 文件，以供项目使用。CSS 预处理器为 CSS 增加一些编程的特性，无需考虑浏览器的兼容性问题”，例如你可以在 CSS 中使用变量、简单的逻辑程序、函数等等在编程语言中的一些基本特性，可以让你的 CSS 更加简洁、适应性更强、可读性更佳，更易于代码的维护等诸多好处。

Sass

Sass: Syntactically Awesome StyleSheets

Sass 是采用 **Ruby** 语言编写的一款 CSS 预处理语言，它诞生于2007年，是最大的成熟的 CSS 预处理语言。最初它是为了配合 HAML（一种缩进式 HTML 预编译器）而设计的，因此有着和 HTML 一样的缩进式风格。

Sass 和 CSS 写法的确存在一定的差异，由于 Sass 是基于 Ruby 写出来，所以其延续了 Ruby 的书写规范。在书写 Sass 时不带有大括号和分号，其主要是依靠严格的缩进方式来控制的。

SCSS 和 CSS 写法无差别，这也是 Sass 后来越来越受大众喜欢原因之一。简单点说，把你现有的“.css”文件直接修改成“.scss”即可使用。

Sass VS SCSS

Sass 和 SCSS 其实是同一种东西，我们平时都称之为 Sass，两者之间不同之处有以下两点：


文件扩展名不同，Sass 是以 “.sass” 后缀为扩展名，而 SCSS 是以 “.scss” 后缀为扩展名

语法书写方式不同，Sass 是以严格的缩进式语法规则来书写，不带大括号({})和分号(;), 而 SCSS 的语法书写和我们的 CSS 语法书写方式非常类似。

.sass

```
1 $color: #f00;  
2 #div1  
3     background-color: $color;  
4
```

.SCSS

```
1 $color: #f00;  
2 #div1{  
3     background-color: $color;  
4 }
```

英文： <http://sass-lang.com>

中文： <https://www.sass.hk>

Github: <https://github.com/sass/sass>

依赖于Ruby： `ruby -v`

Mac： `sudo gem install sass`

Windows:

① **安装ruby**

<https://rubyinstaller.org/downloads/>

② **`gem install sass`**

查看sass版本号： `sass -v`

更新： `gem update sass`

卸载： `gem uninstall sass`

Sass编译成CSS

文件编译:

```
sass <要编译的Sass文件路径>/style.scss:<要输出CSS  
文件路径>/style.css
```

实时:

```
sass --watch <要编译的Sass文件路径>/style.scss:<要  
输出CSS文件路径>/style.css
```

```
sass --watch style:css
```


CSS输出格式

- 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**

sass --watch style:css --style compact

注释

多行注释: `/* */`

单行注释(不会出现在css里): `//`

压缩模式会去掉注释

强制注释: `/*! */`

变量和嵌套

- 用\$符号定义变量 (`$color :#fff`)
- 选择器嵌套
- &符号引用父选择器 (`&:hover` `&-test`)
- 属性嵌套`border:{ left:0;right:0}`
- 特殊变量(一般我们定义的变量都为属性值，可直接使用，但是如果变量作为属性或在某些特殊情况下等则必须要以`#{ $variables}`形式使用。)

mixin混合

```
@mixin name($color:#ccc,$back){
```

```
    ...
```

```
}
```

```
Seletor{
```

```
    @include name(#f00, #0f0);
```

```
}
```

继承和import

- `@extend #div1` 也会继承子选择器
- `@import 'base'; (base.scss)`

数字

数字运算: + - * / ()

绝对值: `abs(-10px)`

四舍五入: `round(5.5)`

向上取整: `ceil(5.5)`

向下取整: `floor(5.5)`

百分比: `percentage(30px / 100px)`

最小值: `min(1,2,3)`

最大值: `max(1,2,3)`

字符串

- 字符串拼接: +
- 大小写转换: to-upper-case(\$var), to-lower-case(\$var)
- 字符串长度: str-length(\$var)
- 索引1开始: str-index(\$target,\$str)
- 插入: str-insert(\$var,\$inserStr,index)

颜色

- `rgb()` `rgba()`
- 更浅: `lighten($color, 20%)`
- 更深: `darken($color, 20%)`
- 更不透明: `opacity($color, 0.3);`
- 更透明: `transparentize($color, 0.3);`

列表list

- 列表长度: `length(5px 10px)`
- 返回索引: `index(1px solid #000, solid)`
- 取出索引对应的值: `nth(5px 10px,2)`
- 列表后面追加值: `append(5px 5px, 10px)`
- 合并列表: `join(5px 5px, 10px 10px, comma(逗号分隔))`

map

`$m: (light: #000, dark: #fff);`
`color: map-get($m, light);`//获取val

返回所有key: `map-keys($m);`

返回所有va'lue: `map-values($m);`

是否包含key: `map-has-key($m, light);`

合并map: `map-merge($m, (gray: #ccc))`

移除: `map-remove($m, light, dark);`

布尔

- $5\text{px} > 10\text{px}$ (\leq 、 \geq 、 $==$ 、 $<$ 、 $>$)
- and、 or、 not



Interpolation

- `#{ $val }`
- 在注释中引用
- 在选择器中引用
- 在属性名中引用

Data_type


- sass -i type-of()
- 1. 数字：如，1、 2、 13、 10px;
- 2. 字符串：有引号字符串或无引号字符串，如，” foo”、 ’ bar’、 baz;
- 3. 颜色：如，blue、 #04a3f9、 rgba(255,0,0,0.5);
- 4. 布尔型：如，true、 false;
- 5. 空值：如，null;
- 6. 值列表：用空格或者逗号分开，如，1.5em 1em 0 2em。
- 7. map: (aa:’ aa’ ,bb:’ bb’)

控制指令

- @if @else if @else
- @for \$i from 开始 through 结束{}
- @for \$i from 开始 to结束{}
- @each \$var in \$list{}
- @while \$i{}

函数

- @function name(参数1, 参数2...){
 @return ...
 - }

 - @warn
 - @error
- 



Thank you

谢

谢

观

看