

Sass简介



我们写了这么久的代码，每次写页面都需要配合CSS，CSS里面重复代码其实很多，包含通配颜色值、容器大小等。那我们能否使用JavaScript声明变量的方式解决这些问题呢？原本的CSS当然是不支持的，但是有其他方案，他就是Sass！！！

Sass世界上最成熟、最稳定、最强大的专业级CSS扩展语言！

为什么使用Sass？

CSS 本身语法不够强大，导致重复编写一些代码，无法实现复用，而且代码也不方便维护。Sass 引入合理的样式复用机制，增加了规则、变量、混入、选择器、继承、内置函数等等特性。

Sass优点

Sass 是一个 CSS 预处理器

Sass 完全兼容所有版本的 CSS

Sass 扩展了 CSS3，增加了规则、变量、混入、选择器、继承、内置函数等等特性

Sass环境搭建



sass 基于 Ruby 语言开发而成，因此安装 sass 前需要安装Ruby。
(注:mac下自带Ruby无需在安装Ruby!)

安装Ruby

Ruby下载地址: <https://rubyinstaller.org/downloads/>

安装完成后需测试安装有没有成功,运行 **CMD** 输入以下命令:

```
1 ruby -v
2 // ruby 3.1.2p20
```

RubyGems

RubyGems 类似 Nodejs 中的 npm

RubyGems 一直以来在国内都非常难访问到, 我们需要将他的源替换国内的

```
1 // 删除替换原gem源
2 gem sources --add https://gems.ruby-
china.com/ --remove https://rubygems.org/
3
4 // 打印是否替换成功
5 gem sources -l
6 // https://gems.ruby-china.com/
```

sass安装

Ruby 自带一个叫做 **RubyGems** 的系统, 用来安装基于 **Ruby** 的软件。我们可以使用这个系统来轻松地安装 **Sass** 和 **Compass**

```
1 // 安装如下(如mac安装遇到权限问题需加 sudo gem  
install sass)  
2 gem install sass  
3 gem install compass  
4 // 检测是否安装成功  
5 sass -v      // Ruby Sass 3.7.4  
6 compass -v   // Compass 1.0.3
```

实时效果反馈

1. Sass安装命令是什么：

- A npm install -g sass
- B gem install -g sass
- C gem install sass
- D gem install compass

答案

1=>C

编译Sass



`sass` 编译有很多种方式，如命令行编译模式、编译软件 `koala` 、工程
化工具 `webpack` 等

编写Sass代码

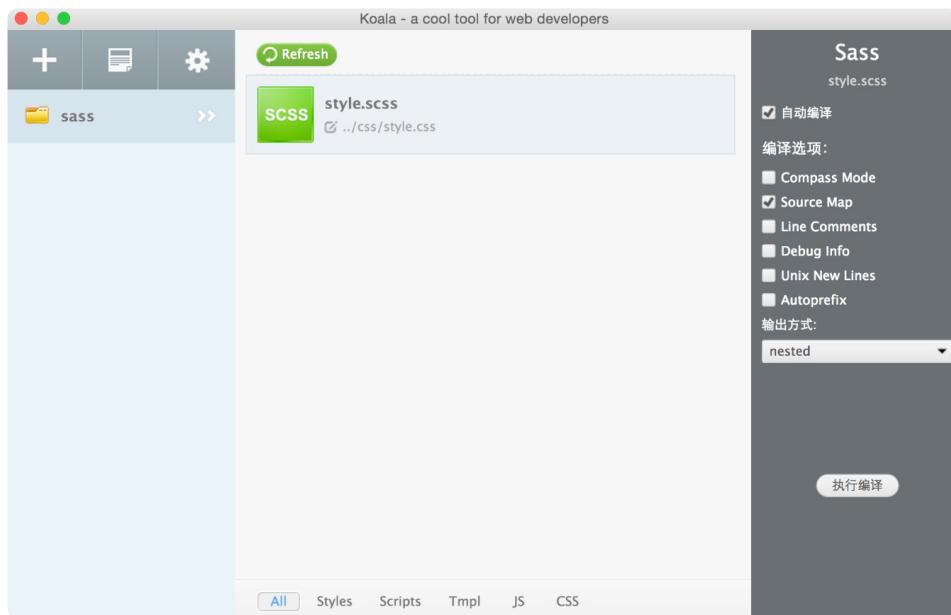
Sass文件的后缀名为 `.scss`

```
1 .box {  
2     width: 300px;  
3     height: 400px;  
4  
5     &-title {  
6         height: 30px;  
7         line-height: 30px;  
8     }  
9 }
```

命令行编译

```
1 sass style.scss style.css
```

koala 软件方式编译



出了上述的几种方式外，我们还可以用工程化软件进行编译Sass，例如：[Webpack](#) (后续讲解)

实时效果反馈

1. 命令行编译Sass的命令是：

- A less
- B lessc
- C sass
- D scss

答案

1=>C

Sass_变量



使用变量

sass让人们受益的一个重要特性就是它为css引入了变量。你可以把反复使用的css属性值定义成变量，然后通过变量名来引用它们，而无需重复书写这一属性值。或者，对于仅使用过一次的属性值，你可以赋予其一个易懂的变量名，让人一眼就知道这个属性值的用途

sass 使用 \$ 符号来标识变量

比如\$highlight-color和\$sidebar-width。为什么选择\$符号呢？因为它好认、更具美感，且在CSS中并无他用，不会导致与现存或未来的css语法冲突

```

1 $highlight-color: #F90;
2 .selected {
3   border: 1px solid $highlight-color;
4 }

```

变量名用中划线还是下划线分隔

`sass` 的变量名可以与 `css` 中的属性名和选择器名称相同，包括中划线和下划线。这完全取决于个人的喜好

```

1 $link-color: blue;
2 a {
3   color: $link_color;
4 }

```

实时效果反馈

1. Sass声明变量，关键字是：

- A
- B
- C
- D

答案

1=>B

Sass_嵌套



`css` 中重复写选择器是非常恼人的。如果要写一大串指向页面中同一块的样式时，往往需要一遍又一遍地写同一个 `ID`

```
1 #content article h1 { color: #333 }
2 #content article p { margin-bottom: 1.4em }
3 #content aside { background-color: #EEE }
```

像这种情况，`sass` 可以让你只写一遍，且使样式可读性更高。在 Sass 中，你可以像俄罗斯套娃那样在规则块中嵌套规则块。`sass` 在输出 `css` 时会帮你把这些嵌套规则处理好，避免你的重复书写。

```

1 #content {
2   article {
3     h1 { color: #333 }
4     p { margin-bottom: 1.4em }
5   }
6   aside { background-color: #EEE }
7 }
```

大多数情况下这种简单的嵌套都没问题，但是有些场景下不行，比如你想要在嵌套的选择器里边立刻应用一个类似于 `:hover` 的伪类。为了解决这种以及其他情况，`sass` 提供了一个特殊结构 `&`

```

1 article a {
2   color: blue;
3   &:hover { color: red }
4 }
```

实时效果反馈

1. Sass中，伪类选择器标识符是什么：

- A @
- B \$
- C &
- D #

答案

1=>C

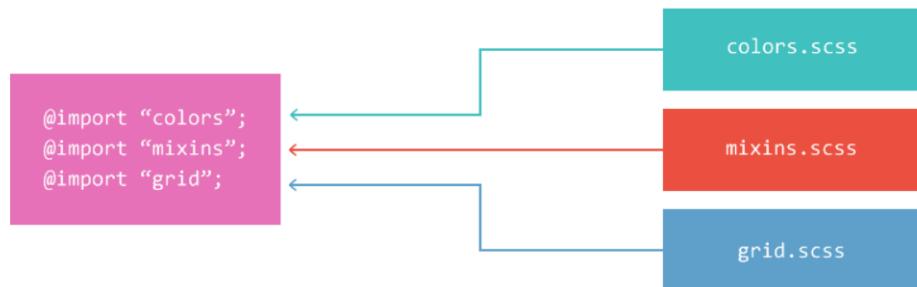
导入Sass文件



`css` 有一个特别不常用的特性，即 `@import` 规则，它允许在一个 `css` 文件中导入其他 `css` 文件。然而，后果是只有执行到 `@import` 时，浏览器才会去下载其他 `css` 文件，这导致页面加载起来特别慢。

`sass` 也有一个 `@import` 规则，但不同的是，`sass` 的 `@import` 规则在生成 `css` 文件时就把相关文件导入进来。这意味着所有相关的样式被归纳到了同一个 `css` 文件中，而无需发起额外的下载请求。

使用 `sass` 的 `@import` 规则并不需要指明被导入文件的全名。你可以省略 `.sass` 或 `.scss` 文件后缀



默认变量值

一般情况下，你反复声明一个变量，只有最后一处声明有效且它会覆盖前边的值

```

1 $link-color: blue;
2 $link-color: red;
3 a {
4     color: $link-color; // red
5 }
```

`!default` 作用，是将这次变量值的优先级降到最低

```

1 $link-color: blue;
2 $link-color: red !default;
3 a {
4     color: $link-color; // blue
5 }
```

嵌套导入

跟原生的 `css` 不同，`sass` 允许 `@import` 命令写在 `css` 规则内

```

1 // aside.scss
2 aside {
3     background: blue;
4     color: white;
5 }
```

```
1 .blue-theme {  
2     @import "blue-theme"  
3 }
```

实时效果反馈

1. Sass中，导入文件的关键字是什么：

- A @import
- B import
- C include
- D #include

答案

1=>A

静默注释



`css` 中注释的作用包括帮助你组织样式、以后你看自己的代码时明白为什么这样写，以及简单的样式说明。但是，你并不希望每个浏览网站源码的人都能看到所有注释。

`sass` 另外提供了一种不同于 `css` 标准注释格式 `/* ... */` 的注释语法，即静默注释，其内容不会出现在生成的 `css` 文件中。静默注释的语法跟 `JavaScript`、`Java` 等类 `C` 的语言中单行注释的语法相同，它们以 `//` 开头，注释内容直到行末。

```

1 body {
2   color: #333; // 这种注释内容不会出现在生成的css
3   padding: 0; /* 这种注释内容会出现在生成的css文件
4   中 */
5 }
```

实际上，`css` 的标准注释格式 `/* ... */` 内的注释内容亦可在生成的 `css` 文件中抹去。当注释出现在原生 `css` 不允许的地方，如在 `css` 属性或选择器中，`sass` 将不知如何将其生成到对应 `css` 文件中的相应位置，于是这些注释被抹掉

```

1 body {
2   color /* 这块注释内容不会出现在生成的css中 */:
3   #333;
4 }
```

Sass_混合



如果你的整个网站中有几处小小的样式类似（例如一致的颜色和字体），那么使用变量来统一处理这种情况是非常不错的选择。但是当你的样式变得越来越复杂，你需要大段大段的重用样式的代码，独立的变量就没办法应付这种情况了。你可以通过 sass 的混合实现大段样式的重用

```

1 @mixin rounded-corners {
2     border-radius: 5px;
3 }
4
5 .notice {
6     background-color: green;
7     border: 2px solid #00aa00;
8     @include rounded-corners;
9 }

```

混合传参

混合并不一定总得生成相同的样式。可以通过在 `@include` 时给混合传参，来定制混合生成的精确样式。

并且，我们还可以指定参数默认值，当参数不传递的时候，可以使用默认值

```

1 @mixin link-colors($color, $hover:blue) {
2     color: $color;
3
4     &:hover {
5         color: $hover;
6     }
7
8 }
9
10 .box1{
11     @include link-colors(red,green);
12 }
13 .box2{
14     @include link-colors(red);

```

实时效果反馈

1. Sass中，混合引入的关键字是什么：

- A @import
- B import
- C include
- D @include

答案

1=>D

使用选择器继承来精简CSS



使用 sass 的时候，最后一个减少重复的主要特性就是选择器继承。选择器继承是说一个选择器可以继承为另一个选择器定义的所有样式。这个通过 @extend 语法实现

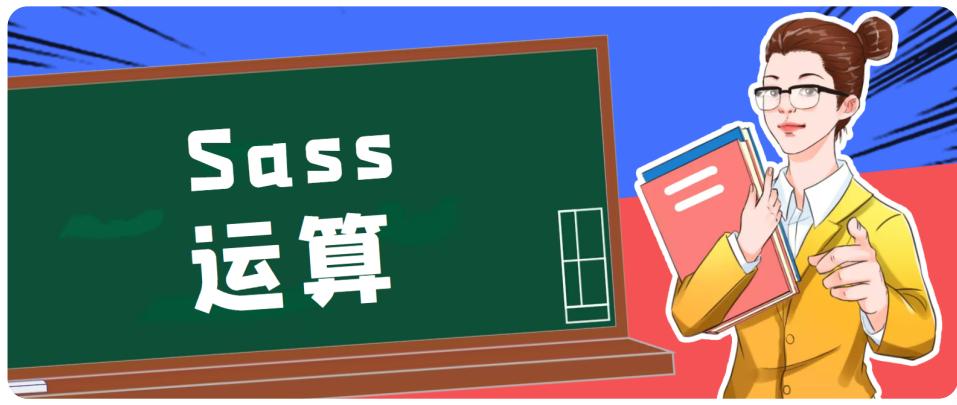
```

1 .error {
2     border: 1px solid red;
3     background-color: #fdd;
4 }
5
6 .seriousError {
7     @extend .error;
8     border-width: 3px;
9 }
```

继承原理

通常使用继承会让你的 css 美观、整洁。因为继承只会在生成 css 时复制选择器，而不会复制大段的 css 属性。

Sass 运算



Sass支持数字的加减乘除等运算 (`+, -, *, /`)，如果必要会在不同单位间转换值

```
1 $size:100px;  
2 .root{  
3     width: $size + 200px;  
4 }
```

温馨提示

运算过程中，不同单位不要参与运算

```
1 $size:1000px;  
2 .root{  
3     width: $size - 200px;  
4 }
```

```
1 $size:1000px;  
2 .root{  
3     width: $size * 2;  
4 }
```

```

1 $size:1000px;
2 .root{
3     width: ($size / 2);
4 }
```

运算中的优先级

圆括号可以用来影响运算的顺序

```

1 $size:1000px;
2 .root{
3     width: ($size - 400) * 2;
4 }
```

应用场景

在移动端适配的时候，通常会有REM计算

例如：根 `font-size` 为 `50px`，设计师给的数值是 `340px`

```

1 $fontREM:50;
2 .root{
3     width:(340rem / $fontREM);
```

控制指令



Sass提供了一些基础的控制指令，比如在满足一定条件时引用样式，或者设定范围重复输出格式。控制指令是一种高级功能，日常编写过程中并不常用到。

@if

当 `@if` 的表达式返回值不是 `false` 或者 `null` 时，条件成立，输出 `{}` 内的代码

```

1 p {
2   @if 1+1==2 {
3     border: 1px solid;
4   }
5   @if 5 < 3 {
6     border: 2px dotted;
7   }
8   @if null {
9     border: 3px double;
10  }
11 }
```

@for

`@for` 指令可以在限制的范围内重复输出格式，每次按要求（变量的值）对输出结果做出变动。

```
1 @for $i from 1 through 3 {  
2     .item-#${$i} {  
3         width: 2px * $i;  
4     }  
5 }
```