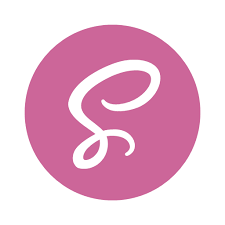
**Sass 教程**



Sass (英文全称：Syntactically Awesome Stylesheets) 是一个最初由 Hampton Catlin 设计并由 Natalie Weizenbaum 开发的层叠样式表语言。

Sass 是一个 CSS 预处理器。

Sass 是 CSS 扩展语言，可以帮助我们减少 CSS 重复的代码，节省开发时间。

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 生成良好格式化的 CSS 代码，易于组织和维护。

Sass 文件后缀为 **.scss**。

**Sass 实例**

*/\* 定义变量与值 \*/*  
$bgcolor: lightblue;  
$textcolor: darkblue;  
$fontsize: 18px;  
  
*/\* 使用变量 \*/*  
body {  
  **background-color**: $bgcolor;  
  **color**: $textcolor;  
  **font-size**: $fontsize;  
}

**阅读本教程前，您需要了解的知识：**

阅读本教程，您需要有以下基础：

* [HTML 教程](https://www.runoob.com/html/html-tutorial.html)
* [CSS 教程](https://www.runoob.com/css/css-tutorial.html)

**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，而且在代码也不方便维护。

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。

我们可以举个例子，我们会在 CSS 中重复使用很多次十六进制的颜色代码，当有了变量之后，如果要改变颜色代码，只要修改变量的值就好了：

**Sass 实例**

*/\* 定义颜色变量，要修改颜色值，修改这里就可以了 \*/*  
$primary\_1: #a2b9bc;  
$primary\_2: #b2ad7f;  
$primary\_3: #878f99;  
  
*/\* 使用变量 \*/*  
.main-header {  
  **background-color**: $primary\_1;  
}  
  
.menu-**left** {  
  **background-color**: $primary\_2;  
}  
  
.menu-**right** {  
  **background-color**: $primary\_3;  
}

**Sass 是如何工作的？**

浏览器并不支持 Sass 代码。因此，你需要使用一个 Sass 预处理器将 Sass 代码转换为 CSS 代码。

# Sass 安装

本章节我们主要介绍 Sass 的安装与使用。

### NPM 安装

我们可以使用 npm（NPM 使用介绍）来安装 Sass。

npm install -g sass

**注：**国内 npm 建议使用淘宝镜像来安装，参考：[NPM 国内慢的问题解决](https://www.runoob.com/w3cnote/npm-slow-use-cnpm.html)

### Windows 上安装

我们可以使用 Windows 的包管理器 [Chocolatey](https://chocolatey.org/) 来安装：

choco install sass

### Mac OS X (Homebrew)安装

Mac OS 可以使用 [Homebrew](https://brew.sh/) 包管理器来安装：

brew install sass/sass/sass

更多安装方法可以查看官网：<https://sass-lang.com/install>

## 还可以使用vscode的插件：easy sass，

### 1）在vscode的插件库里面搜索安装

### 2）安装好后，需要做一些配置：

#### a.点击easy sass的设置按钮，选中扩展配置

|  |
| --- |
|  |

#### 会弹出如下界面：

|  |
| --- |
|  |

## 在搜索框里面输入compile，出现如下界面，这个选项需要勾选

|  |
| --- |
|  |

## 搜索target，在出现的界面中配置输出路径（注意它是从当前工作区为参考来设置路径的）

|  |  |
| --- | --- |
| 项目目录结构 | 路径设置 |

## 配置好后需要在您的项目中新建一个sass目录用来写sass文件

|  |
| --- |
|  |

## 注意sass文件的后缀名是.scss

## 使用实例1：嵌套使用

|  |  |
| --- | --- |
| html  |  | | --- | | <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>Document</title>      <!-- 注意：浏览器不支持sass只能引入css文件 -->      <link rel="stylesheet" href="../css/index.css">  </head>  <body>      <ul class="one">          <li>              1              <ul class="two">                  <li>1-1</li>                  <li>1-2</li>                  <li>1-3</li>              </ul>          </li>          <li>              2              <ul>                  <li>2-1</li>                  <li>2-2</li>                  <li>2-3</li>              </ul>          </li>          <li>              3              <ul>                  <li>3-1</li>                  <li>3-2</li>                  <li>3-3</li>              </ul>          </li>      </ul>  </body>  </html> | |
| sass文件:index.scss  |  | | --- | | // 给one的第一个li设置样式  .one{      > li{          &:first-child{              border: 1px solid red;              .two{                  li{                      &:first-child{                          background-color: pink;                      }                  }              }          }      }  } | | 生成的css文件：index.css   |  | | --- | | .one > li:first-child {    border: 1px solid red;  }  .one > li:first-child .two li:first-child {    background-color: pink;  } | | |

#### 注意：嵌套是sass最常用的功能，但是嵌套最好不要超过3层，否者很难阅读

## 使用介绍

我们的教程使用的是 npm 安装的 sass，安装完成后可以查看版本：

$ sass --version 1.22.12 compiled with dart2js 2.5.0 接下来我们创建一个 runoob-test.scss 文件，内容为：

## runoob-test.scss 文件代码:

*/\* 定义变量与值 \*/*  
$bgcolor: lightblue;  
$textcolor: darkblue;  
$fontsize: 18px;  
  
*/\* 使用变量 \*/*  
body {  
  **background-color**: $bgcolor;  
  **color**: $textcolor;  
  **font-size**: $fontsize;  
}

然后在命令行输入下面命令，即将 .scss 文件转化的 css 代码：

$ sass runoob-test.scss

@charset "UTF-8";

/\* 定义变量与值 \*/

/\* 使用变量 \*/

body {

background-color: lightblue;

color: darkblue;

font-size: 18px;

}

我们可以在后面再跟一个 .css 文件名，将代码保存到文件中：

$ sass runoob-test.scss runoob-test.css

这是会在当前目录下生成 runoob-test.css 文件，代码如下：

@charset "UTF-8";

/\* 定义变量与值 \*/

/\* 使用变量 \*/

body {

background-color: lightblue;

color: darkblue;

font-size: 18px;

}

/\*# sourceMappingURL=runoob-test.css.map \*/

# \*Sass 变量

变量用于存储一些信息，它可以重复使用。

Sass 变量可以存储以下信息：

* 字符串
* 数字
* 颜色值
* 布尔值
* 列表
* null 值

Sass 变量使用 **$** 符号：

$variablename: value;

以下实例设置了四个变量：myFont, myColor, myFontSize, 和 myWidth。

变量声明后我们就可以在代码中使用它：

## Sass 代码：

$myFont: Helvetica, sans-serif;  
$myColor: red;  
$myFontSize: 18px;  
$myWidth: 680px;  
  
body {  
  **font-family**: $myFont;  
  **font-size**: $myFontSize;  
  **color**: $myColor;  
}  
  
#container {  
  **width**: $myWidth;  
}

将以上代码转换为 CSS 代码，如下所示：

## Css 代码：

body {  
  **font-family**: Helvetica, sans-serif;  
  **font-size**: 18px;  
  **color**: red;  
}  
  
#container {  
  **width**: 680px;  
}

## Sass 作用域

Sass 变量的作用域只能在当前的层级上有效果，如下所示 h1 的样式为它内部定义的 green，p 标签则是为 red。

## Sass 代码：

$myColor: red;  
  
h1 {  
  $myColor: green;   *// 只在 h1 里头有用，局部作用域*  
  **color**: $myColor;  
}  
  
p {  
  **color**: $myColor;  
}

将以上代码转换为 CSS 代码，如下所示：

## Css 代码：

h1 {  
  **color**: green;  
}  
  
p {  
  **color**: red;  
}

### !global

当然 Sass 中我们可以使用 **!global** 关键词来设置变量是全局的：

## Sass 代码

$myColor: red;  
  
h1 {  
  $myColor: green !global;  *// 全局作用域*  
  **color**: $myColor;  
}  
  
p {  
  **color**: $myColor;  
}

现在 p 标签的样式就会变成 green。

将以上代码转换为 CSS 代码，如下所示：

## Css 代码

h1 {  
  **color**: green;  
}  
  
p {  
  **color**: green;  
}

**注意：**所有的全局变量我们一般定义在同一个文件，如：**\_globals.scss**，然后我们使用 [**@include**](https://www.runoob.com/sass/sass-mixin-include.html) 来包含该文件。

# \*Sass 嵌套规则与属性

Sass 嵌套 CSS 选择器类似于 HTML 的嵌套规则。

如下我们嵌套一个导航栏的样式：

## Sass 代码：

nav {  
  ul {  
    **margin**: 0;  
    **padding**: 0;  
    **list-style**: none;  
  }  
  li {  
    **display**: inline-block;  
  }  
  a {  
    **display**: block;  
    **padding**: 6px 12px;  
    **text-decoration**: none;  
  }  
}

实例中，ul, li, 和 a 选择器都嵌套在 nav 选择器中

将以上代码转换为 CSS 代码，如下所示：

## Css 代码：

nav ul {  
  **margin**: 0;  
  **padding**: 0;  
  **list-style**: none;  
}  
nav li {  
  **display**: inline-block;  
}  
nav a {  
  **display**: block;  
  **padding**: 6px 12px;  
  **text-decoration**: none;  
}

## Sass 嵌套属性

很多 CSS 属性都有同样的前缀，例如：font-family, font-size 和 font-weight ， text-align, text-transform 和 text-overflow。

在 Sass 中，我们可以使用嵌套属性来编写它们：

## Sass 代码：

**font**: {  
  family: Helvetica, sans-serif;  
  **size**: 18px;  
  weight: bold;  
}  
  
text: {  
  align: center;  
  transform: lowercase;  
  **overflow**: hidden;  
}

将以上代码转换为 CSS 代码，如下所示：

## Css 代码：

**font-family**: Helvetica, sans-serif;  
**font-size**: 18px;  
**font-weight**: bold;  
  
**text-align**: center;  
**text-transform**: lowercase;  
text-overflow: hidden;

# Sass @import

Sass 可以帮助我们减少 CSS 重复的代码，节省开发时间。

我们可以安装不同的属性来创建一些代码文件，如：变量定义的文件、颜色相关的文件、字体相关的文件等。

## Sass 导入文件

类似 CSS，Sass 支持 **@import** 指令。

@import 指令可以让我们导入其他文件等内容。

CSS @import 指令在每次调用时，都会创建一个额外的 HTTP 请求。但，Sass @import 指令将文件包含在 CSS 中，不需要额外的 HTTP 请求。

Sass @import 指令语法如下：

@import filename;

**注意：**包含文件时不需要指定文件后缀，Sass 会自动添加后缀 .scss。

此外，你也可以导入 CSS 文件。

导入后我们就可以在主文件中使用导入文件等变量。

以下实例，导入 variables.scss、colors.scss 和 reset.scss 文件。

## Sass 代码：

@import "variables";  
@import "colors";  
@import "reset";

接下来我们创建一个 reset.scss 文件：

## reset.scss 文件代码：

html,  
body,  
ul,  
ol {  
  **margin**: 0;  
  **padding**: 0;  
}

然后我们在 standard.scss 文件中使用 @import 指令导入 reset.scss 文件：

## standard.scss 文件代码：

@import "reset";  
  
body {  
  **font-family**: Helvetica, sans-serif;  
  **font-size**: 18px;  
  **color**: red;  
}

将以上代码转换为 CSS 代码，如下所示：

## Css 代码：

html, body, ul, ol {  
  **margin**: 0;  
  **padding**: 0;  
}  
  
body {  
  **font-family**: Helvetica, sans-serif;  
  **font-size**: 18px;  
  **color**: red;  
}

## Sass Partials

如果你不希望将一个 Sass 的代码文件编译到一个 CSS 文件，你可以在文件名的开头添加一个下划线。这将告诉 Sass 不要将其编译到 CSS 文件。

但是，在导入语句中我们不需要添加下划线。

Sass Partials 语法格式：

\_filename;

以下实例创建一个 \_colors.scss 的文件，但是不会编译成 \_colors.css 文件：

## \_colors.scss 文件代码：

$myPink: #EE82EE;  
$myBlue: #4169E1;  
$myGreen: #8FBC8F;

如果要导入该文件，则不需要使用下划线：

## 实例

@import "colors";  
  
body {  
  **font-family**: Helvetica, sans-serif;  
  **font-size**: 18px;  
  **color**: $myBlue;  
}

**注意：**请不要将带下划线与不带下划线的同名文件放置在同一个目录下，比如，\_colors.scss 和 colors.scss 不能同时存在于同一个目录下，否则带下划线的文件将会被忽略。

# \*Sass @mixin 与 @include

@mixin 指令允许我们定义一个可以在整个样式表中重复使用的样式。

@include 指令可以将混入（mixin）引入到文档中。

## 定义一个混入

混入(mixin)通过 @mixin 指令来定义。 @mixin name { property: value; property: value; ... }

以下实例创建一个名为 "important-text" 的混入：

## Sass 代码：

@mixin important-text {  
  **color**: red;  
  **font-size**: 25px;  
  **font-weight**: bold;  
  **border**: 1px solid blue;  
}

**注意：**Sass 的连接符号 - 与下划线符号 \_ 是相同的，也就是 @mixin important-text { } 与 @mixin important\_text { } 是一样的混入。

### 使用混入

@include 指令可用于包含一混入：

## Sass @include 混入语法：

selector {  
  @include mixin-name;  
}

因此，包含 important-text 混入代码如下：

## 实例

.danger {  
  @include important-text;  
  **background-color**: green;  
}

将以上代码转换为 CSS 代码，如下所示：

## Css 代码：

.danger {  
  **color**: red;  
  **font-size**: 25px;  
  **font-weight**: bold;  
  **border**: 1px solid **blue**;  
  **background-color**: green;  
}

混入中也可以包含混入，如下所示：

## 实例

@mixin special-text {  
  @include important-text;  
  @include link;  
  @include special-**border**;  
}

## 向混入传递变量

混入可以接收参数。

我们可以向混入传递变量。

定义可以接收参数的混入：

## 实例

*/\* 混入接收两个参数 \*/*  
@mixin bordered($color, $width) {  
  **border**: $width solid $color;  
}  
  
.myArticle {  
  @include bordered(blue, 1px);  *// 调用混入，并传递两个参数*  
}  
  
.myNotes {  
  @include bordered(red, 2px); *// 调用混入，并传递两个参数*  
}

以上实例的混入参数为设置边框的属性 (color 和 width) 。

将以上代码转换为 CSS 代码，如下所示：

## Css 代码：

.myArticle {  
  **border**: 1px solid **blue**;  
}  
  
.myNotes {  
  **border**: 2px solid red;  
}

混入的参数也可以定义默认值，语法格式如下：

## 实例

@mixin bordered($color: blue, $width: 1px) {  
  **border**: $width solid $color;  
}

在包含混入时，你只需要传递需要的变量名及其值：

## 实例

@mixin sexy-**border**($color, $width: 1in) {  
  **border**: {  
    **color**: $color;  
    **width**: $width;  
    style: dashed;  
  }  
}  
p { @include sexy-**border**(blue); }  
h1 { @include sexy-**border**(blue, 2in); }

将以上代码转换为 CSS 代码，如下所示：

## Css 代码：

p {  
  **border-color**: **blue**;  
  **border-width**: 1in;  
  **border-style**: dashed; }  
  
h1 {  
  **border-color**: **blue**;  
  **border-width**: 2in;  
  **border-style**: dashed;  
}

### 可变参数

有时，不能确定一个混入（mixin）或者一个函数（function）使用多少个参数，这时我们就可以使用 **...** 来设置可变参数。

例如，用于创建盒子阴影（box-shadow）的一个混入（mixin）可以采取任何数量的 box-shadow 作为参数。

## 实例

@mixin **box-shadow**($shadows...) {  
      -moz-**box-shadow**: $shadows;  
      -webkit-**box-shadow**: $shadows;  
      **box-shadow**: $shadows;  
}  
  
.shadows {  
  @include **box-shadow**(0px 4px 5px #666, 2px 6px 10px #999);  
}

将以上代码转换为 CSS 代码，如下所示：

## Css 代码：

.shadows {  
  -moz-box-shadow: 0px 4px 5px #666, 2px 6px 10px #999;  
  -webkit-box-shadow: 0px 4px 5px #666, 2px 6px 10px #999;  
  **box-shadow**: 0px 4px 5px #666, 2px 6px 10px #999;  
}

## 浏览器前缀使用混入

浏览器前缀使用混入也是非常方便的，如下实例：

## 实例

@mixin transform($property) {  
  -webkit-transform: $property;  
  -ms-transform: $property;  
  transform: $property;  
}  
  
.myBox {  
  @include transform(rotate(20deg));  
}

将以上代码转换为 CSS 代码，如下所示：

## Css 代码：

.myBox {  
  -webkit-transform: rotate(20deg);  
  -ms-transform: rotate(20deg);  
  transform: rotate(20deg);  
}

# Sass @extend 与 继承

@extend 指令告诉 Sass 一个选择器的样式从另一选择器继承。

如果一个样式与另外一个样式几乎相同，只有少量的区别，则使用 @extend 就显得很有用。

以下 Sass 实例中，我们创建了一个基本的按钮样式 .button-basic，接着我们定义了两个按钮样式 .button-report 与 .button-submit，它们都继承了 .button-basic ，它们主要区别在于背景颜色与字体颜色，其他的样式都是一样的。

## Sass 代码：

.button-basic  {  
  **border**: none;  
  **padding**: 15px 30px;  
  **text-align**: center;  
  **font-size**: 16px;  
  **cursor**: pointer;  
}  
  
.button-report  {  
  @extend .button-basic;  
  **background-color**: red;  
}  
  
.button-submit  {  
  @extend .button-basic;  
  **background-color**: green;  
  **color**: white;  
}

将以上代码转换为 CSS 代码，如下所示：

## Css 代码：

.button-basic, .button-report, .button-submit {  
  **border**: none;  
  **padding**: 15px 30px;  
  **text-align**: center;  
  **font-size**: 16px;  
  **cursor**: pointer;  
}  
  
.button-report  {  
  **background-color**: red;  
}  
  
.button-submit  {  
  **background-color**: green;  
  **color**: white;  
}

使用 @extend 后，我们在 HTML 按钮标签中就不需要指定多个类 class="button-basic button-report" ，只需要设置 class="button-report" 类就好了。

@extend 很好的体现了代码的复用。

# Sass 函数

Sass 定义了各种类型的函数，这些函数我们可以通过 CSS 语句直接调用。

|  |  |
| --- | --- |
| **序号** | **函数类别** |
| 1 | [Sass 字符串相关函数](https://www.runoob.com/sass/sass-string-func.html) |
| 2 | [Sass 数字相关函数](https://www.runoob.com/sass/sass-numeric-func.html) |
| 3 | [Sass 列表(List)相关函数](https://www.runoob.com/sass/sass-list-func.html) |
| 4 | [Sass 映射(Map)相关函数](https://www.runoob.com/sass/sass-map-func.html) |
| 5 | [Sass 选择器相关函数](https://www.runoob.com/sass/sass-selector-func.html) |
| 6 | [Sass Introspection 相关函数](https://www.runoob.com/sass/sass-introspection-func.html) |
| 7 | [Sass 颜色相关函数](https://www.runoob.com/sass/sass-color-func.html) |

# 更多，参考：[Sass教程 Sass中文文档 | Sass中文网](https://www.sass.hk/docs/)