sass 基础知识

一、sass基概述

1.sass的概念

sass是一种css“预处理器”，css预处理器是用一种专门的编程语言，为css增加了编程的特性，通过css预处理器编辑成正常的css使用。比如给css增加变量，按照dom的方式嵌套编写，增加函数等原本css不具备的特性，但是最后还是需要把这个sass文件编转成css文件，让css编写的更快。

(1)预处理简介

- sass是一种CSS“预处理器”

- CSS预处理器是用一种专门的编程语言，为css增加了编程的特性

- 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。支持嵌套、变量和逻辑等。可以让CSS更加简洁、提高代码复用性、逻辑分明等等

- 常见的css预处理器，sass，less,stylus

(2)sass和scss的关系

sass是和html一样有严格的缩进风格，和css编写规范有着很大的出入，是不使用花括号和分号的，所以不被广为接受。

- sass和scss其实是一样的css预处理语言，其后缀名是分别为 .sass和.scss两种。SASS版本3.0之前的后缀名为.sass，而版本3.0之后的后缀名.scss。

- 两者是有不同的，继sass之后scss的编写规范基本和css一致，sass时代是有严格的缩进规范并且没有‘{}’和‘；’。而scss则和css的规范是一致的。所以我们选择使用scss

3.sass的安装

查看安装视频

（1）使用npm安装sass

- 查看是否安装npm，打开`cmd`,输入`npm -v`

- 打开`cmd`,输入`npm install -g sass`，安装成功后检查`sass --version`

- 如果上面的方法因为网的问题没有安装成功，可以使用淘宝镜像

- 打开`cmd`,输入`npm install -g cnpm --registry=https://registry.npm.taobao.org`

- 再使用cnpm安装sass` cnpm install sass`

（2）依赖ruby安装sass

- 依赖ruby

- 下载rudy，下载地址` https://share.weiyun.com/4zMfauyw`里面的ruby软件,

- 根据自己电脑的系统选择x86（32位）、x64（64位）

- 查看ruby是否安装成功，打开`cmd`,输入`ruby -v`

- 安装sass

- mac：`sudo gem install sass`

- windows:`gem install sass`

- 安装完之后在`cmd`中检查 `scss -v`

4.文件编译和监听

- 使用终端或vscode终端在要编译的文件夹下打开，注意路径

- 在项目目录中创建css和scss文件目录，便于在scss目录中书写scss文件，使用编译方式将转化好的文件生成与css文件目录

- 文件夹监听`sass --watch scss:css`，可以使用简写`sass -w scss:css`

- 文件监听`sass --watch 1.scss:1.css`

- 任何文件夹不能出现中文字符

5.编译出现的错误【重点！！！】

编译中会出现很多错，比如编译之后没有任何效果，报错等，可参照以下方式改进

- sass项目的编译整个项目不允许有任何中文字符的文件夹，watch编译乱码：watch编译的路径不能有中文

- 不运行可能是你的安全助手或者防火墙阻止了ruby，注意弹窗，不要阻止

- 路径写错，注意路径的关系，应在scss和css文件的上层目录编译

- vscode编译报错，可使用windows自带的cmd进行编译

- powershell编译报错，在终端切换成command prompt 也就是cmd编译，或者直接使用使用windows自带的cmd进行编译

- sass编译乱码：

在\Ruby23-x64\lib\ruby\gems\2.3.0\gems\sass-3.4.22\lib\sass\engine.rb的所有require后加一句 Encoding.default\_external = Encoding.find('utf-8')

二、基本规则

1.注释规则

SASS共有两种注释风格。

- 标准的CSS注释 /\* 注释 \*/ ，会保留到编译后的文件。

- 单行注释 // 注释，只保留在SASS源文件中，编译后被省略

2.编码规则

在 Ruby 1.9 及以上环境中运行 Sass 时，Sass 对文件的编码格式比较敏感，首先会根据 CSS spec 判断样式文件的编码格式，因此，scss需要加入`@charset "UTF-8";`来保证编译过程编码方式，否则有可能在编译后的css中出现中文的乱码。

在scss中使用`@charset "UTF-8";`会报错的原因是使用的ruby安装，在\Ruby23-x64\lib\ruby\gems\2.3.0\gems\sass-3.4.22\lib\sass\engine.rb的所有require后加一句 Encoding.default\_external = Encoding.find('utf-8')可以尝试解决

文本

低可信度描述已自动生成

3.嵌套规则

- 按照html标签结构嵌套书写

- 将子元素的选择器和样式列表直接包裹在父元素的样式列表中

|  |
| --- |
| div {  width: 200px;  height: 200px;  background-color: red;  p {  color: white;  font-size: 20px;  a {text-decoration: none;}  }  } |

4.导入规则

- Sass 的 @import ，允许其导入 SCSS 或 Sass 文件。被导入的文件将合并编译到同一个 CSS 文件中。相当于将一个sass导入到另一个sass中用

- `@import "需要被引入的文件.scss";`

- 将@import指令写到需要引入的scss文件中不要使用url()，直接把路径写在双引号内。

5.可能产生的编译错误

在运算中，可能会出现多种出错情况，或者运行不生效的可能，原因是系统I/O调用顺序导致的。

输入输出I/O流可以看成对字节或者包装后的字节的读取就是拿出来放进去双路切换。

此刻的I/O调用的一瞬间的快慢会导致读取的顺序发生改变，因此，可能会出现报错，无法读取等情况。

三、scss语法

1.变量的使用

- `$变量名:值;`创建一个变量名

- 变量可以承接长度单位、值、颜色，以及其他变量

- 变量名尽量见名之意

- 可以创建局部变量，作用在{}以内的范围

- 注意全局变量的名称尽量不要和局部变量名称重复，以免出错

|  |
| --- |
| $myColor:#25aab2;  $myHeight:50px;  $blue:blue;  $bgColor:$blue;  div{  $jbwidth:200px;  width:$jbwidth;  height: $myHeight;  background-color: $myColor;  }  $bgColor:$blue;  div{  $jbwidth:200px;  width:$jbwidth;  height: $myHeight;  } |

2. 父选择器`&`

- 父选择器`&`的使用是代表当前作用域所有的结构集合

- 如果需要使用伪类，必须使用&进行占位

|  |
| --- |
| div {  width: 200px;  height: 200px;  background-color: red;  p {  color: white;  font-size: 20px;  a {  text-decoration: none;  }  &:hover {  color: blue;  }  }  }  <div>  <p>请访问<a href="#">这里</a></p>  </div> |

3. 计算功能

SASS允许在代码中使用算式，注意运算的单位前后尽量保持一致。

|  |
| --- |
| body {  margin: (14px/2);  top: 50px + 100px;  } |

4.插值语句

- 之前创建变量是用于样式中的值，但是选择器的名字不能使用变量，但可以使用插值#{}包裹变量后使用。

- 而且 使用 #{} 可以避免 Sass 运行运算表达式，比如/除号，有时候不是要除号而是需要斜杠。如`border-radius: #{$a}/#{$b};`使用插值 / 可以不看作除号

|  |
| --- |
| $name : button;  $myborder:border;  $x:10px;  $y:5px;  // 应用于 class 和属性  div>#{$name} {  #{$myborder }: 1px solid #ccc;  }  //结构性伪类选择器（在for循环中使用变量i）  div:nth-child(#{$i}){}  // 应用于复杂的属性值  div {  border-radius:#{$x}/#{$y};  } |

四、混合器指令

1. 定义混合指令 @mixin

声明混合指令就像创建一个@keyframes关键帧，需要一个名字，和一个定义样式的域{ }

- `@mixin`后添加名称与样式`@mixin name {样式列表}`

- 混合也需要包含选择器和属性，甚至可以用 & 引用父选择器

- 混合指令的名称可以自定义，但不要使用数字开头，按照css要求的命名标准

|  |
| --- |
| //自定义一个按钮的圆角和阴影，所有按钮都可以使用  @mixin btn-style {  border-radius: 5px;  box-shadow: 5px 5px 5px 0 #666;  } |

2. 引用混合样式 @include

- 使用 @include 指令引用混合样式，格式是在其后添加混合名称 `@include name;`

- 将`@include` 指令和调用混合名称直接放在需要的样式列表中

- 定义了混合指令就要用，需要在哪个地方用就使用 `@include`来引用`@mixin`的指令名

|  |
| --- |
| @mixin btn-style {  text-align: center;  border-radius: 5px;  box-shadow: 5px 5px 5px 0 #666;  }  .block-btn {  @include btn-style;  // 根据需要自定义的样式  width: 150px;  height: 50px;  line-height: 50px;  font-size: 20px;  color: #fff;  background-color: rgb(18, 182, 247);  } |

3. 传递参数

- 可以传递参数（可选）`@include name(30px);`也可以配合变量使用`@include name($变量);`

- 接收形式参数需要`@mixin name($x){样式列表}`格式

|  |
| --- |
| @mixin btn-style($w, $h) {  border: 0;  border-radius: 5px;  box-shadow: 5px 5px 5px 0 #666;  background-color: red;  color: #fff;  width: $w;  height: $h;  }  .a-btn {@include btn-style(150px,50px);}  .b-btn {@include btn-style(120px,40px);} |

【练习】

> 要求：设计一个元素居中的混合指令，只要应用的元素都可以上下左右居中在父元素中

> 使用子元素定位，margin负值需要用到子元素的宽度和高度的一半

> 宽度和高度是每个元素都不同的自定

|  |
| --- |
| @mixin box-center($w,$h) {  width: $w;  height: $h;  position: absolute;  left: 50%;  top: 50%;  margin-top: -($h)/2;  margin-left: -($w)/2;  }  .baba {  width: 400px;  height: 400px;  background-color: red;  & {position: relative;}  .erzi {  @include box-center(100px,200px);  background-color: yellow;  }  } |

五、继承指令

在设计网页的时候常常遇到这种情况：一个元素使用的样式与另一个元素完全相同，但又添加了额外的样式。通常会在 HTML 中给元素定义两个 class，一个通用样式，一个特殊样式。在sass中可以使用继承，来完成通用样式

1.@extend

- 在样式列表中使用`@extend 需要继承的选择器`

- 延伸复杂的选择器,允许延伸任何的选择器,如:hover`@extend .item:hover;`

- 多重延伸 `@extend .btn1;@extend .text;`,或者用逗号分隔`@extend .btn1， .text;`

|  |
| --- |
| .btn {  width: 150px;  height: 50px;  border-radius: 10px;  }  .btn1 {  background-color: red;  @extend .btn;  }  .btn2 {  box-shadow: 5px 5px 5px black;  @extend .btn1;  } |

多重延展

|  |
| --- |
| .btn {  width: 150px;  height: 50px;  border-radius: 10px;  }  .text {  font-size: 30px;  color: #fff;  }  .btn1 {  background-color: red;  @extend .btn;  }  .btn2 {  box-shadow: 5px 5px 5px black;  @extend .btn1，.text;  } |

2.占位符选择器 %

Sass 额外提供了一种特殊类型的选择器：占位符选择器，与id,class 选择器写法相似，只是 # 或 . 替换成了 %。但必须通过 @extend 指令调用

- 占位符选择器`%name`与id,class 选择器写法相似，只是 # 或 . 替换成了 %

- 必须通过 @extend 指令调用`@extend %name;`

- 占位符选择器,在编译后的css文件中不会出现

|  |
| --- |
| %aBtn{  font-size: 20px;  text-align: center;  color: #fff;  border-radius: 5px;  box-shadow: 3px 3px 5px 0 #666;  }  .btn {  width: 200px;  height: 50px;  background-color: red;  @extend %aBtn;  } |

六、高级语法

1.条件语句

- @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

- @if 声明后面可以跟多个 @else if 声明，或者一个 @else 声明

- 判断的条件

- $a == 10px;， $a != 10px;，$a > 10px; , $a <= 10px;

- 多个条件，并且 $a > 10px and $a < 30px

|  |
| --- |
| $w: 300px;  $h: 100px;  .box {  width: $w;  height: $h;  background-color: red;  @if $w==100px {  border: 10px solid black;  } @else if $w==200px {  border: 10px dotted black;  }@else {  border: 10px double black;  };  } |

2.循环语句

- @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

- 方式1：@for $i from {开始值} through {结束值}（through 包含结束值，to不包含结束值）

- 方式2：@for $i from {开始值} to {结束值}- `@for $var from 起始值 to结束值`

- 以下案例将i转换为可以使用的数字，必须使用`#{$i}`插值语句

|  |
| --- |
| @for $i from 1 through 6 {  .list2>li:nth-child(#{$i}) { background-color: #333\*$i; }  } |

七、函数

1.内置函数

（1）字符串函数

字符串函数顾名思意是用来处理字符串的函数。Sass 的字符串函数主要包括两个函数：

unquote($string)：删除字符串中的引号；

quote($string)：给字符串添加引号。

|  |
| --- |
| .demo1 { content: unquote('Hello Sass') ; }//Hello Sass  .demo2 { content: unquote("Hello Sass"); }//Hello Sass  .demo3 { content: quote('Hello Sass');}//"Hello Sass"  .demo4 { content: quote(HelloSass);}//"HelloSass" |

（2）列表函数

列表函数主要包括一些对列表参数的函数使用，其中常用的一种是：

length($list)：返回一个列表的长度值；函数主要用来返回一个列表中有几个值，简单点说就是返回列表清单中有多少个值。函数中的列表参数之间使用空格隔开，不能使用逗号，否则函数将会出错。

|  |
| --- |
| div:before{  content:” length(10px)”  }  length(10px)  结果:1  length(10px 20px (border 1px solid) 2em)  结果:4  length(border 1px solid)  结果:3 |

（3）数字函数

- 四舍五入`round()`将数值四舍五入，转换成一个最接近的整数，两边单位不统一会报错

- 向上取整`ceil()` 带单位px取整结果，单位被去掉

- 向下取整`floor ()` 带单位px取整结果，单位被去掉

- 随机数`random($limit: 100)`获取多少范围的整数，不写数字就是0-1之间的数字，看效果刷新页面无效，需要改变sass，才能重新编译

（4）颜色函数

- 在scss中写的rgb(255,10,210)会转为十六禁止，将十六进制放进rgba在加一个透明度会变味rgba颜色

- 使用`rgb()`将颜色转为十六进制

- 使用`rgba()`将十六进制转为`rgba()`颜色

|  |
| --- |
| //在scss中写的rgb(255,10,210)会转为十六禁止，将十六进制放进rgba在加一个透明度会变味rgba颜色  .color {  background-color: rgb(255,10,210);  color: rgba(#f9a4a2,0.4);  }  //在css文件中  .color {  background-color: #ff0ad2;  color: rgba(249, 164, 162, 0.4); } |

2.自定义函数

- @function，也可以传递若干个全局变量给函数作为参数。

- 一个函数可以含有多条语句，需要调用 @return 输出结果

- 在自定义函数前添加前缀避免命名冲突

|  |
| --- |
| //全局变量  $o-width:300px;  @function my-width($o){  //局部变量  $j-width:100px;  //返回值  @return ($o - $j-width);  }  //使用指令的函数  @function shadow($x...){  @if length($x) >=1 {  @return $x;  }  @else {  @return (10px 10px 10px 0 #666);  }  }  .box {  width: my-width($o-width);  height: 200px;  background-color: black;  //没传参数，用默认效果  box-shadow: shadow();  //传参数就有自己的效果  box-shadow: shadow(30px 30px 10px 0 green);  } |