**SASS基础知识汇总**

**一.sass定义:**

是一款强化css的辅助工具

在css的语法上，增加了变量，嵌套，混合，导入，函数等高级功能。这些拓展令css更加强大与优雅。sass运行在服务器端使用，需要通过node.js解析器后,转换成.css才能被浏览器解析执行。

**二.sass安装**

**步骤:**

(1)安装Node

(2)运行命令 npm install -g node-sass,安装node-sass包

(3)查看是否安装成功 node-sass -v查看版本号。

**三.将scss转换为css**

**情况1：手动将scss文件夹下的1个scss文件转换为css文件夹下的1个css**文件

命令:node-sass scss/1.scss css/1.css

Note：scss/1.scss和css/1.css之间有空格隔开

**情况2:手动将scss文件夹下的所有scss文件转化为css下的所有css文件**

命令:node-sass scss -o css

**情况3：自动监听scss文件夹下的一个scss文件，只要scss文件一旦改变，则css下对应的css文件就自动改变。**

命令:node-sass -w scss/1.scss css/1.css

**情况4：自动监听scss文件夹下的所有scss文件，只要scss文件夹下的任意.scss文件发生改变，点击保存后，自动更新css文件夹下的内容。**

命令: node-sass -w scss -o css

**四.SCSS的基础语法**

**1.变量**

**(1)定义规则:scss中的变量**使用$开头，可以包含\_ -,不能以数字开头，见名知意，例如$background; $left-wrap等

**(2)注意事项:**

①声明变量时，变量值可以应用其它的变量

②变量定义在{}规则块外边，整个样式文件中都可以使用；如果定义在{}规则块内部，只能在当前规则块中使用；

③！default规则

如果此变量之前声明过，就用之前声明的值；如果此变量没有声明过就用当前声明的值。

**(3)例子**

// 注意事项2:变量定义在{}规则块外边，整个样式文件中都可以使用

$background: red;

$width: 100px;

$jd\_red: blue;

$jd\_red: yellow !default;

// 注意事项3:!default规则表示，如果前面已经定义了$js\_red，就用之前已经定义好的样式blue，如果之前没有定义就用现在定义好的yellow；

.d1 {

  //注意事项1: 声明变量时，变量值可以应用其它的变量;如果定义在{}规则块内部，只能在当前规则块中使用

  $height: $width;

  width: $width;

  height: $height;

  background: $background;

  border: 1px solid $jd\_red;

}

**2.嵌套**

**主要有三种常用的:**

(1)父子之间嵌套--使用{}

(2)同一个元素上的多类选择器--使用&.class2{}

(3)同一级元素上的伪类选择器--使用&:hover{}

例子:

.d1 {

  width: 200px;

  height: 200px;

  background: $d1\_background;

  position: relative;

  .d2 {

    width: 100px;

    height: 100px;

    background: blue;

    position: absolute;

    top: 50px;

    left: 50px;

    &:hover {

      background: yellow;

    }

    &.d3 {

      border: 3px solid yellow;

    }

  }

}

**3.在全局scss文件中导入局部scss文件(@import)**

**(1).局部scss文件**

**定义:**在SCSS中，以下划线开头,那么这个文件就被称为是局部scss文件，局部scss文件不会自动转化为css文件。需要引入到全局scss文件中，再由全局scss文件一起转化为css文件。

**特点:**

①每一个局部scss文件就是一个小模块;

②SCSS在做转换时,不会把局部scss文件转换成css文件;

③项目中会将局部scss文件全部导入到全局scss文件中,再由全局scss文件生成css文件

④并且一个局部文件可以被多个SCSS文件引用

**(2)在全局scss文件中引入局部scss文件**

**语法:@import "局部文件名"**，其中,局部文件名称省略下划线和后缀。@import和“局部文件名”中间用空格隔开.

例如:在全局文件中导入局部scss文件:\_functions.scss

语法:**@import "functions";**

**例子:定义局部scss文件 \_title.css;并在5.全局scss文件.scss中引入局部scss文件，经过自动转换后，查看结果。**

**答:**

**①创建局部scss文件:\_title.scss**

$title\_width: 100px;

$wrap\_height: 100px;

.title {

  width: $title\_width;

  height: $wrap\_height;

}

**②在全局scss文件.scss中引入**

@import "title";

**③将全局scss转换为css**

**Node-sass scss -o css,发现局部的scss没有独自转换，只有全局的scss才转换为css。**

.title {

  width: 100px;

  height: 100px; }

**4.混合器(@mixin)**

**注解:** **其实混合器可以同等与一个函数来进行学习**

**4-1.定义和调用**

**(1)定义混合器**

定义:把定义混合器当作定义函数，只不过函数中的function用@mixin代替;

表达式:@mixin 混合器名称(形参1，形参2，形参3){样式内容}

**(2)调用混合器**

 定义: 可以把其理解为函数的调用，在"混合器名"前面加了一个@include.

@include 混合器名称(实参1，实参2，实参3)

**4-2.混合器传递参数**

   混合器在传递参数时，可以参照函数中的参数解构进行理解。只不过，不能用对象语法进行传递，但是可以给形参指定默认值，并且可以实现实参形参一一对应。

**(1)定义混合器：**

@mixin 混合器名($形参1:默认值，$形参2:默认值){样式}

@mixin block\_style($width: 100px, $height: 100px, $background: green) {

  width: $width;

  height: $height;

  background: $background;

  margin-top: 10px;

  border-radius: 20%;

}

**(2)调用混合器：**

@include 混合器名($形参1：实参1)

.d4 {

    //   调用混合器时，只要传入想要修改的值即可。

    @include block\_style($background: yellow);

  }

**例子:在一个父元素中创建4个形状相同，但颜色不同的长方形。**

@mixin block\_style($width: 100px, $height: 100px, $background: green) {

  width: $width;

  height: $height;

  background: $background;

  margin-top: 10px;

  border-radius: 20%;

}

.parent {

  width: 230px;

  //   float: left;

  display: flex;

  justify-content: space-around;

  align-items: center;

  flex-flow: row wrap;

  .d1 {

    //   调用混合器时，只要传入想要修改的值即可。

    @include block\_style($background: red);

  }

  .d2 {

    //   调用混合器时，只要传入想要修改的值即可。

    @include block\_style($background: blue);

  }

  .d3 {

    //   调用混合器时，只要传入想要修改的值即可。

    @include block\_style($background: green);

  }

  .d4 {

    //   调用混合器时，只要传入想要修改的值即可。

    @include block\_style($background: yellow);

  }

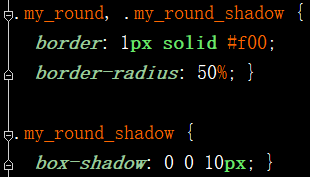
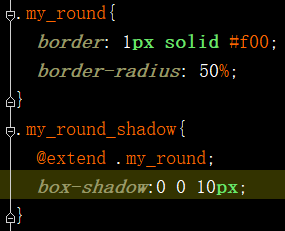
}

**5.继承（@extend）**

继承就是说一个选择器可以继承另一个选择器定义的所有样式

在css中的表现形式是两个选择器共有的部分，变成了群组选择器

**SCSS CSS**



练习

|  |
| --- |
| 在选择器.my-border中定义1px solid transparent  宽度400，距离左边有20px。  在选择器.my-bottom-border中继承.my-border  设置底边框为红色，鼠标悬停时底边框为蓝色 |

三.运算

1普通运算符的运算(,运算包括+ - x \ %,其中x和%没有问题)

(1)加

|  |
| --- |
| **作用:**  **①**会在不同单位间转换值  **width**:1**in**+8**pt**;  ②可以用于字符串的拼接  **注意:**  **①**scss不能转换相对单位  **height**:1**rem**+1**em**;转换失败  **②(字符串拼接规则**:**引号与非引号连接,谁在前结果为谁.**)  如果用引号去连接无引号的字符串，结果是有引号的  如果用无引号去连接有引号的字符串，结果是无引号  Eg:  **p**::**before**{  **content**:**"Microsoft"**+**yahei**;  **font-family**: **A**+**"rial"**; }  结果  **content**: **"Microsoftyahei"**; **font-family**: **Arial**; |

(2)减法

|  |
| --- |
| - 会被优先解析为变量名，所以使用变量和减法，需要-前后添加空格  **width**: **$size** - **$my-width** |

(3)除法(/)

|  |
| --- |
| **/作用:**①分隔符②除法  ①分隔符  **p**{ **font**:10**px**/5**px**;}  ②除法  在以下情况视为除法运算  1.如果运算符两边至少有一部分是变量或者函数的返回值  2.如果值被小括号包裹，视为除法  3.除法运算式是其他算术表达式的一部分，视为除法  **p**{ **$width**:1000**px**;  **width**: **$width**/2;  **height**: (500**px**/2);  **margin**:5**px**+8**px**/2**px**;} |

(4)运算表达式与其他值连用时，用空格做连接

|  |
| --- |
| margin:4px+5px auto; |

(5)在有引号的字符串中，使用**#{}插值语句**可以添加动态的值

|  |
| --- |
| **content**: **"I ate** #{16+23} **baozis"**; |

2.颜色的运算

|  |
| --- |
| 颜色值是分段计算的，红+红 绿+绿 蓝+蓝  #rrggbb+#rrggbb=#rr+rr gg+gg bb+bb  rgb(a,b,c)+rgb(d,e,d)=rgb(a+d, b+e, c+d)  rgba(11,22,33,0.1)+rgba(22,33,44,0.1)=rgba(33,55,77,0.1)  注解:两个rgba相加，alpha的值，必须相等才可以计算. |

四.函数

1.scss定义了多种函数，有些函数甚至直接可以在css中使用

|  |
| --- |
| 1.颜色函数  rgba(red,green,blue,alpha)  hsl(hue,saturation,lightness)  hue:色调 取值0~360 3个色段 每120一个色段  saturation：饱和度0.0%~100.0%  lightness:亮度 0.0%~100.0% |
| 2.数字函数  round($value) 四舍五入  ceil($value)向上取整  floor($value)  max($v1,$v2,......)  min($v1,$v2,......)  random() |
| 3.字符串函数  unquote($string) 删除字符串的引号  quote($string) 给字符串添加引号  To-upper-case(“abc”)把字符串转换成大写  To-lower-case(“ABC”)把字符串转换成小写 |

2.自定义函数

|  |
| --- |
| @function 函数名($n){  函数体；  @return 结果；  }  注意:Function前和return前用@开头,形参用$开头 |

四.控制指令

|  |
| --- |
| **@if** 1+1==4{**border-radius**:50%} **@else if**(1-1==-1){**border-radius**:30%} **@else** {**border-radius**:10%}  @if , @else if , @else  注解:相当于js中if(){}else if(){}else{}  不同点:①在关键字前面增加@;条件的小括号可以加也可以不加  boolean表达式，可以添加括号，也可以不加 |