sass 讲义

预习讲义如有错别字和输入错误，请及时与老师联系进行更改，谢谢！

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的关系：

1. 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。
2. 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

- sass老版本，scss新版本
- sass编写严格不被广泛接受
- scss与css规范一致，更容易接受

2.依赖ruby安装sass

- 依赖ruby

    - 下载rudy，下载地址 `https://share.weiyun.com/mG3HCLHR` 里面的软件
    - 查看自己电脑是32位还是64位，按照实际情况安装x86或者x64
    - 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` 或者按照提示 `scss -version` 出现版本号就可以了
- 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
    - 文件夹监听 `sass --watch scss:css`
    - 文件监听 `sass --watch 1.scss:1.css`
    - 可以使用简写 `sass -w scss:css`
    - 任何文件夹不能出现中文字符

3.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 -v`
- 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`
- 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
    - 文件夹监听 `sass --watch scss:css`
    - 文件监听 `sass --watch 1.scss:1.css`
    - 可以使用简写 `sass -w scss:css`
    - 任何文件夹不能出现中文字符

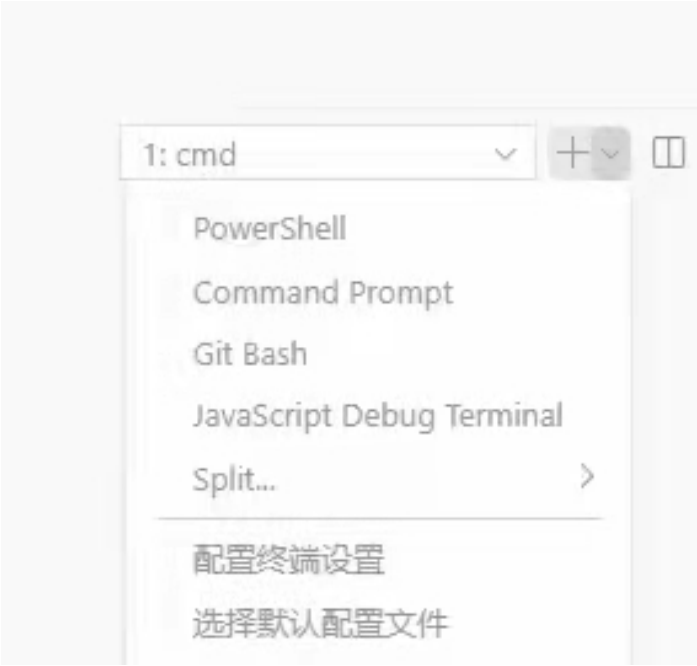
```
C:\Users\熊伟杰>x64
'x64' 不是内部或外部命令，也不是可运行的程序
或批处理文件。

C:\Users\熊伟杰>sass -v
Could not find an option or flag "-v".

Usage: sass <input.scss> [output.css]
       sass <input.scss>:<output.css> <input/>:<output/> <dir/>

=== Input and Output ==================
    --[no-]stdin               Read the stylesheet from stdin.
    --[no-]indented            Use the indented syntax for input from stdin.
-I, --load-path=<PATH>         A path to use when resolving imports.
                               May be passed multiple times.
-s, --style=<NAME>             Output style.
                               [expanded (default), compressed]
    --[no-]charset             Emit a @charset or BOM for CSS with non-ASCII characters.
                               (defaults to on)
    --[no-]error-css           When an error occurs, emit a stylesheet describing it.
                               Defaults to true when compiling to a file.
    --update                   Only compile out-of-date stylesheets.

=== Source Maps =======================
    --[no-]source-map          Whether to generate source maps.
                               (defaults to on)
    --source-map-urls          How to link from source maps to source files.
                               [relative (default), absolute]
    --[no-]embed-sources       Embed source file contents in source maps.
    --[no-]embed-source-map    Embed source map contents in CSS.

=== Other =============================
-w, --watch                    Watch stylesheets and recompile when they change.
    --[no-]poll                Manually check for changes rather than using a native watcher.
                               Only valid with --watch.
    --[no-]stop-on-error       Don't compile more files once an error is encountered.
-i, --interactive              Run an interactive SassScript shell.
-c, --[no-]color               Whether to use terminal colors for messages.
    --[no-]unicode             Whether to use Unicode characters for messages.
-q, --[no-]quiet               Don't print warnings.
    --[no-]trace               Print full Dart stack traces for exceptions.
-h, --help                     Print this usage information.
    --version                  Print the version of Dart Sass.

C:\Users\熊伟杰>sass --version
1.33.0 compiled with dart2js 2.13.0

C:\Users\熊伟杰>
```

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

- sass项目的编译整个项目不允许有任何中文字符的文件夹
- 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
- 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

```
PS D:\web\06CSS\DAY08> sass --watch scss:css
sass : 无法加载文件 C:\Users\web\AppData\Roaming\npm\sass.ps1，因为在此系统上禁止运行脚本。有关详细信息，请参阅 https:/go.
microsoft.com/fwlink/?LinkID=135170 中的 about_Execution_Policies。
所在位置 行:1 字符: 1
+ sass --watch scss:css
+ ~~~~
    + CategoryInfo          : SecurityError: (:) [], PSSecurityException
    + FullyQualifiedErrorId : UnauthorizedAccess
PS D:\web\06CSS\DAY08>
```



二、嵌套与书写

1.嵌套的规则

- 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;
    }
  }
}
```

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. @import导入方式

- 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。

```
//import1.scss 这是一个简单的css reset
* {margin: 0;padding: 0;}
a {text-decoration: none;}
//import2.scss
@import "./import1.scss";
a {
  color: red;
  font-size: 50px;
}
```

三、变量的使用

SassScript 最普遍的用法就是变量，变量以美元符号开头，赋值方法与 CSS 属性的写法一样

1. 创建变量

- `$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

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

2. 变量的调用

- 变量的调用，如：height: $myHeight;
- 变量也可以调用其他变量

```
$myHeight:50px;
$blue:blue;
div{
  width:200px;
  height: $myHeight;
  $blue:blue;
  $bgColor:$blue;
  background-color: $bgColor;
}
```

3.局部变量

- 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

```
$blue:blue;
$bgColor:$blue;
div{
  $jbwidth:200px;
  width:$jbwidth;
  height: $myHeight;
}
```

四、混合指令

混合指令（Mixin）用于定义可"重复使用的样式"，比如 .float-left。比如之前的兼容，浏览器的内核很多种一起写的时候可以使用混合指令写在一个混合指令里，完成所有的兼容，这样比较清晰

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;
}
```

清除浮动

```
@mixin clearfix {
  &::after {
    content: "\200B";
    display: block;
    height: 0;
    clear: both;
  }
  *zoom: 1;
}
.list {
  background-color: red;
  list-style: none;
  li {
    width: 100px;
    height: 100px;
    float: left;
    font-size: 20px;
    border: 1px solid #000;
    background-color: yellow;
  }
  & {
    @include clearfix;
  }
}
```

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

基本使用方式，写两个div当按钮

```
@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);
}
.a-btn {
  @include btn-style;
  // 根据需要自定义的样式
  display: block;
  text-decoration: none;
  width: 120px;
  height: 40px;
  line-height: 40px;
  font-size: 16px;
  color: #fff;
  background-color: rgb(18, 182, 247);
}
<div class="block-btn">用div做的按钮</div>
<a href="#" class="a-btn">用a做的按钮</a>
```

3. 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
- 参数变量 `@mixin name($x...){样式列表}`

传递参数

```scss
@mixin btn-style($w, $h) {
  //共有样式
  text-align: center;
  display: block;
  border-radius: 5px;
  box-shadow: 5px 5px 5px 0 #666;
  background-color: rgb(18, 182, 247);
  text-decoration: none;
  color: #fff;
  //特有样式
  width: $w;
  height: $h;
  line-height: $h;
}
.block-btn {@include btn-style(150px,50px);}
.a-btn {@include btn-style(120px,40px);}
```

有时，不能确定混合指令需要使用多少个参数，比如一个关于 box-shadow 的混合指令不能确定有多少个 'shadow' 会被用到。

```scss
@mixin mybox-shadow($s...){
  box-shadow: $s;
}
.box {
  width: 100px;
  height: 100px;
  background-color: yellow;
  // 增加删除参数的数量
  @include mybox-shadow(0 0 20px 5px green);
}
```

【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

```scss
@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;
  .erzi {
    @include box-center(100px,200px);
    background-color: yellow;
  }
  & {position: relative;}
}
```

五、继承

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;`
- 可以多重延伸 `@extend .btn1;@extend .text;`

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

其他选择器也可以继承比如hover

```
.item:hover {
  background-color: rgb(125, 186, 255);
}
.list{
  li:hover {
    @extend .item:hover;
  }
}
```

多重延伸

```
.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;
  @extend .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
使用

- 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

```scss
%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、运算

【注意】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。

1. 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

- 通过 `#{}` 插值语句可以在选择器或属性名中使用变量
- 使用 `#{}` 可以避免 Sass 运行运算表达式，直接编译 CSS,如 `border-radius: #{$a}/#{$b};` 使用插值 / 可以不看作除号

2. 数字运算

- Sass脚本支持数字的加减乘除、取整等运算 (+, -, *, /, %)
- 计算方式可以是值直接计算，如： `10px * 10 => 100px`
- 计算方式可以是变量计算，如： `width: $a+$b;`
- 加法遇到字符串会拼接，但字符串要在前半部分 `"acb"+20px`
- 运算符于值之间尽量加空格，以免被认为是变量的一部分
- 除法比较特殊，使用时需要谨慎
- 【注意】scss终究适用于样式的渲染，因为I/O流问题，尽量减少使用渲染样式的计算

```
$a: 100px;
$b: 50px;
$c: 20px;
.box {
  width: $a + $b;
  height: 10px * 10;
  margin-top: $a - $b;
  margin-left: $a/2;
   //变量和变量但除法会报错
  background-color: red;
}
```

3. 颜色值运算

颜色值的运算是分段计算进行的，也就是分别计算红色，绿色，以及蓝色的值

```
p {
  计算 01 + 04 = 05 02 + 05 = 07 03 + 06 = 09
  color: #010203 + #040506;
}
p {
  //01 * 2 = 02 02 * 2 = 04 03 * 2 = 06
  color: #010203 * 2;
}
p {
  // color: rgba(255, 255, 0, 0.75);
  color: rgba(255, 0, 0, 0.75) + rgba(0, 255, 0, 0.75);
}
```

七、控制指令

1.if

- @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
- 条件局限性很强，单个判断可以如:$a == 100px$,a>=200px

```
$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.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
- `@for $var from 起始值 through 结束值` ,起始值和结束值必须是整数值
- 此种方式的遍历索引区间是[start,end]
- 以下案例将i转换为可以使用的数字，必须使用 `#{$i}` 插值语句

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

八、函数

1.自定义函数

@function与@mixin非常相似，但这两者的第一点不同在于sass本身就有一些内置的函数，方便我们调用，如强大的color函数；其次就是它返回的是一个值，而不是一段css样式代码。

- 与 mixin 相同，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
- 自定义函数与 mixin 相同，都支持 variable arguments
- length($list) 返回一个列表的长度值，属于列表函数

```
//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);
}
```

2.字符串函数

(1)删除字符串中的引号

- 不管是双引号还是单引号包裹的字符串，引号皆被去掉；
- 只能删除字符串最前边和最后边的引号，没法去掉中间的引号；

```
.demo1 { content: unquote('Hello Sass') ; }//Hello Sass
.demo2 { content: unquote("Hello Sass"); }//Hello Sass
```

(2)给字符串添加引号

- 若字符串本身带有引号，就不添加；
- 若字符串带有单引号，则跟换为双引号；

```
.demo1 { content: quote('Hello Sass');}//"Hello Sass"
.demo2 { content: quote(HelloSass);}//"HelloSass"
```

3.数字函数

【注意】数字函数也存在计算问题，因此也存在系统I/O调用顺序问题。在计算时出现问题不要纠结。

(1)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
- 可以使用算式计算结果够取整，以及百分比取整
- 带单位px取整结果，单位被去掉

```
.demo {
  // 四舍五入，两边单位不统一会报错
  width: round(12.3px); //12px;
  height: round(2px / 3px);//1
  margin: round(2.2%);//2%
}
```

(2)向上取整

- ceil($value) 向上取整
- 可以使用算式计算结果够取整，以及百分比取整
- 带单位px取整结果，单位被去掉

```
.demo {
  width: ceil(2.3px);//3px;
  height: ceil(2px / 3px);//1
  margin: ceil(2.8%);//3%
}
```

(3)向下取整

- floor($value) 向下取整
- 可以使用算式计算结果够取整，以及百分比取整
- 带单位px取整结果，单位被去掉

```
.demo {
  width: floor(2.3px);//2px
  height: floor(2px / 3px);//0
  margin: floor(2.8%);//2%;
}
```

(4)最小值和最大值

- `min($numbers…)` 找出几个数值之间的最小值
- `max($numbers…)` 找出几个数值之间的最大值

```
.demo {
  width: max(100px,200px);//200px
  height: min(50px,100px); //50px
}
```

(5)随机数

- `random()` 获取随机数
- `random($limit: 100)` 获取多少范围的整数
- 随机数为无单位数字，若需要单位如：unquote(random($limit: 100)+"px"),或者直接+px进行拼接
- 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

(1)rgb和rgba

- 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)hsl【了解】

hsl和hsla在css3中也可以直接写，它代表饱和度，他分别代表，色相，饱和度，明度。
色相（H）是色彩的基本属性，就是平常所说的颜色名称，如红色、黄色等。
饱和度（S）是指色彩的纯度，越高色彩越纯，低则逐渐变灰，取0-100%的数值。
明度（V），亮度（L），取0-100%。

- `hsl(色相,饱和度,亮度)` 转为十六进制颜色

- H色相：取值为：0 - 360，0或360～120表示红色，120～240表示绿色，240～360表示蓝色
- S饱和度：取值为：0.0% - 100.0%
- L亮度：取值为：0.0% - 100.0%