1. css中的函数

1. css函数了解

1. 在前面我们使用过很多个CSS函数。

- 比如：rgb，rgba，translate，rotate，scale等。
- css函数通常可以帮助我们更加灵活的来编写样式值。

2. 下面我们再学习几个非常好用的CSS函数。

- var：使用CSS定义变量。
- calc：计算CSS的值，通常用于计算元素的大小或位置。
- blur：毛玻璃（高斯模糊）效果。
- grandient：颜色渐变函数。

2. var使用css变量

1. CSS中可以自定义属性，**属性名需要以两个减号（--）开始**。
2. 属性值则可以是任何有效的CSS值。
3. 案例：定义变量和使用变量的方式。

```
<style>
  html{
    /* 1. 在css中定义一个变量 */
    --main-color: #f40
  }
  .box{
    width: 100px;
    height: 100px;
    /* 2. 通过var函数来使用这个变量 */
    background-color: var(--main-color);
  }
  .title{
    color: var(--main-color);
  }
</style>

<body>
  <div class="box">
    我是一个盒子
  </div>
  <h3 class="title">我是个标题</h3>
</body
```

4. 定义变量的规则，也就是定义选择器的规则，是自定义属性的可见作用域（只在选择器内部有效）。

- 所以推荐将自定义属性定义在html中，也可以使用:root选择器（:root代表的就是html）。

5. **变量的优势：**

- 例如我们页面的主要颜色是某个颜色，那我们就可以将他定义成一个变量。
- 这样使用就会很方便。

2. less

01. css的弊端

> 维护CSS的弊端。CSS是一门非程序式语言，没有变量，函数，SCOPE（作用域）等概念。

1. CSS需要书写大量看似没有逻辑的代码，CSS冗余度是比较高的。
2. 不方便维护及扩展，不利于复用。
3. CSS没有很好的计算能力（不能运算）。
4. 对非前端开发工程师来讲，往往会因为缺少CSS编程经验而很难写出组织良好且易于维护的CSS代码项目。

02. less简介

1. less（Leaner Style Sheets的缩写）是一门CSS扩展语言，也成为CSS预处理器。作为CSS的一种形式的扩展，它并没有减少CSS的功能，而是在现有的CSS语法上，为CSS加入程序式语言的特性。
2. 它在CSS的语法基础之上，引入了变量，Mixin（混入），运算以及函数等功能，大大简化了CSS的编写，并且降低了CSS的维护成本，就像它的名称所说的那样，Less可以让我们用更少的代码做更多的事情。
3. less中文网址：http://lesscss.cn/
4. 常见的CSS预处理器：Sass，Less，Stylus
5. Less是一门CSS预处理语言，它扩展了CSS的动态特性。

03.【重点】less编译

方式一：下载Node环境，通过npm包管理下载less工具，使用less工具对代码进行编译。

> 在实际开发中常用的方式。

```
npm install less -g          // 安装到全局环境
npm install less@2.7.1 -g    // 全局安装指定版本安装
// 或者 这将在项目文件夹中安装 lessc 的最新正式版本，并将其添加到 package.json 文件中的
devDependencies 配置段中
npm i less --save-dev
```

方式二：通过VSCode插件来编译成CSS或者在线编译。

1. vscode中安装 less编译插件： Easy Less 插件。
2. 安装成功后重启vscode软件，新建点less后缀名文件（.less），保存后会自动生成对应.css文件。

3. 安装成功后 Easy Less 扩展设置中设置导出CSS的路径（不能让CSS和.less文件存放在一起，后续打包不方便）。

```
// settings.json文件
 "less.compile": {
    "out": "../css/"    // 设置css导出
  }
```

方式三：引入CDN的less编译代码，对less进行实时的处理。

```
<link rel="stylesheet/less" type="text/css" href="styles.less" />
<script src="https://cdn.jsdelivr.net/npm/less@4" ></script>
```

方式四：将less编译的js代码下载到本地，执行js代码对less进行编译。

1. 在官网里面下载，再进行引入。
2. 下载地址：https://cdn.jsdelivr.net/npm/less@4

注意：less是兼容css的，所以在less里面可以正常的书写css样式，只是将css的扩展名改成.less结尾而已。

04. less变量

1. less变量是指没有固定的值，可以改变的。因为CSS中的一些颜色和数值等经常使用。
2. 变量语法：@变量名：值；
3. **变量的命名规则：**

- 必须有@为前缀。
- 不能包含特殊字符。
- 不能以数字开头。
- 大小写敏感。

05. less嵌套

1. less嵌套语法：

```
/* less嵌套语法 */
#header{
  .logo{
    width:300px;
  }
}
```

2. css语法表现：

```
/* 选择器的嵌套 */
#header .logo{
  width:300px;
}
```

3. less语法嵌套：**如果遇见（交集|伪类|伪元素选择器）**

- 内层选择器的前面如果没有加 & 符号，则它被解析为复选择器的后代。
- 如果有 & 符号，它就被解析为父元素自身或者父元素的伪类。

```
/* 注意这个案例一定要注意，否则写的伪元素无效。 */
a:hover{
  color:red;
}

/* less嵌套中语法为 */
a{
  &:hover{
    color:red;
  }
}
```

06. less运算

1. 任何数字、颜色或者变量都可以参与运算。
2. less提供了加（+）、减（-）、乘（*）、除（/）算数运算符。

```
// 定义一个变量@width。后面参与加法运算加5.
@width:10px + 5;
div{
  border: @width solid red;
}

// 生成的css文件
div{
  border: 15px solid red;
}

// less甚至还可以这样写
width:(@width + 5) *2;

// 注意事项：在less中运算无效的时候，可以给运算添加一个括号。
```

3. 运算注意事项：

- 乘号（*）和除号（/）的写法。
- **运算符左右有个空格隔开 1px + 5。**
- **对两个不同单位的值之间的运算，运算结果的值取第一个值的单位。**
- **如果两个值之间只有一个值有单位，则运算结果就取该单位。**

```less
div{
  // 1. 这里运算一定要加括号，不加括号运算无效，2023年1月22日，我在使用的时候由于没有加括号，导致运算无效，检测半天才发现是括号的问题。
  // 2. 单位以px为准。
  // 3. 两个参数参与运算，如果只有一个数有单位，则最后的结果就是以这个单位为准的。
  width: (500 / 50px);
}

.box{
  // 1. 这里单位以rem单位为准。
  // 2. 两个数参与运算，如果二个数都有单位，而且单位不一样，最后的结果以第一个单位为准。
  width: (500rem / 50px)
}
```

07.【重点】less混入

> 混入又称混合（mixins）。

混入的基本使用：

1. 在原来的CSS编写过程中，多个选择器中可能会有大量相同的代码。
2. 我们希望可以将这些代码进行抽取到一个独立的地方，任何选择器都可以进行复用。
3. 在less中提供了混入（Mixins）来帮助我们完成这样的操作。

注意：混合（Mixin）将一组属性从一个规则集（或混入）到另一个规则集的方法。

```less
// 1.1.less 定义变量
// @widthMain:100px;
// @commonMargin:10px;

// 2.1.混入的基本使用
.nowrap_ellipsis{
  overflow: hidden;
  white-space: nowrap;
  text-overflow: ellipsis;
}

.box1{
  width: 100px;
  background-color: yellow;
  // 混入的基本使用
  .nowrap_ellipsis();
}

.box2{
  width: 50px;
  background-color: purple;
  // 混入的基本使用
  .nowrap_ellipsis();
```

```
}
```

混入带参使用：

1. 混入带参，可以定义默认参数。
2. 如果在使用的时候不传入参数则使用默认参数，否则就是用传入的参数。
3. **注意：混入在没有参数的情况下，小括号可以省略，但是不建议这样使用。**

```less
// 2.2.混入带参数（将边框的宽度和颜色都定义成变量）
.box-border(@width:5px,@color:red){
  border: @width solid @color;
}

.box1{
  width: 50px;
  background-color: yellow;
  .nowrap_ellipsis();
   // 混入带参使用，如果不传入参数则使用默认参数。
  .box-border();
}

.box2{
  width: 50px;
  height: 100px;
  background-color: purple;
  .nowrap_ellipsis();
   // 混入带参使用，传入参数
  .box-border(20px,blue);
}
```

08.【重点】less映射

> 映射（Maps）一般和混入一起使用。

```less
// 3.1.混入和映射（Map）结合使用
.box-size{
  width:200px;
  height:200px
}

.box1{
  // 映射的使用，类似于对象
  width: .box-size()[width];
  background-color: yellow;
  .nowrap_ellipsis();
   // 混入带参使用
  .box-border();
}
```

```
.box2{
  width: 50px;
  // 映射的使用，类似于对象
  height: .box-size()[height];
  background-color: purple;
  .nowrap_ellipsis();
   // 混入带参使用
  .box-border(20px,blue);
}
```

> **注意：映射和混入的结合：混入也可以当作一个自定义函数来使用。**

```
// 4. 映射和混入的结合：混入也可以当作一个自定义函数来使用。
@htmlFontSize:50px;
.pxToRem(@px){
  result:(@px / @htmlFontSize) * 1rem;
}

.box3{
  width: .pxToRem(200px)[result];
  height: 100px;
  background-color: yellowgreen;
}
```

09. less继承

1. 继承：**使用关键字 extend**。
2. 和混入（mixins）作用类似,于复用代码;
3. 和混入（mixins）相比,**继承代码最终会转化成并集选择器**;

```
// 5. less继承
.boxStyle{
  border: 10px solid red;
}
.box4{
  width: 100px;
  height: 100px;
  background-color: orange;
  // 注意：继承使用
  &:extend(.boxStyle);
}
```

10. less内置函数

1. less内置了多种函数用于转换颜色、处理字符串、算术运算等。
2. 内置函数手册: https://less bootcss .com/functions/
3. **内置函数使用得比较少。**

11.【重点】less作用域

1. less作用域: spore。
2. 在查找一个变量时， 首先在本地查找变量和混合(mixins) ;
3. 如果找不到，则从父级作用域继承。

```
// 6. 作用域 scope
.box6{
  width: 100px;
  height: 100px;
  background-color: plum;
  @bc:blue;
  // box6里面相当于是父级作用域。
  p{
    // p里面相当于是自己的作用域。
    // 注意：还有一个比较重要的，在自己作用域内，即便是将@bc定义在background-color这个属性后面，也无妨。
    // @bc:red;
    width: 50px;
    height: 50px;
    background-color: @bc;
    @bc:red;
  }
}
```

12. less注释

> 注释: commoents

1. 单行注释: // 注释内容
2. 多行注释: /*注释内容*/

```
/* 一个块注释
 * style comment! */
@var: red;

// 这一行被注释掉了!
@var: white;
```

13. less导入

> 导入：importing。

1. 导入的方式和CSS的用法是一致的。
2. **导入语法：@import '文件名'**。
3. 导入一个.less文件,此文件中的所有变量就可以全部使用了。
4. 如果导入的文件是.less 扩展名,则可以将扩展名省略掉。

```
// 引入 library.less
@import "library";
```