CSS预处理器 - Less

Less简介

什么是Less?

`Less`是一种编程语言，通过`Less`编写类似于`css`的代码，最后编译成正常的`CSS`文件，给项目使用。

Less是一门`CSS`预处理语言，功能上比`CSS`更强大，在`CSS`的基础上增加了**变量、混合、循环、函数等**功能，让`CSS`更易维护、方便制作主题、扩充。

注：**不能直接在浏览器引入使用，需要编译成`CSS`**

Less的优点

1. Less能够使用嵌套编写使结构更清晰
2. Less可以轻松的生成在工作中使用的`CSS`，使代码更简洁
3. 通过使用变量可以更快的实现维护
4. Less提供使用操作，使得编码更快且更容易修改
    1. 不需要考虑前缀

Less的安装及编译

1、通过npm命令行安装Less

1）以管理员身份打开命令提示符，运行前查看是否安装了npm（npm -v），如果没有安装node.js包即可，下载node：http://nodejs.cn/download/

```
npm install -g Less
```

2）安装 Less 后，就可以在命令行上调用 Less 编译器了

```
Lessc styles.Less
```

3）这将输出编译之后的 CSS 代码到`stdout`，你可以将输出重定向到一个文件

```
Lessc styles.Less > styles.css
```

2、编辑器中Less插件的安装

vsCode中，安装easy Less插件即可

3、安装koala

下载 koala： http://koala-app.com/index-zh.html

> `koala`是现在市面上，最好用的 CSS预处理语言 图形编译工具，可以变多多种 CSS预处理语言，例如：Less、sass、compass

优点：

- 支持中文（修改后一定要重启一下）
- 有图形界面
- 可以很方便的修改输出路径
- 修改压缩方式方便
- 可以设置编译选项
  - source Map 生成源文件隐射
  - line Comments 行注释 （当前css在Less中的位置）
  - debug Info 调试信息
  - strict Math 严格运算模式，数学运算必须加上`()`
  - strict Units 严格单位模式
  - autoprefix 自动添加前缀

> 其他使用方法，参见官网：http://Lesscss.org/usage/

Less语法

1、变量

在单个位置控制常用值，存储数据的容器

> 当一个人名字比较长的时候
>
> 例如：巴勃罗.迭戈.荷瑟.山迪亚哥.弗朗西斯科.德.保拉.居安.尼波莫切诺.克瑞斯皮尼亚诺.德.罗斯.瑞米迪欧斯.西波瑞亚诺.德.拉.山迪西玛.特立尼达.玛利亚.帕里西奥.克里托.瑞兹.布拉斯科.毕加索
>
> 简写：毕加索

语法：

@变量名：变量值;

@毕加索 : 巴勃罗.迭戈.荷瑟.山迪亚哥.弗朗西斯科.德.保拉.居安.尼波莫切诺.克瑞斯皮尼亚诺.德.罗斯.瑞米迪欧斯.西波瑞亚诺.德.拉.山迪西玛.特立尼达.玛利亚.帕里西奥.克里托.瑞兹.布拉斯科.毕加索

通过`@`声明变量。

使用变量时，直接用变量名 替代 变量值：

```
@w:100px;

.box{
    width:@w;
}
------------------------
.box {
  width: 100px;
}
```

案例：设置一个宽高永远一致的按钮

```
@w:200px;

.btn{
    width: @w;
    height: @w;
    border-radius: 50%;
    border: 1px solid gray;
    background:#ccc;
    text-align:center;
    line-height:@w;
}
-------------------------
.btn {
  width: 200px;
  height: 200px;
  border-radius: 50%;
  border: 1px solid gray;
  background:#ccc;
  text-align:center;
  line-height:200px;
}
```

更多的变量使用方案

- 属性值
- 插值
    - 针对于属性值的，例如：图片引入地址

```
@imgurl:"../images";
```

```
.box1{
    background: url('@{imgurl}/1.jpg');
}
.box2{
    background: url('@{imgurl}/2.jpg');
}
.box3{
    background: url('@{imgurl}/3.jpg');
}
-----------------------------------------
.box1 {
  background: url('../images/1.jpg');
}
.box2 {
  background: url('../images/2.jpg');
}
.box3 {
  background: url('../images/3.jpg');
}
```

> 作为插入到一部分内容的变量，要包含在`{}`

- 针对选择器的

```
@primary:primary;

.el-button-@{primary}{
    width: 100px;
}
------------------
.el-button-primary {
  width: 100px;
}
```

- 针对属性名的

```
@bgColor:background-color;

.box{
    width: 200px;
    height: 200px;
    @{bgColor}:red;
}
---------------------
.box {
  width: 200px;
  height: 200px;
  background-color: red;
}
```

- 用变量定义变量

```
@a:"I am a teacher";
@b:"a";

div::before{
    content: @@b;
}
----------------------
div::before {
  content: "I am a teacher";
}
```

  - `@a` - `"I am a teacher"`
  - `@b` - `"a"`

变量使用注意事项

1. 变量的声明**不是**必须在使用前

```
div{
    width: @w;
}
@w:200px;
-----------------------
div {
  width: 200px;
}
```

2. 变量名同作用域下不能重复，否则会被更改掉

```
@w:200px;
@w:300px;
div{
    width: @w;
}
-----------------------
div {
  width: 300px;
}
```

2、作用域

Less中，是有作用域的，作用域会影响当前的变量，在什么范围内是有效的，是否会替换

```
1. @w:300px;
2. .box1{
3.     @w:200px;
4.     width:@w;
5. }
6. .box2{
7.     width:@w;
8. }
9. ---------------------------
10. .box1 {
11.  width: 200px;
  }
12. .box2 {
13.   width: 300px;
14. }
```

这里的第`1`行 和 第`3`行，声明了同一个变量`@w`，但是可以看到，下面的`box1`的值和`box2`的值是不一样的，并没有像上面我们说的那样发生替换，原因就是，这里的作用域不同

红色是全局作用域范围，紫色是局部作用域范围，变量只在当前画出来的作用域内有效

当局部作用局找不到变量，会向外查找，一直到全局作用域，**如果全局作用域也没有，编译会报错**

同一个作用域的时候，以最后一次出现为准

```
.box1{
    @w:200px;
    width:@w;
    @w:300px;
    @w:400px;
}
--------------------
.box1 {
  width: 400px;
}
```

3、注释

- 单行注释

  ```
  //这是单行注释
  ```

  这个注释的内容，不会被解析到`css`文件内

- 多行注释

  ```
  /*
    这是多行注释
  */
  ```

  这个注释内容，会被解析到`css`文件内

4、嵌套

类似于HTML的结构，通过嵌套，来表现父子级关系，不建议嵌套过多层（包含选择器：不推荐超过三层）

```
.box{
    .nav{
        border: 1px solid #000;
    }
}
--------------------------------------
.box .nav {
  border: 1px solid #000;
}
```

5、父选择器

```
.clearfix{
    &::after{
        content: '';
        display: block;
        clear: both;
    }
}
-----------------------------------
.clearfix::after {
  content: '';
  display: block;
  clear: both;
}
```

`&`表示父级的意思

```
.box{
    .warp &{
        width: 200px;
    }
}
-----------------------------------
.warp .box {
  width: 200px;
}
```

6、深入嵌套- 冒泡

在页面的任何位置，我们都可以放心的写一些指令，例如`@font-face`

```
div{
    font:50px / 100px "fonts";
    @font-face {
        font-family: 'fonts';
        src: url('../font/font.ttf');
    }
}
-----------------------------------------------------
div {
  font: 50px / 100px "fonts";
}

@font-face {
  font-family: 'fonts';
  src: url('../font/font.ttf');
}
```

这里的`@font-face`最终解析的时候，不会被设置到div内，而是会分离出来。对于样式多的时候，很棒~ 因为我们不用特意去找最外层写。

7、深入嵌套- 命名空间

```
.box1{
    background: red;
    .fonts{
        font-size: 30px;
    }
}
.box2{
    .box1 .fonts;
}
---------------------------------------
.box1 {
  background: red;
}
.box1 .fonts {
  font-size: 30px;
}
.box2 {
  font-size: 30px;
}
```

这一定要指明 xx 下面的`.fonts`,因为作用域不一样，不可以直接`.fonts`

8、运算

`css`预处理器的强大，不止在于变量，还有超级好用的运算。

案例：在不使用怪异盒模型的情况下，设置一个padding为`50`，可视宽高为`800 * 800`的盒子

```
.box{
    width:800px - 50 * 2;
    height:800px - 50 * 2;
    padding:50px;
}
---------------------------
.box {
  width: 700px;
  height: 700px;
  padding: 50px;
}
```

- 运算符使用注意事项(加减乘除)
  - 运算符的左右，都保留空格
  - 运算的时候，最好包裹在`()`内
    - font 复合样式容易有问题哦

```
font: 100px - 20px "微软雅黑";
---------------------------------------
font: 100px - 20px "微软雅黑";
```

这里的内容被认为是不可以做减法的，所以没有进行计算

  - 当运算数值具有单位时，需要注意：
    - 一个单位：最终结果根据唯一的这个单位
    - 两个单位：看运算符，`*`除了px的其他单位，`+ - /`为第一个单位
- 有哪些内容可以参与运算
  - 数值
  - 变量

```
@wa:100px;
.box{
    width: @wa - 20px;
    height: (@wa - 10px) * 2;
    border:(@wa - 91px) / 3 solid red;
}
-----------------------------
.box {
  width: 80px;
  height:180px;
    border: 3px solid #ff0000;
}
```

  - 颜色值

```
@c:#006622;
.box{
    width: @c / 2;
}
-----------------------------
.box {
  width: #003311;
}
```

9、忽略内容

有些样式中，本身就带有/，而且并不是用于计算的，例如font：

```
.box{
    font: (100px - 20px) ~'/' (200px - 20px) "微软雅黑";
}
-----------------------------------------------------
.box {
  font: 80px / 180px "微软雅黑";
}
```

> 除了font，background的复合样式也经常会用到

10、混合

混合是个更好用的东西，就是把另一个东西，混合进来

- 无参数混合

```
.box{
    height: 200px;
    background: red;
}
.content{
    .box;//也可以是 .box()
}
-----------------------------------------
.box {
  height: 200px;
  background: red;
}
.content {
  height: 200px;
  background: red;
}
```

混合分为两种形式，一种叫无参数，就是上面这种，另一种叫传参混合，无参数混合的时候，可以混合某个子级样式

```
.box{
    height: 200px;
    width: 200px;
    .a{
        color: red;
    }
}
.box1{
    .box .a;
```

```
}
---------------------------------
.box {
  height: 200px;
  width: 200px;
}
.box .a {
  color: red;
}
.box1 {
  color: red;
}
```

- 传参混合（数值方面更加自由）

```
.box(@a,@b){
    height: @a;
    background: @b;
}
.content{
    .box(200px,red);
}
-------------------------
.content {
  height: 200px;
  background: #ff0000;
}
```

这种方式，对于数值方面更加自由，但是也有缺点，加了`()`的那个样式，会发现在css中没有出现

而且，如果使用的时候，不传入参数，会编译失败，这时候我们可以设置默认值

- 默认值设置

```
.box(@a:200px,@b:red){
    height: @a;
    background: @b;
}
.content{
    .box;
}
---------------------------------
.content {
  height: 200px;
  background: #ff0000;
}
```

- arguments

当你发现，所有的传入参数都是加给一个样式的时候，就可以使用啦~

```
.border(@w:2px,@s:solid,@c:red){
    border: @arguments;
}
.box{
    .border;
}
-----------------------------------
.box {
  border: 2px solid #ff0000;
}
```

- 在有默认值的情况下，如果只想修改其中一个值，一定要指定修改的变量名

```
.border(@w:2px,@s:solid,@c:red){
    border: @arguments;
}
.box{
    .border(@c:blue);
}
-----------------------------------
.box {
  border: 2px solid #0000ff;
}
```

否则在正常情况下，传入的值和接受的位置是一一对应的。

11、深入混合

相同的自定义函数不会被替换

```
.box(){
    width: 100px;
}
.box(){
    height: 200px;
}
.box1{
    .box();
}
---------------------------------
```

```
.box1 {
  width: 100px;
  height: 200px;
}
```

混合匹配

```
.border(l){
    border-left: 10px solid red;
}
.border(r){
    border-right: 10px solid red;
}

.box{
    .border(r);
}
-------------------------------------------------
.box {
  border-right: 10px solid red;
}
```

- 不一样的参数会匹配出不一样的结果，上述案例，匹配的为参数 `r`
- 需要匹配的内容放在所有参数最前面

匹配任意值

`@_`表示允许匹配任意值

```
.border(l){
    border-left: 10px solid red;
}
.border(r){
    border-right: 10px solid red;
}
.border(@_){
    background: red;
}
.box{
    .border(d);
}
-----------------------------------------
.box {
  background: red;
}
```

可以看到上述的`.border`没有参数是`d`的，但是因为`@_`可以匹配任意参数，所以最后结果被添加上了`background:red`

利用混合特性，达到返回值效果

```
.box(){
    @w:100px;
    @h:200px;
}
.box1{
    .box();
    width: @w;
    height: @h;
}
---------------------------------------
.box1 {
  width: 100px;
  height: 200px;
}
```

- 在混合中定义的变量，在调用混合的作用域中也可以使用，我们可以利用这一特性达到返回值的效果

```
.average(@a,@b){
    @average: (@a + @b) / 2;
}
.box{
    .average(10px,20px);
    padding: @average;
}
```

1. 这里首先调用`.average`传入`10px,20px`
2. 然后在`.average`中声明一个变量`@average`
3. 在调用`.average`的位置，就可以使用变量`@average`

12、精灵图（雪碧图）

每一张图片都会走一次请求，请求次数过多，会造成性能问题，所以在这里，我们可以把一些可以合并的图片，放到一张图上，用background-position来控制图片显示的部分

尽量减少图片请求次数，也是网站优化必不可少的一部分

13、判断

我可不是你想调用就能调用的混合，有些时候，我也是有条件的，通过`when`关键词，可以为设定条件调用

```
.box(@w) when(@w>0){
    width:@w;
}
.box1{
    .box(40px);
}
--------------------------------------
.box1 {
  width: 40px;
}
```

- 只有当传入的数字大于0时才可以
- 当when后面`()`的结果为真，则添加后续样式，否则不执行

比较运算符

`>、<、>=、<=、=`

逻辑运算符

1. `and`可以连接两个条件，并且两个条件都需要满足，才可以添加后续样式

   ```
   .box(@w,@h) when (@w>0) and (@h>0){
       width: @w;
       height: @h;
   }
   .box1{
       .box(100px,0px);
   }
   ```

   - 因为高不符合条件，所以并没有添加上样式。

     (椰果和布丁我都要，不然我就不喝了)

2. `,`任何一个结果符合条件，就执行后面的代码（或）

```
.box(@w,@h) when (@w>0),(@h>0){
    width: @w;
    height: @h;
}
.box1{
    .box(100px,0px);
}
----------------------------------------
.box1 {
  width: 100px;
  height: 0px;
}
```

- 同样的值，同样的判断，但是在这里，我们添加上了后续样式

  (类似于：椰果或者布丁，我都行，其他的我不想喝)

3. `not`只要不是这个判断条件成立，就可以

```
.box(@w,@h) when not (@w=100){
    width: @w;
    height: @h;
}
.box1{
    .box(100px,120px);
}
```

- 因为这里的`@w = 10px`成立，所以样式没有添加上（排除当前写的这个情况，别的都可以）

  (类似于：只要不是巧克力奶茶，我都可以)

类型检查功能

- 基本检查类型
  - iscolor：是否为颜色
  - isnumber：是否为数字
  - isstring：是否为字符串
- 升级
  - ispixel：是否为像素
  - ispercentage：是否为百分比
  - isem：是否为em

条件混合

当上述条件不满足，希望执行另外一种情况的时候`default`

```
.box(@w,@h) when not (@w=100){
```

```
    width: @w;
    height: @h;
}
.box(@w,@h) when (default()){
    width: 0px;
    height: 0px;
}
.box1{
    .box(100px,120px);
}
---------------------------------------
.box1 {
  width: 0px;
  height: 0px;
}
```

- 因为上述的条件不满足，所以执行了`default`的情况

14、循环

利用判断，我们可以限制循环的停止条件

```
.width(@num) when (@num > 0){
    .width(@num - 1);
    width:100px * @num
}
div{
    .width(3);
}
---------------------------------------
div {
  width: 100px;
  width: 200px;
  width: 300px;
}
```

案例：利用循环做个列表

15、继承

继承是Less中的一个伪类，它可以匹配当前继承中的所有样式

如果用混合，操作以下css样式

```
.box{
    width: 200px;
```

```
    height: 200px;
    .a{
        color: red;
    }
}
.box1{
    .box;
}
----------------------------------
.box {
  width: 200px;
  height: 200px;
}
.box .a {
  color: red;
}
.box1 {
  width: 200px;
  height: 200px;
}
.box1 .a {
  color: red;
}
```

混合中，也同样给.box1 下面的 .a添加了样式，但其实我想要的只是.box1

我们利用继承可以得到想要的样式

```
.box{
    width: 200px;
    height: 200px;
    .a{
        color: red;
    }
}
.box1:extend(.box){

}
-------------------------------------------
.box,
.box1 {
  width: 200px;
  height: 200px;
}
.box .a {
  color: red;
}
```

这样就没有.a的样式干扰啦~

当然如果你想要所有的，也可以用`all`

```
.box{
    width: 200px;
    .a{
        color: red;
    }
}
.box{
    background: red;
}
.box1:extend(.box all){

}
--------------------------------
.box,
.box1 {
  width: 200px;
}
.box .a,
.box1 .a {
  color: red;
}
.box,
.box1 {
  background: red;
}
```

和混合的最终结果是一样的，但是在显示上面是有差别的，这里最终的显示会用群组选择器

16、深入继承

```
.a{
    color: red;
}
.b{
    background: green;
}
.c{
    &:extend(.a);
}
----------------------------
.a,
.c {
  color: red;
}
```

```
.b {
  background: green;
}
```

上一节课，我们说了通过extend可以进行扩展，加上`all`关键词，更是不得了，可以把子级都扩展过来，那如果我想扩展的参数，不止一个呢？？

多参数继承

```
.a{
    color: red;
}
.b{
    background: green;
}
.c{
    &:extend(.a,.b);
}
---------------------------------
.a,
.c {
  color: red;
}
.b,
.c {
  background: green;
}
```

以逗号进行间隔，填入多个想要扩展的class

```
.c{
    &:extend(.a,.b);
}
```

可以针对当前扩展的，选择是否添加关键字

注意：扩展必须是在最后的

```
&:extend(.a,.b):hover{

};
```

上面这样的写法是错误的

```
&:hover:extend(.a,.b){

};
```

这是正确的写法

- 继承采用的是`精确匹配`,也就是选择器必须长得就一模一样
- 扩展必须写到最后
- 目的是为了减少基类

```
<a class="fonts color"></a>
```

```
.fonts{
    font-size:20px;
    color:red;
}
.color{
    color:green;
}
```

以下是使用Less的

```
<a class="color"></a>
```

```
.fonts{
    font-size:20px;
    color:red;
}
.color{
    &:extend(.fonts);
    color:green;
}
```

把`fonts`作为扩展添加给color，当然也有缺点，如果你的`fonts`修改了，color也会更着改，当然这也是好处。

17、合并

允许对可以接受多个属性的，例如：背景、阴影进行拼接合并

利用`,`隔断的，使用`+`

```
.box(){
    width: 200px;
    height: 200px;
    box-shadow+: 10px 10px 5px #000;
```

```
}
.box2{
    .box();
    box-shadow+: inset 5px 5px 5px red;
}
-----------------------------------------------
.box2 {
  width: 200px;
  height: 200px;
  box-shadow: 10px 10px 5px #000,
          inset 5px 5px 5px red;
}
```

利用`空格`，隔断的，使用`+_`

```
.box(){
    width: 200px;
    height: 200px;
    transform+_: translate(100px);
}
.box2{
    .box();
    transform+_: rotate(3600deg);
}
-----------------------------------------------------
.box2 {
  width: 200px;
  height: 200px;
  transform: translate(100px) rotate(3600deg);
}
```

- 为避免无意义的链接，所以在这里要求，凡是需要拼接的都要加上符号`+`或`+_`

18、!important 关键词

正常用法

```
.box{
    font-size: 20px !important;
}
```

运用在混合模式

```
.box{
    width: 200px;
    height: 200px;
```

```
}
.box1{
    .box() !important;
}
----------------------------------------
.box {
  width: 200px;
  height: 200px;
}
.box1 {
  width: 200px !important;
  height: 200px !important;
}
```

会给混合中的每一条都添加上`!important`

19、导入

在制作项目的时候，特别是大型项目的时候，我们经常会拆分出很多的css，来进行管理，Less也是同理

|---- reset.Less 清除默认样式

|---- base.Less 基础样式

|-----variable.Less 变量

|-----页面.Less 某个页面的Less样式

以上的文件组成中，`variable.Less 变量`是我们经常会用到的，而且很可能在每个页面都需要用到，我们不会再每个页面重复去声明这些变量，而是整理在一个`Less`中，需要使用的地方引入

当然如果，你自己定义的方法多了，也是需要单独拆分为一个单独的`Less`的

语法

```
@import 'Less/variable.Less';
```

这样，你就可以在引入这个文件的页面，放心使用这些变量了，而且后期在修改的时候，你也不需要一个个页面去找变量，打开这个Less文件就可以了。

- 如果导入的是`Less`文件，后缀名可以省略
- 导入位置可以随意放置

导入参数

- reference：使用Less文件，但不输出

在css中也是可以通过这样的方式引入文件的

```
@import 'css/base.css';
```

但是注意：

- 必须是样式表中导入
- 必须是在当前样式表的最前面
- 结束必须要加上分号
- IE下导入不能超过35条
- IE9以下不支持

所以，同样的导入方式，我们还是推荐使用link

20、编码

和`html`页面一样，`css`页面偶尔也会出现中文，例如同学命名图片的时候，当然还可能出现其他语言，所以对于Less和`css`，我们同样也可以设定编码

```
@charset "utf-8";
```

Less函数

常用其他函数

color

它是一个代表颜色的字符串（将字符串转换为颜色值）；

```
div{
  background: color('pink');
  color:color('#ff0000');
}
----------------------------------------
div{
  background: #ffc0cb;
  color: #ff0000;
}
```

convert

数字从一个单位转换为另一个单位

第一个参数包含带有单位的数字，第二个参数包含单位。如果单位兼容，则转换该数字。如果它们不兼容，则第一个参数将不加修改地返回。

兼容的单位组:

- 长度：`m`，`cm`，`mm`，`in`，`pt`和`pc`，
- 时间：`s`和`ms`，
- 角度：`rad`，`deg`，`grad`和`turn`。

参数：

- `number`：带单位的浮点数。
- `identifier`，`string`或`escaped value`：单位

```
convert(1s,ms)
----------------------------------------
1000ms
```

unit

默认函数仅在保护条件内可用且与任何其他mixin不匹配时才返回true

Less支持许多测量，包括绝对单位，如英寸，厘米，点等。 它可以更改尺寸单位。

参数：

- `dimension`：一个数字，有或没有维度。
- `unit`:(可选）要更改的单位，或者如果省略则删除单位。

```
div{
  font-size:unit(20,px);
}
----------------------------------------
div{
  font-size:20px;
}

unit(10em)
----------------------------------------
10
```

常用列表函数

length

它将以逗号或空格分隔的值列表作为参数，并返回表示列表中元素数量的整数

```
p{
    @list:  "benz", "toyota","honda","renault","bmw","tata","ford","skoda";

    @val: length(@list);

    font-size:@val + px;
 }
----------------------------------------
 p{
  font-size:8px;
 }
```

extract

它将返回列表中指定位置的值

```
p{
  @list: 10px, 20px, 30px, 40px;
  @val: extract(@list, 2);
  font-size:@val;
  color:#F79F81;
}
----------------------------------------
 p{
  font-size:20px;
  color:#F79F81;
 }
```

常用数学函数

ceil

它将数字向上舍入为下一个最大整数

```
ceil(0.7);
----------------------------------------
1
```

floor

它将数字向下取整为下一个最小整数

```
floor(3.3);
----------------------------------------
3
```

percentage

它将浮点数转换为百分比字符串

```
percentage(0.3);
----------------------------------------
30%
```

round

它舍入浮点数

```
round(5.78);
----------------------------------------
6
```

sqrt

它返回一个数字的平方根

```
sqrt(25);
----------------------------------------
25
```

abs

它提供了数字的绝对值

```
abs(30ft);
----------------------------------------
30ft;
```

min

它指定一个或多个参数的最小值

```
min(10,5,20,30);
----------------------------------------
5
```

max

它指定一个或多个参数的最大值

```
max(10,20,30,40);
----------------------------------------
40
```

常用类型函数

isnumber

它使用一个值作为参数，并返回 *true* ，如果它是一个数字或 *false*

```
isnumber(1234);
----------------------------------------
true
```

isstring

它使用一个值作为参数，并返回 *true* ，如果它是一个字符串或 *false*

```
isstring(1234);
----------------------------------------
false
```

iscolor

它使用一个值作为参数，并返回 *true* (如果值是颜色)或 *false* (如果不是)

```
iscolor(red);
----------------------------------------
true
```

常用颜色定义函数

rgb

它创建从红色，绿色和蓝色值的颜色

红色:包含介于0 - 255之间的整数或介于0 - 100%之间的百分比

绿色:它包含介于0 - 255之间的整数或介于0 - 100%之间的百分比

蓝色:包含介于0 - 255之间的整数或介于0 - 100%之间的百分比

```
rgb(220,20,60)
----------------------------------------
#dc143c
```

rgba

红色，绿色，蓝色和alpha（RGBA）值创建透明颜色对象

hsl

根据色调，饱和度和亮度（HSL）值创建不透明的颜色对象

hue :它包含介于0 - 360之间的整数，表示度数

饱和度:它包含介于0 - 1之间的数字或介于0 - 100%之间的百分比

亮度:它包含介于0 - 1之间的数字或介于0 - 100%之间的百分比

```
hsl(120,100%,50%)
----------------------------------------
#00ff00
```

常用颜色通道函数

hue

在HSL颜色空间中，提取颜色对象的色调通道

```
background: hue(hsl(75, 90%, 30%));
----------------------------------------
background: 75;
```

alpha

提取颜色对象的alpha通道

```
background: alpha(rgba(5, 15, 25, 1.5));
----------------------------------------
background:1.5;
```

常用颜色操作函数

lighten

它增加了元素中颜色的亮度

它有以下参数：

color： 它代表颜色对象。

amount： 它包含0 - 100%之间的百分比。

方法：它是可选参数，通过将其设置为相对，用于相对于当前值进行调整。

```
@color-base:#3bafcc;
@color-ligten:lighten(@color-base, 10%); //#63c0d7

.myclass1{
   color:@color-ligten;
}

----------------------------------------
.myclass1 {
  color:#63c0d7;
}
```

darken

它改变元素中颜色的强度或饱和度

它有以下参数：

color： 它代表颜色对象。

amount： 它包含0 - 100%之间的百分比。

方法：它是可选参数，通过将其设置为*相对*，用于相对于当前值进行调整。

```
@color-base:#3bafcc;
@color-darken:darken(@color-base, 10%); //#2c8fa8

.myclass2{
  color:@color-darken;
}
----------------------------------------

.myclass2 {
 color:#2c8fa8;
}
```

fadein

它增加了所选元素的不透明度

它有以下参数：

color： 它代表颜色对象。

amount： 它包含0 - 100%之间的百分比。

方法：它是可选参数，通过将其设置为*相对*，用于相对于当前值进行调整。

```
@color-base:#3bafcc;
@color-fadein:fadein(@color-base,10%);

.myclass1{
  color:@color-fadein;
}

----------------------------------------

background: #3bafcc;
```

fadeout

它减少所选元素的不透明度

它有以下参数：

color： 它代表颜色对象。

amount： 它包含0 - 100%之间的百分比。

方法：它是可选参数，通过将其设置为*相对*，用于相对于当前值进行调整

```
@color-base:#3bafcc;
@color-fadeout:fadeout(@color-base,10%);

.myclass1{
  color:@color-fadeout;
}

----------------------------------------

background: rgba(59, 175, 204, 0.9);
```

fade

它用于设置所选元素的颜色的透明度

它有以下参数：

color： 它代表颜色对象。

amount： 它包含0 - 100%之间的百分比。

```
@color-base:#3bafcc;
@color-fade:fade(@color-base,50%);

.myclass1{
  color:@color-fade;
}

----------------------------------------

background: rgba(59, 175, 204, 0.5);
```

不常用函数

其他函数

image - size： 它用于从文件检查图像的维度

image - width： 它检查文件中图像的宽度

image-height： 它检查图像从文件的高度

data - uri：数据uri是统一资源标识符(URI)模式，其在网页中嵌入资源

default： 默认函数仅在保护条件内可用且与任何其他混合程序不匹配时才返回true

get - unit： get-unit函数返回其中存在参数的单位，其数字和单位

字符串函数

逃逸：

它通过对特殊字符使用URL编码来对字符串或信息进行编码。 您无法编码一些字符，例如**，** ， **/** ，**？** ，**@** ，**&** ， **+** ，**～**，**！** ， **$** ，**'**和您可以编码的一些字符，例如 **** ，**#**， **>** **^** ，**(**，**)**， **{**，**}** ，**:>**，**>** ，**，** **]** ， **[和 =** 。

e：它是一个字符串函数，它使用string作为参数，并返回不带引号的信息。 它是一个CSS转义，它使用 *～"一些内容"*转义的值和数字作为参数

更换：它用于替换字符串中的文本。 它使用一些参数：

string :它搜索字符串并替换

pattern :它搜索正则表达式模式

replacement :它替换与模式匹配的字符串

flags :这些是可选的正则表达式标志

%格式：此函数格式化一个字符串

数学函数

sin：它返回数字上的弧度

asin：它指定返回-pi / 2和pi / 2之间的值的数字的反正弦(反正弦)

cos：它返回指定值的余弦值，并在没有单位的数字上确定弧度

acos：它指定返回0和pi之间的值的数字的反余弦(反余弦)

tan：它指定数字的正切

atan：它指定指定数字的反正切(反正切)

pi：它返回pi值

pow：它指定第一个参数的值增加到第二个参数的权力

mod：它返回相对于第二个参数的第一个参数的模数。 它还处理负点和浮点数

sqrt：它返回一个数字的平方根

类型函数

iskeyword：它使用一个值作为参数，并返回 *true* (如果值是关键字)或 *false* (如果不是)

isurl：它使用一个值作为参数，并返回 *true* (如果值为url)或 *false* (如果值不为)

ispixel：如果值是以像素为单位的数字，或者 *false* ，则以值为参数返回 *true*

isem：它采用一个值作为参数，并返回 *true* ，如果值是em值或 *false* (如果值不是)

ispercentage：它采用一个值作为参数，如果值不是百分比，则返回 *true* ，如果值为百分比，或返回 *false*

isunit： 如果值是指定单位中作为参数提供的数字，则返回 *true* ，如果值不是指定单位中的数字，则返回 *false*

颜色定义函数

argb： 创建格式为 `#AARRGGBB` 的十六进制颜色 (注意不是 #RRGGBBAA),这种格式被用在IE滤镜中，以及.NET和Android开发中

hsla：根据色调，饱和度，亮度和alpha（HSLA）值创建透明颜色对象

hsv： 根据色调，饱和度和值（HSV）值创建不透明的颜色对象

hsva：根据色调，饱和度，值和alpha（HSVA）值创建透明颜色对象

颜色通道函数

saturation：在HSL颜色空间中，提取彩色对象的饱和通道

lightness：在HSL颜色空间中，从颜色对象提取亮度通道

hsvhue： 在HSV色彩空间中，提取色彩对象的色调通道

hsvsaturation： 在HSL颜色空间中，提取彩色对象的饱和通道

hsvvalue： 在HSL颜色空间中，提取颜色对象的值通道

luma： 计算颜色对象的亮度值

red：提取彩色对象的红色通道

green：提取彩色对象的绿色通道

blue：提取彩色对象的蓝色通道

颜色操作函数

desaturate：它降低了元素中颜色的强度或饱和度

saturate：它改变元素中颜色的强度或饱和度

spin：它用于旋转所选元素的颜色的角度

mix：它将两种颜色与不透明度混合

tint：它将颜色与白色混合，同时减少颜色的比例

shade：它将颜色与黑色混合，因为您减少了颜色的比例

greyscale：它从所选元素中的颜色中丢弃饱和度

contrast：它设置元素中颜色的对比度

颜色混合函数

multiply：它将两种颜色相乘，并返回结果颜色

screen：它需要两种颜色，并返回更明亮的颜色。它与*乘法*函数相反

overlay：它通过组合*乘法*和*屏幕*的效果来生成结果

softlight：它的工作方式类似于*覆盖*，但它仅使用颜色的一部分，其中柔和突出显示其他颜色

hardlight：它的工作方式类似于*覆盖*，但颜色的作用相反

difference：它从第一输入颜色中减去第二输入颜色

exclusion：它的工作原理类似于*差异*功能，但对比度较低

average：它计算每个通道(RGB)基础上的两种输入颜色的平均值

negation：它与opposite函数相反，其从第二颜色中减去第一颜色