

LESS 动态样式语言

(陈华旺-chenhuawang@itany.com)

目录 [LESS 动态样式语言]

1、LESS简介

2、LESS的使用

- 2.1、Node环境下得LESS编译器

- 2.2、vscode 插件自动编译

- 2.3、浏览器自动编译

- 2.4、GULP 自动化 workflow 编译

3、LESS的基础语法

- 3.1、LESS注释

- 3.2、LESS的变量定义

- 3.3、选择器嵌套

- 3.4、父级选择器名称替代符号: `&`

- 3.5、样式继承

- 3.6、样式混合

- 3.6.1、简单混合器

- 3.6.2、带参混合器

- 3.6.3、导引混合器

4、LESS的四则运算

5、LESS 内置函数

- 5.1、图片函数

- 5.2、单位函数

- 5.3、字符串函数

- 5.4、列表函数

- 5.5、数学函数

- 5.6、颜色函数

- 5.6.1、颜色操作函数

- 5.6.2、颜色混合函数

6、LESS 模块化

7、CSS 编写的一点建议

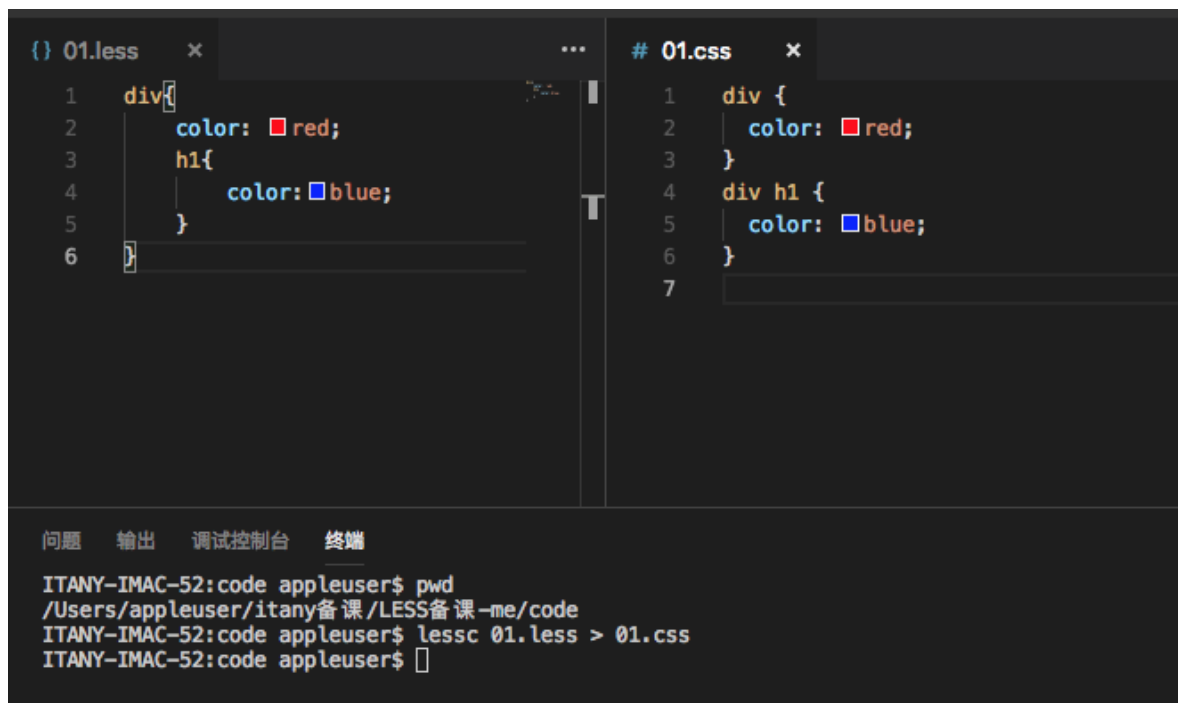
1、LESS简介

- Less 是一个**Css 预编译器**,意思指的是它可以扩展Css语言,添加功能如允许变量(variables),混合(mixins),函数(functions) 和许多其他的技术, 让你的Css更具维护性, 主题性, 扩展性。
- Less 可运行在 Node 环境,浏览器环境和Rhino环境.同时也有3种可选工具供你编译文件和监视任何改变。

2、LESS的使用

2.1、Node环境下得LESS编译器

- 可在NODE环境下 安装 全局LESS 编译器, `[sudo] npm install less -g` , 全局编译命令 **lessc**
- 对less 文件进行编译 `lessc style.less > style.css`



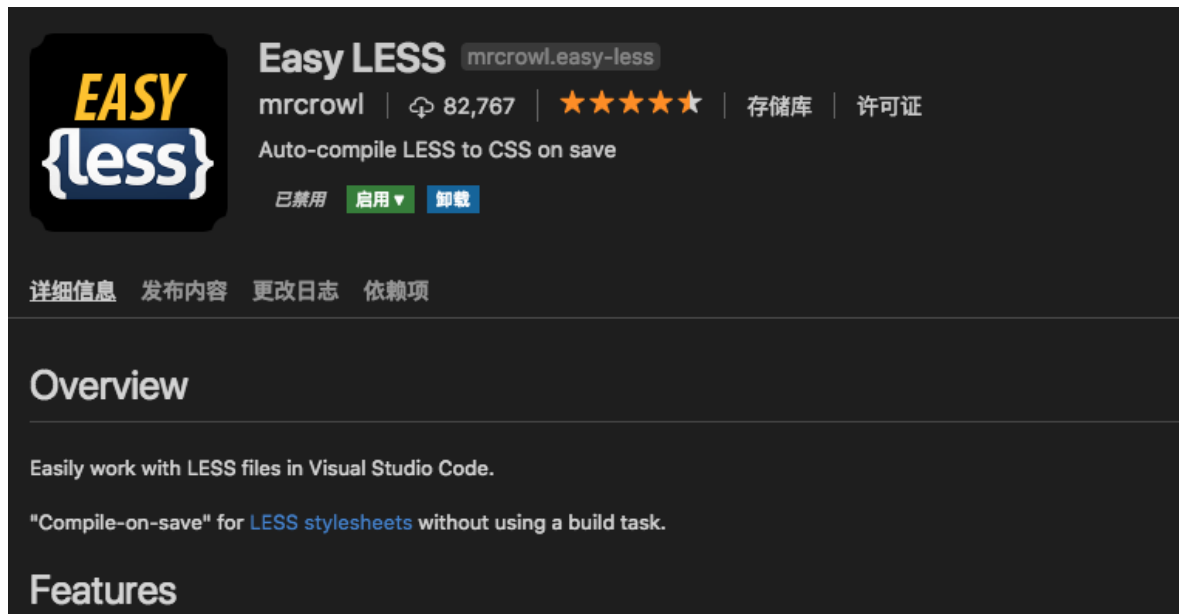
```
{ } 01.less x
1  div{
2    color: red;
3    h1{
4      color: blue;
5    }
6  }

# 01.css x
1  div {
2    color: red;
3  }
4  div h1 {
5    color: blue;
6  }
7

问题 输出 调试控制台 终端
ITANY-IMAC-52:code appleuser$ pwd
/Users/appleuser/itany备课/LESS备课-me/code
ITANY-IMAC-52:code appleuser$ lessc 01.less > 01.css
ITANY-IMAC-52:code appleuser$
```

2.2、vscode 插件自动编译

- 在vscode编译器安装 **Easy LESS** 编译器, 可在.less文件被保存时, 自动在相同位置 编译并保存一个同名的 .css文件



2.3、浏览器自动编译

- LESS提供JS 文件版本的 HTML 自动编译插件，可在相关页面直接使用 LESS语法

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <meta http-equiv="X-UA-Compatible" content="ie=edge">
7      <title>Document</title>
8      <!-- <script src="./js/less.min.js"></script> -->
9      <!-- 传统CSS -->
10     <!-- <style type="text/css">
11         div{
12             color:red;
13         }
14         div h1{
15             color: blue;
16         }
17     </style> -->
18     <!--
19         less 语法
20         一定要 指定 语法类容
21         且 代码需要在 编译文件之前
22     -->
23     <style type="text/less">
24         div{
25             color:red;
26             h1{
27                 color: blue;
28             }
29         }
```

```

30     </style>
31     <script src="./js/less.min.js"></script>
32 </head>
33 <body>
34     <div>
35         测试
36         <h1>测试</h1>
37     </div>
38     <h1>比较</h1>
39 </body>
40 </html>

```

2.4、GULP 自动化工作流编译

- 在项目中 新建 gulp 工作流 使用 工作流自动完成less 编译
- 依赖编译包 `gulp-less`

```

1  // gulpfile 文件 ==> watch bug 修复
2  const gulp = require("gulp");
3  const less = require("gulp-less");
4  const watch = require("gulp-watch");
5
6  gulp.task("less", function(){
7      gulp.src("./less/**/*.less")
8          .pipe(less())
9          .pipe(gulp.dest("css"));
10 });
11
12 gulp.task("watch", function(){
13     watch("./less/**/*.less", function(){
14         gulp.start('less');
15     })
16 });
17
18 gulp.task("default", ["less", "watch"], function(){
19     console.log("启动LESS自动编译.....");
20     console.log("=====");
21 });

```

3、LESS的基础语法

3.1、LESS注释

- 编译保留注释 `/* */`，编译后的css文件中会保留该注释
- 编译删除注释 `//`，编译后的css文件中不会保留该注释

3.2、LESS的变量定义

- LESS 的值变量定义：@变量名:变量值;，定义后的变量可在 css 选择器中直接使用 @变量名

```
1  /* 保留注释 */
2  // 删除注释
3
4  // 1.基础变量定义
5  @white: #fff;
6  @light: #f8f9fa;
7  @gray: #6c757d;
8  @secondary: #6c757d;
9  @gray-dark: #343a40;
10 @dark: #343a40;
11 @blue: #007bff;
12 @indigo: #6610f2;
13 @purple: #6f42c1;
14 @pink: #e83e8c;
15 @red: #dc3545;
16 @orange: #fd7e14;
17 @yellow: #ffc107;
18 @green: #28a745;
19 @teal: #20c997;
20 @cyan: #17a2b8;
21
22 @size:20px;
23
24 // 变量可引用变量
25 @link-color:@cyan;
26 @link-color-hover:@green;
27
28 // 变量可使用 驼峰命名法
29 @textDecoration:none;
30
31 // 变量支持多值定义
32 @border:1px solid @pink;
33
34 div{
35     color:@dark;
36     background-color: @blue;
37 }
38 #aa{
39     font-size: @size;
40     padding: 10px @size;
41 }
42 a:link{
43     color: @link-color;
44     text-decoration: @textDecoration;
45 }
46 a:hover{
47     color: @link-color-hover;
```

```

48     text-decoration: @textDecoration;
49 }
50 li{
51     border: @border;
52 }

```

- LESS 路径变量定义：@变量名:值，定义后可在属性值上使用，@{变量名}

```

1 // 2、路径变量定义
2 @img-base-url: "./imgs";
3 div{
4     // 使用是@{变量名} 不能出现多余空格
5     // background-image: url("@{ img-base-url }/logo.png");
6     background-image: url("@{img-base-url}/logo.png");
7 }

```

- LESS 选择器变量定义：@变量名:值，定义后可在选择器上使用，@{变量名}

```

1 // 3、选择器变量定义
2 @select1: banner;
3 .@{select1} {
4     font-weight: bold;
5     line-height: 40px;
6     margin: 0 auto;
7 }
8 div #@{select1}{
9     color: @red;
10 }

```

- LESS 属性变量定义：@变量名:值，定义后可在选择器上使用，@{变量名}，前缀-@{变量名}

```

1 // 4、属性变量定义
2 @property: color;
3 .widget {
4     @{property}: #0ee;
5     background-@{property}: #999;
6 }
7 @chrome: -webkit;
8 @firefox: -moz;
9 @ie: -ms;
10 @opera: -o;
11 div{
12     transform: rotate(7deg);
13     @{chrome}-transform: rotate(7deg);
14     @{firefox}-transform: rotate(7deg);
15     @{ie}-transform: rotate(7deg);

```

```

16   @{{opera}}-transform: rotate(7deg);
17 }

```

- LESS 可变变量定义：@变量名:值，定义后可在选择器上使用，@@变量名

```

1 // 5、可变变量使用
2 @size:24px;
3 @name:size;
4 .section {
5     font-size: @@name;
6 }

```

- LESS 变量作用域：变量可以定义在整改 LESS文件的顶级 作为 **全局变量存在**，也可以定义在选择器的内部 作为 **局部变量存在**

```

1 // 6、变量作用域
2 h1{
3     color: @red;
4     // 无法加载 @aa变量
5     // border: @aa;
6 }
7 h2{
8     @red:blue;
9     @aa:1px solid @red;
10    color: @red;
11    border: @aa;
12 }
13 h3{
14    color: @red;
15    // 无法加载 @aa变量
16    // border: @aa;
17 }

```

- 变量懒加载：LESS 允许 先使用变量 后定义变量

```

1 // 7、变量懒加载
2 ul{
3     color: #007bff;
4     border: 1px solid @cc;
5 }
6 @cc:#dedede;

```

- 变量默认值：LESS 中可通过定义变量先后的方式，实现变量默认值的操作,无论后定义的变量在 less文件的什么位置，less文件中所有引用该变量的属性都会改变

3.3、选择器嵌套

- LESS 语法中允许实现 选择中定义选择器的语法，可以快速实现 选择器嵌套规则

```
1 // 选择器嵌套
2 @white: #fff;
3 @light: #f8f9fa;
4 @gray: #6c757d;
5 @secondary: #6c757d;
6 @gray-dark: #343a40;
7 @dark: #343a40;
8 @blue: #007bff;
9 @indigo: #6610f2;
10 @purple: #6f42c1;
11 @pink: #e83e8c;
12 @red: #dc3545;
13 @orange: #fd7e14;
14 @yellow: #ffc107;
15 @green: #28a745;
16 @teal: #20c997;
17 @cyan: #17a2b8;
18
19 ul{
20     list-style: none;
21     li{
22         font-size: 12px;
23         color: @orange;
24         > a{
25             text-decoration: none;
26         }
27         > i{
28             color: @secondary;
29         }
30     }
31 }
32
33 div{
34     color: @cyan;
35     p,h1,h2,h3,h4,h5,h6{
36         padding: 0px;
37         margin: 0px;
38     }
39 }
40
41 table{
42     border-collapse: collapse;
43     @size:1px;
44     th,tr{
45         border: @size solid black;
46         td{
47             @size:2px;
```



```

48         border-left:@size solid @red;
49     }
50 }
51 }

```

3.4、父级选择器名称替代符号：&

less可以使用&符号在 选择器内部用于指向父级选择器名称。

- 父级选择器 经常被用在伪类 或 伪元素选择器上

```

1 // 1、父级选择器 经常被用在伪类 或 伪元素选择器上
2 a{
3     color: red;
4     // 标示当前 a 标签中得 a 标签
5     // a:hover{
6     //     color: blue;
7     // }
8     &:hover{
9         color: blue;
10    }
11 }
12 div{
13     font-size: 20px;
14     &::after{
15         content: "测试文本";
16     }
17 }

```

- 父级选择器 也可用在样式 分组中

```

1 // 2、父级选择器 也可用在样式 分组中
2 .btn{
3     @success:#28a745;
4     @error:#dc3545;
5     @info:#17a2b8;
6     @warning:#fd7e14;
7     @border:1px solid;
8     &-success{
9         background-color:@success;
10        border: @border @success;
11    }
12    &-error{
13        background-color: @error;
14        border: @border @error;
15    }
16    &-info{
17        background-color: @info;

```

```

18     border: @border @info;
19 }
20 &-warning{
21     background-color: @warning;
22     border: @border @warning;
23 }
24 &link{
25     text-decoration: none;
26 }
27 }

```

- 多重样式名组合

```

1 // 3、多重样式名组合
2 .link {
3     & + & {
4         color: red;
5     }
6     & & {
7         color: green;
8     }
9     && {
10        color: blue;
11    }
12    &, & {
13        color: cyan;
14    }
15    & > & {
16        color: orange;
17    }
18 }

```

- 多级父选择器

```

1 // 4、多级父选择器
2 // & 标示父级所有选择器，不关心选择器层级
3 .content .info{
4     &:hover{
5         color: red;
6     }
7     & &{
8         font-size: 20px;
9     }
10 }
11 .nav{
12     .nav-bar{
13         & > &{

```

```

14         color: red;
15     }
16 }
17 }

```

- 分组选择器迭代

```

1 // 5、分组选择器迭代
2 h1,h2,h3,h4,h5,h6,p{
3     margin: 0px;
4     // LESS 会使用两两配对的方式 实现 选择器组合
5     & &{
6         padding: 0px;
7     }
8 }

```

3.5、样式继承

LESS 允许对相同样式进行继承操作，减少对相同样式的重复定义

```

1 // 样式继承
2 .tip{
3     width: 200px;
4     height: 40px;
5     font-size: 20px;
6     line-height: 40px;
7     text-align: center;
8     color: white;
9     background-color: #6f42c1;
10    a{
11        text-decoration: none;
12        &:hover{
13            color: blue;
14        }
15    }
16    .text{
17        margin: 10px;
18        padding: 10p;
19    }
20 }
21
22 .border-tip{
23     // 实现对现有样式的继承操作
24     // LESS 只会完成对 指定样式的继承，不会继承其子选择的设置
25     // LESS 中 继承依然可以使用 css的相关层级选择器，进行定位，但无法直接继承非同级选
    择器
26     &:extend(.tip .text);
27     border: 1px solid #6610f2;

```

```

28 }
29
30 .tip2{
31     // 实现对样式继承时，写入 关键字 all 时，可完成所有属性继承，包括 子选择器的样式
32     &:extend(.tip all);
33     margin: 20px;
34 }
35
36 // 继承 伪类写法
37 // 继承相关选择器的样式
38 .tip3:extend(.tip .text){
39     overflow: hidden;
40 }
41 // 继承可以使用 多继承方式
42 .tip4:extend(.tip):extend(.tip .text){
43     display: inline;
44 }
45
46 // 继承 无法使用变量方式进行 匹配，extend 会直接忽略
47 @name:aaa;
48 .@{name}{
49     color: red;
50 }
51 div:extend(@{name}){
52     font-size: 20px;
53 }

```

3.6、样式混合

3.6.1、简单混合器

- LESS 允许进行样式预定义，并使用混合语法将预定义样式 加载进行 指定样式中
- LESS 中**除包含标签选择器、伪类选择器、伪元素选择器，以外的其它选择器**，均可以作为 混合样式使用
- LESS 在混合样式时，会将指定的混合器中所有的样式和层级进行组合

```

1 // 样式混合
2 // 1、简单混合
3 .wbs{
4     font-size: 24px;
5 }
6 #aa{
7     text-decoration: underline;
8 }
9 div{
10     padding: 10px;
11 }
12 #dd .ff{
13     background-color: white;

```

```

14 }
15 .pp #ll{
16     margin: 10px;
17 }
18 #dd div{
19     font-family: 'Courier New', Courier, monospace;
20 }
21 .link:hover{
22     color: red;
23 }
24 // 混合器 组合样式时，会将选择器 所有层级进行 整体组合
25 #itany{
26     font-weight: bold;
27     a{
28         color: blue;
29     }
30 }
31
32 .cc{
33     display: inline-block;
34     .wbs;
35     #aa;
36     // div;
37     #dd .ff;
38     .pp #ll;
39     // #dd div;
40     // .link:hover;
41     #itany;
42 }

```

3.6.2、带参混合器

- LESS 允许定义 一个类似函数的 带参混合器，该混合器无法作为普通选择器进行使用，但允许在调用该混合器时使用 自定义数据
- 混合器定义： `混合器选择器(变量,.....){样式}`
- 混合器使用： `普通选择器{混合器选择器(变量,.....);}`

```

1 // 带参混合器
2 #font-color(@color){
3     color: @color;
4 }
5 .border(@width,@style,@color){
6     border: @width @style @color;
7 }
8 table{
9     border-collapse: collapse;
10    #font-color(red);
11    td{

```

```

12     font-size: 12px;
13     .border(1px, solid, black);
14 }
15 }
16 ul{
17     list-style: none;
18     #font-color(blue);
19     li{
20         font-size: 12px;
21         .border(2px, dashed, yellow);
22     }
23 }
24
25 // 参数默认值
26 .setColumns(@columns:100px 2,@column-fill:auto,@column-gap:10px,@column-
27 rule:2px dashed blue){
28     columns: @columns;
29     column-fill: @column-fill;
30     column-gap: @column-gap;
31     column-rule: @column-rule;
32 }
33 #d1{
34     width:400px;
35     .setColumns(100px 3,balance,5px,1px solid red);
36 }
37 #d2{
38     width:400px;
39     .setColumns;
40 }
41 // 案例
42 #hover(@target,@size,@color){
43     &{
44         font-size: @size;
45     }
46     @{{target}}:hover{
47         color: @color;
48     }
49 }
50 ul{
51     list-style: none;
52     li{
53         padding: 10px 0px;
54         #hover(a, 20px, red);
55     }
56 }

```

3.6.3、导引混合器

- LESS 允许定义 根据条件进行 参数选择的混合器，类似于 JS 中的 `switch case`，在LESS 中成为 导引表达式

```
1 // 导引混合器
2 // 导引表达式，即使用when进行判断
3 .hello(@font-size) when(@font-size<20){
4     font-size:@font-size;
5     color:red;
6 }
7 .hello(@font-size) when(@font-size>=20){
8     font-size:30px;
9     color:blue;
10 }
11 .hello(@font-size){ //总是会匹配
12     background-color: #ccc;
13 }
14 .c2{
15     .hello(18px);
16 }
17 .c3{
18     .hello(50px);
19 }
```

4、LESS的四则运算

- LESS 中支持对部分CSS数据进行 四则运算

```
1 @num1:21px;
2 @num2:@num1+4px;
3 @num3:@num1+@num2;
4 @num4:(@num1+3)*2;
5 // @num5:@num1%2;
6 @num5:@num1/2;
7 @color:#666/2;
8 @bgColor:@color+#222;
9 @color2:#123456+#234; //颜色双位相加
10
11 #d1{
12     font-size: @num2;
13     font-size: @num3;
14     font-size: @num4;
15     font-size: @num5;
16     color:@color;
17     border:unit(@num1/2+3,px) dashed red;
18     background-color: @bgColor;
19     background-color: @color2;
20 }
```

5、LESS 内置函数

5.1、图片函数

- LESS 内置的图片函数，主要用于获取图片的长宽像素,图像长宽函数 只能在node环境中使用 和 base64转换
 - image-size：从文件获取的图像尺寸。
 - 参数：string: 获取尺寸的文件。
 - 返回：宽px 高px
 - image-width：从文件获取的图像宽度。
 - 参数：string: 获取尺寸的文件。
 - 返回：宽px
 - image-height：从文件获取的图像高度。
 - 参数：string: 获取尺寸的文件。
 - 返回：高px

```
1 // 图片函数
2 // image-size 获取图片的长宽
3 h2{
4     padding: image-size("../imgs/logo.png");
5 }
6 img{
7     width: image-width("../imgs/qq.jpeg");
8     height: image-height("../imgs/qq.jpeg");
9 }
```

5.2、单位函数

- LESS 内置对 数值单位进行转换和处理的函数
 - convert(arg1,arg2)：将数字从一种单位 计算转换到 另一种类型。
 - 第一个参数为带单位的数值，第二个参数为单位。

如果两个参数的单位是兼容的，则数字的单位被转换。如果两个参数的单位不兼容，则原样返回第一个参数。
 - 兼容的单位组：

长度: m, cm, mm, in, px
时间: s ms,
角度: rad, deg
 - 返回： number
 - unit(arg1,arg2)：移除或者改变属性值的单位。
 - 参数：

arg1: 数字，带或不带单位。

arg2: 可选参数, 将要替换成的单位, 如果省略则移除原单位。

```
1 // 单位函数
2 //convert()函数
3 .c2{
4     animation-delay: convert(3000ms,s);
5     transition-duration: convert(2s,ms);
6     width: convert(20mm,px);
7     transform: rotate(convert(0.5rad,deg));
8     height: convert(30deg,px); //无法转换, 返回原始数据
9 }
10 //unit()函数
11 .c1{
12     font-size:unit(20,px);
13     font-size: unit(20px,rem);
14     font-size: unit(20px); //删除单位
15 }
```

5.3、字符串函数

- 字符串替换函数

- replace (替换): 用另一个字符串替换文本.

- 参数:

- string: 搜索和替换用的字符串.

- pattern: 一个字符串或者能搜索的正则表达式.

- replacement: 匹配模式成功的替换字符串.

- flags: (可选的) 正则表达式匹配标识 (全匹配还是...) .

- 返回: 替换过值后的字符串.

```
1 // 字符串函数
2 //replace()函数
3 #d1:after{
4     content:replace("welcome to itany.", "o", "0");
5     content:replace("welcome to itany.", "o", "0", g);
6     content:replace("this is string.", "(string)", "new $1"); // $1是占位符, 表示第1
    个匹配值
7     content:replace("welcome to nanjing.", "^welcome to nanjing.$", "aa");
8 }
```

5.4、列表函数

- length (长度): 返回集合中的值的数目.

- 参数: `list` - 以空格或逗号隔开的值集合.

- 返回: 集合的 **值** 个数.

- `extract` (提取)：返回集合中指定索引的值。
 - 参数：
 - `list` - 逗号或者空格隔开的值集合。
 - `index` - 用于返回集合中指定位置值的整型数字。
 - 返回：集合指定位置处的值。

```

1 // 列表函数
2 //length()函数
3 @names1:"tom","jack","alice";
4 @names2:"tom" "jack" "alice" "mike";
5 @str:"aabbcc";
6 #d1{
7     font-size:unit(length(@names1),rem);
8     font-size:unit(length(@names2),rem);
9     font-size:unit(length(@str),rem);
10 }
11
12 //extract()函数
13 @myBorder:2px solid red;
14 #d2{
15     border: @myBorder;
16     color:extract(@myBorder,3); //提取第3个字段
17     font-size: unit(extract(@myBorder,1),rem);
18 }

```

5.5、数学函数

- `ceil`：向上取整。
 - 参数： `number` - 浮点数。
 - 返回： `integer`

示例： `ceil(2.4)`

输出： `3`

- `floor`：向下取整。
 - 参数： `number` - 浮点数。
 - 返回： `integer`

示例： `floor(2.6)`

输出： `2`

- `percentage`：将浮点数转换为百分比字符串。
 - 参数： `number` - 浮点数。
 - 返回： `string`

示例: `percentage(0.5)`

输出: `50%`

- `round`: 四舍五入

- 参数: `number`: 浮点数 `decimalPlaces`: 可选参数, 四舍五入取整的小数点位置, 默认值为0。

- 返回: `number`

示例: `round(1.67)`

输出: `2`

示例: `round(1.67, 1)`

输出: `1.7`

- `sqrt`: 计算一个数的平方根, 原样保持单位。

- 参数: `number` - 浮点数。

- 返回: `number`

示例: `sqrt(25cm)`

输出: `5cm`

示例: `sqrt(18.6%)`

输出: `4.312771730569565%`

- `abs`: 计算数字的绝对值, 原样保持单位。

- 参数: `number` - 浮点数。

- 返回: `number`

示例 #1: `abs(25cm)`

输出: `25cm`

示例 #2: `abs(-18.6%)`

输出: `18.6%`

- `sin`: 正弦函数。处理时会把没有单位的数字认为是弧度值。

- 参数: `number` - 浮点数。

- 返回: `number`

示例:

`sin(1);` // 1弧度角的正弦值 `sin(1deg);` // 1角度角的正弦值 `sin(1grad);` // 1百分度角的正弦值

输出:

`0.8414709848078965;` // 1弧度角的正弦值 `0.01745240643728351;` // 1角度角的正弦值 `0.015707317311820675;` // 1百分度角的正弦值

- asin: 反正弦函数。返回以弧度为单位的角度, 区间在 $-\pi/2$ 到 $\pi/2$ 之间。

- 参数: `number` - 区间在 $[-1, 1]$ 之间的浮点数。
- 返回: `number`

示例:

```
asin(-0.8414709848078965) asin(0) asin(2)
```

输出:

```
-1rad 0rad NaNrad
```

- cos: 余弦函数。处理时会把没有单位的数字认为是弧度值。

- 参数: `number` - 浮点数。
- 返回: `number`

示例:

```
cos(1) // 1弧度角的余弦值 cos(1deg) // 1角度角的余弦值 cos(1grad) // 1百分度角的余弦值
```

输出:

```
0.5403023058681398 // 1弧度角的余弦值 0.9998476951563913 // 1角度角的余弦值 0.9998766324816606 // 1百分度角的余弦值
```

- acos: 反余弦函数。返回以弧度为单位的角度, 区间在 0 到 π 之间。

- 参数: `number` - 区间在 $[-1, 1]$ 之间的浮点数。
- 返回: `number`

示例:

```
acos(0.5403023058681398) acos(1) acos(2)
```

输出:

```
1rad 0rad NaNrad
```

- tan: 正切函数。处理时会把没有单位的数字认为是弧度值。

- 参数: `number` - 浮点数。
- 返回: `number`

示例:

```
tan(1) // 1弧度角的正切值 tan(1deg) // 1角度角的正切值 tan(1grad) // 1百分度角的正切值
```

输出:

```
1.5574077246549023 // 1弧度角的正切值 0.017455064928217585 // 1角度角的正切值 0.015709255323664916 // 1百分度角的正切值
```

- atan: 反正切函数。返回以弧度为单位的角度, 区间在 $-\pi/2$ 到 $\pi/2$ 之间。

- 参数: `number` - 浮点数。

- 返回: `number`

示例:

```
atan(-1.5574077246549023) atan(0) round(atan(22), 6) // 22四舍五入保留6位小数后的反正切值
```

输出:

```
-1rad 0rad 1.525373rad;
```

- `pi`: 返回 π (`pi`);

- 参数: `none`

- 返回: `number`

示例: `pi()`

输出: 3.141592653589793

- `pow`: 乘方运算。假设第一个参数为A，第二个参数为B，返回A的B次方。返回值与A有相同单位，B的单位被忽略。

- 参数: `number`: 基数 - 浮点数。 `number`: 幂指数 - 浮点数。

- 返回: `number`

示例:

```
pow(0cm, 0px) pow(25, -2) pow(25, 0.5) pow(-25, 0.5) pow(-25%, -0.5)
```

输出:

```
1cm 0.0016 5 NaN NaN%
```

- `mod`: 取余运算。假设第一个参数为A，第二个参数为B，返回A对B取余的结果。返回值与A有相同单位，B的单位被忽略。这个函数也可以处理负数和浮点数。

- 参数: `number`: 浮点数。 `number`: 浮点数。

- 返回: `number`

示例:

```
mod(0cm, 0px) mod(11cm, 6px); mod(-26%, -5);
```

输出:

```
NaNcm; 5cm -1%;
```

- `min`: 最小值运算。

- 参数: `value1, ..., valueN` - 一个或多个待比较的值。

- 返回: 最小值。

示例: `min(5, 10);`

输出: `5`

示例: `min(3px, 42px, 1px, 16px);`

输出: `1px`

- max：最大值运算。
 - 参数： `value1, ..., valueN` - 一个或多个待比较的值。
 - 返回： the highest value.

示例： `max(5, 10);`

输出： `10`

示例： `max(3%, 42%, 1%, 16%);`

输出： `42%`

5.6、颜色函数

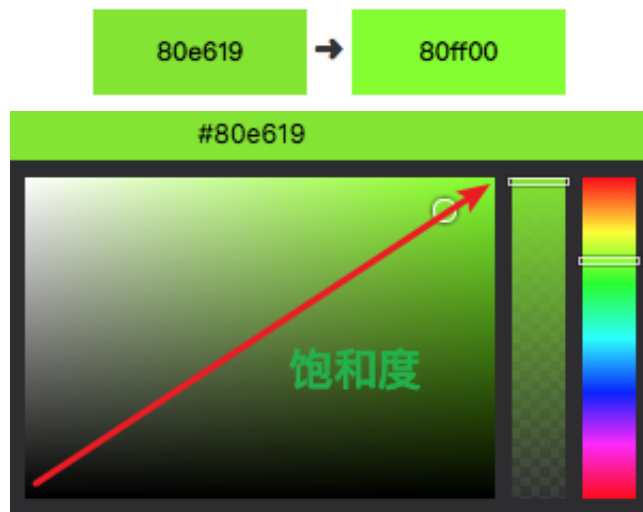
5.6.1、颜色操作函数

颜色值运算有几点注意事项：参数必须单位/格式相同；百分比将作为绝对值处理，比如 10% 增加 10%，结果是 20% 而不是 11%；参数值只能在限定的范围内；they do not wrap around (这一句不清楚意思，可能是指参数值不会在超过范围后自动从另一侧“穿越”回去。)。返回值时，除了十六进制的颜色值 (hex versions) 外将对其他格式做简化处理。

- saturate：增加一定数值的颜色饱和度。
 - 参数： `@color`：颜色对象 `@amount`：百分比 0-100%

示例： `saturate(#80e619, 20%)`

输出： `#80ff00`



- desaturate：降低一定数值的颜色饱和度。
 - 参数： `color`：颜色对象. `amount`：一个百分比 0-100%.

示例： `desaturate(#80e619, 20%)`

输出： `#80cc33`



- lighten:增加一定数值的颜色亮度。

- 参数: `color`: 颜色对象. `amount`: 一个百分比 0-100%.

示例: `lighten(#80e619, 20%)`

输出: `#b3f075`



- `darken`: 降低一定数值的颜色亮度。

- 参数: `color`: 颜色对象. `amount`: 一个百分比 0-100%.

示例: `darken(#80e619, 20%)`

输出: `#4d8a0f`



- `fadein`: 降低颜色的透明度 (或增加不透明度), 令其更不透明。对不透明的颜色无效

- 参数: `color`: 颜色对象. `amount`: 一个百分比 0-100%.

示例: `fadein(rgba(128, 242, 13, 0.5), 10%)`

输出: `rgba(128, 242, 13, 0.6)`

- `fadeout`: 增加颜色的透明度 (或降低不透明度), 令其更透明。对不透明的颜色无效。

- 参数: `color`: 颜色对象. `amount`: 一个百分比 0-100%.

示例: `fadeout(rgba(128, 242, 13, 0.5), 10%)`

输出: `rgba(128, 242, 13, 0.4)`

- `fade`: 给颜色 (包括不透明的颜色) 设定一定数值的透明度。

- 参数: `color`: 颜色对象(A color object) `amount`: 百分比 0-100%

示例: `fade(rgba(128, 242, 13, 0.5), 10%)`

输出: `rgba(128, 242, 13, 0.1)`

- `spin`: 任意方向旋转颜色的色相角度 (hue angle)。

旋转范围 0-360, 超过一周后将从起点开始继续旋转 (+-控制方向), 比如旋转360度与720度是相同的结果。

颜色值会通过RGB格式转换，这个过程不能保留灰色的色相值（灰色没有饱和度，色相值也就没有意义了），因此要确定使用函数的方法能够保留颜色的色相值

例如不要这样使用函数：`@c: saturate(spin(#aaaaaa, 10), 10%);`

而应该用这种方法代替：`@c: spin(saturate(#aaaaaa, 10%), 10);`

- 参数：`color`：颜色对象. `angle`：任意数字表示角度（+ 或 - 表示方向）

示例：

```
color:spin(#f2330d, 30);
```

```
color:spin(#f2330d, -30);
```

输出：

```
color: #f2a60d;
```

```
color: #f20d59;
```



- `mix`：根据比例混合两种颜色，包括计算不透明度。
 - 参数：`color1`：一个颜色对象. `color2`：一个颜色对象. `weight`：可选项：平衡两种颜色的百分比, 默认 50%。

示例：

```
mix(#ff0000, #0000ff, 50%) mix(rgba(100,0,0,1.0), rgba(0,100,0,0.5), 50%)
```

输出：

```
#800080
```

```
rgba(75, 25, 0, 0.75)
```



5.6.2、颜色混合函数

- multiply：分别将两种颜色的红绿蓝 (RGB) 三种值做乘法运算，然后再除以255，输出结果是更深的颜色。

- 参数：@color1：颜色对象(A color object) @color2：颜色对象(A color object)
- 例如：

<code>multiply(#ff6600, #000000);</code>		
<code>ff6600</code>	<code>000000</code>	<code>000000</code>
<code>multiply(#ff6600, #333333);</code>		
<code>ff6600</code>	<code>333333</code>	<code>331400</code>
<code>multiply(#ff6600, #666666);</code>		
<code>ff6600</code>	<code>666666</code>	<code>662900</code>
<code>multiply(#ff6600, #999999);</code>		
<code>ff6600</code>	<code>999999</code>	<code>993d00</code>
<code>multiply(#ff6600, #cccccc);</code>		
<code>ff6600</code>	<code>cccccc</code>	<code>cc5200</code>
<code>multiply(#ff6600, #ffffff);</code>		
<code>ff6600</code>	<code>ffffff</code>	<code>ff6600</code>
<code>multiply(#ff6600, #ff0000);</code>		
<code>ff6600</code>	<code>ff0000</code>	<code>ff0000</code>
<code>multiply(#ff6600, #00ff00);</code>		
<code>ff6600</code>	<code>00ff00</code>	<code>006600</code>
<code>multiply(#ff6600, #0000ff);</code>		
<code>ff6600</code>	<code>0000ff</code>	<code>000000</code>

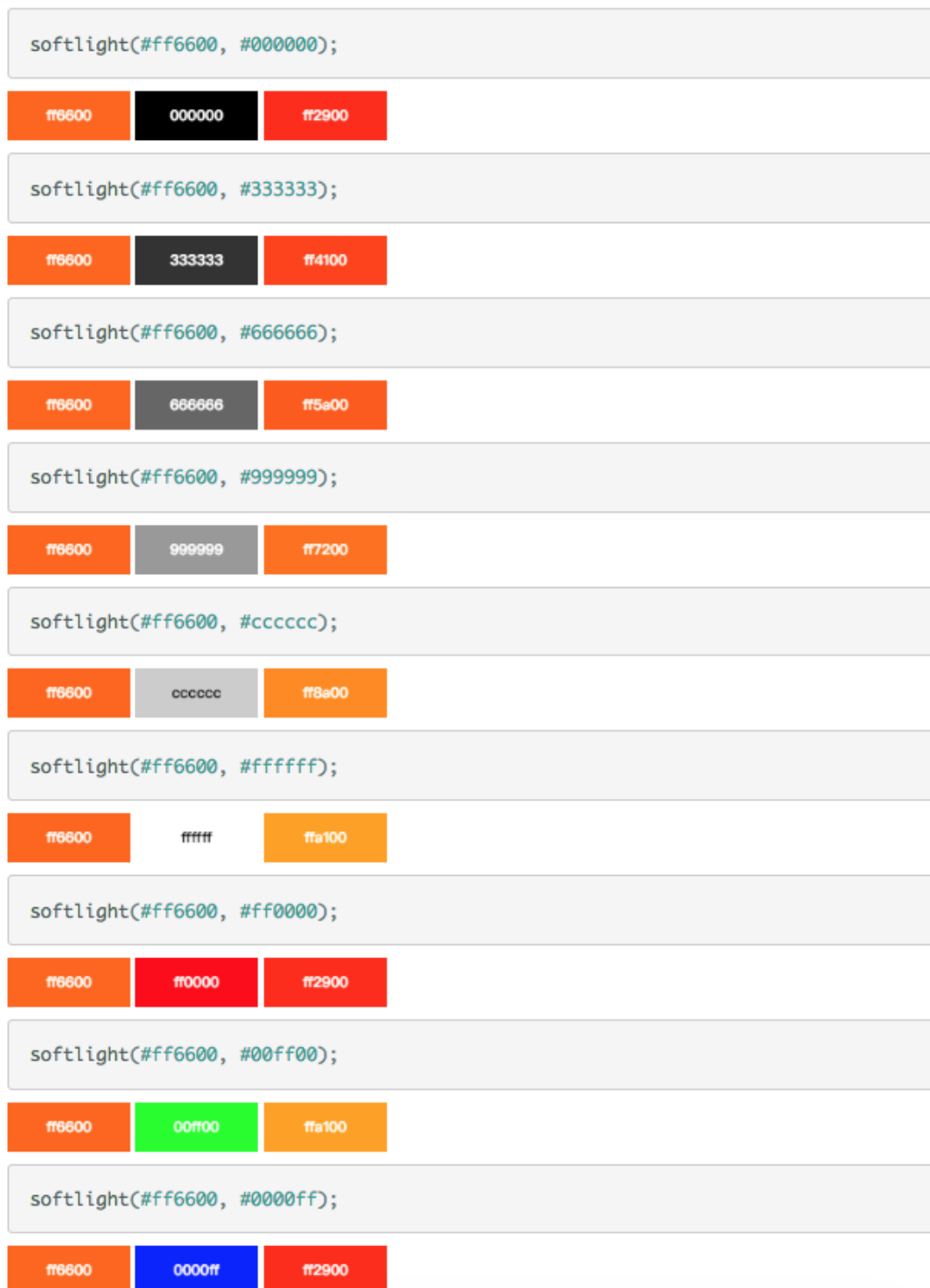
- screen：与 `multiply()` 函数效果相反，输出结果是更亮的颜色。
- 参数：@color1：颜色对象(A color object) @color2：颜色对象(A color object)
- 例如：

screen(#ff6600, #000000);		
ff6600	000000	ff6600
screen(#ff6600, #333333);		
ff6600	333333	ff8533
screen(#ff6600, #666666);		
ff6600	666666	ffa366
screen(#ff6600, #999999);		
ff6600	999999	ffc299
screen(#ff6600, #cccccc);		
ff6600	cccccc	ffe0cc
screen(#ff6600, #ffffff);		
ff6600	ffffff	ffffff
screen(#ff6600, #ff0000);		
ff6600	ff0000	ff6600
screen(#ff6600, #00ff00);		
ff6600	999999	ffff00
screen(#ff6600, #0000ff);		
ff6600	999999	ff66ff

- overlay：结合 multiply() 与 screen() 两个函数的效果，令浅的颜色变得更浅，深的颜色变得更深。**注意**：输出结果由第一个颜色参数决定。
 - 参数：@color1：颜色对象，是用于叠加的颜色，也是结果是更亮还是更暗的决定因素。@color2：颜色对象，被叠加的颜色。
 - **例如**：

overlay(#ff6600, #000000);		
ff6600	000000	ff0000
overlay(#ff6600, #333333);		
ff6600	333333	ff2900
overlay(#ff6600, #666666);		
ff6600	666666	ff5200
overlay(#ff6600, #999999);		
ff6600	999999	ff7a00
overlay(#ff6600, #cccccc);		
ff6600	cccccc	ffa300
overlay(#ff6600, #ffffff);		
ff6600	ffffff	ffe000
overlay(#ff6600, #ff0000);		
ff6600	ff0000	ff0000
overlay(#ff6600, #00ff00);		
ff6600	00ff00	ffcc00
overlay(#ff6600, #0000ff);		
ff6600	0000ff	ff0000

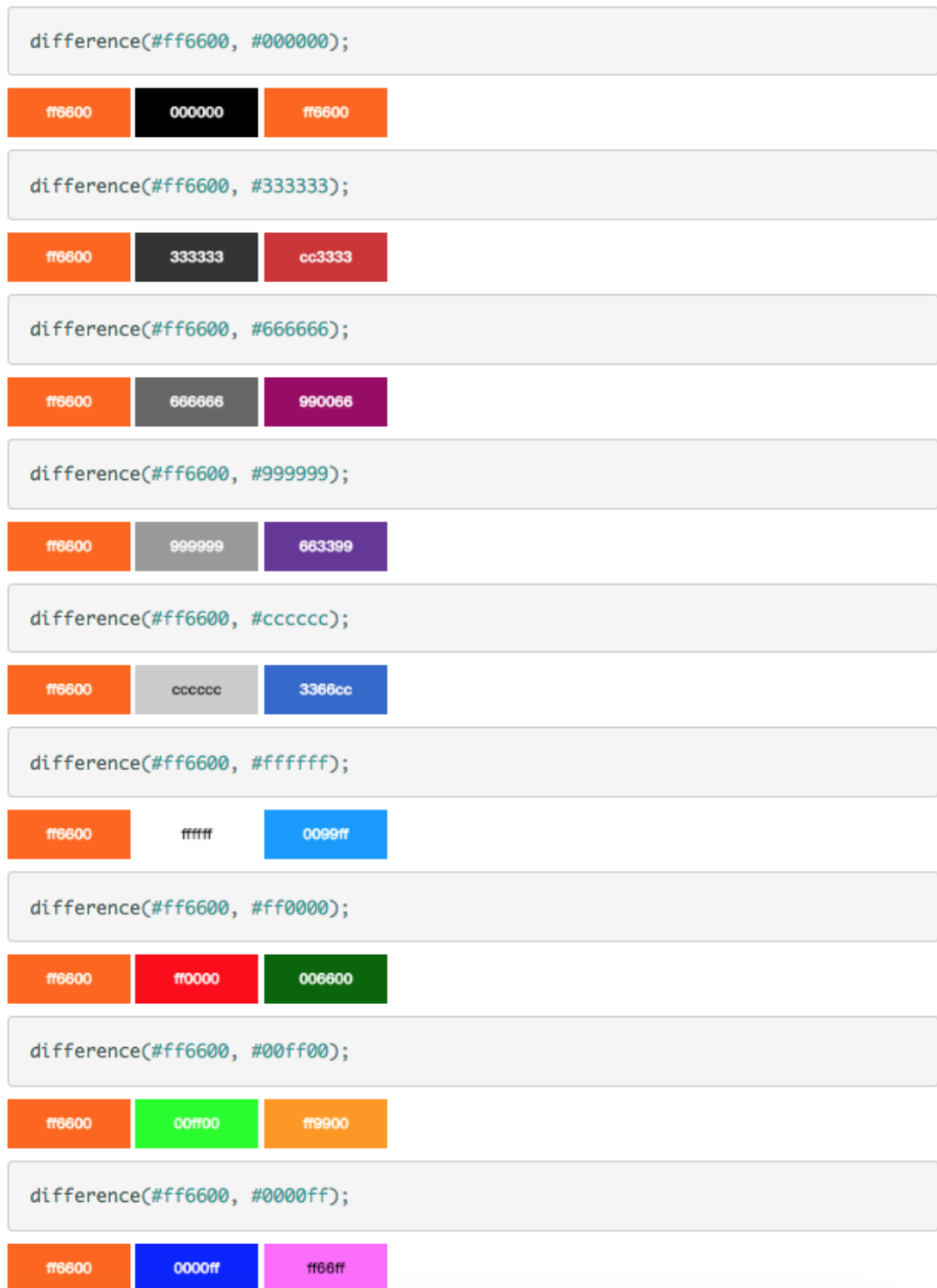
- softlight：与 overlay() 函数效果相似，只是当纯黑色或纯白色作为参数时输出结果不会是纯黑色或纯白色。
 - 参数：@color1: 混合色（光源） @color2: 被混合的颜色
 - 例如:



- **hardlight** : 与 `overlay()` 函数效果相似, 不过由第二个颜色参数决定输出颜色的亮度或黑度, 而不是第一个颜色参数决定。
 - 参数: `@color1`: 混合色 (光源) , `@color2`: 颜色对象, 用于叠加颜色, 也是结果是更亮还是更暗的决定因素。
 - **例如:**

hardlight(#ff6600, #000000);		
ff6600	000000	000000
hardlight(#ff6600, #333333);		
ff6600	333333	662900
hardlight(#ff6600, #666666);		
ff6600	666666	cc5200
hardlight(#ff6600, #999999);		
ff6600	999999	ff8533
hardlight(#ff6600, #cccccc);		
ff6600	cccccc	ff2900
hardlight(#ff6600, #ffffff);		
ff6600	ffffff	ffffff
hardlight(#ff6600, #ff0000);		
ff6600	ff0000	ff0000
hardlight(#ff6600, #00ff00);		
ff6600	00ff00	00ff00
hardlight(#ff6600, #0000ff);		
ff6600	0000ff	0000ff

- difference: 从第一个颜色值中减去第二个（分别计算 RGB 三种颜色值），输出结果是更深的颜色。
 - 参数: @color1: 被减的颜色对象 @color2: 减去的颜色对象
 - 例如:



- exclusion：效果与 `difference()` 函数效果相似，只是输出结果差别更小 (lower contrast)。 (译注：对应Photoshop中的“差值/排除”。)
 - 参数：`@color1`：被减的颜色对象 `@color2`：减去的颜色对象
 - 例如：

exclusion(#ff6600, #000000);		
ff6600	000000	ff6600
exclusion(#ff6600, #333333);		
ff6600	333333	cc7033
exclusion(#ff6600, #666666);		
ff6600	666666	997a66
exclusion(#ff6600, #999999);		
ff6600	999999	668599
exclusion(#ff6600, #cccccc);		
ff6600	cccccc	338fcc
exclusion(#ff6600, #ffffff);		
ff6600	ffffff	0099ff
exclusion(#ff6600, #ff0000);		
ff6600	ff0000	006600
exclusion(#ff6600, #00ff00);		
ff6600	00ff00	ff9900
exclusion(#ff6600, #0000ff);		
ff6600	0000ff	ff66ff

- average：分别对 RGB 的三种颜色值取平均值，然后输出结果。
 - 参数：@color1: 颜色对象(A color object) @color2: 颜色对象(A color object)
 - 例如：

average(#ff6600, #000000);		
ff6600	000000	803300
average(#ff6600, #333333);		
ff6600	333333	994d1a
average(#ff6600, #666666);		
ff6600	666666	b36633
average(#ff6600, #999999);		
ff6600	999999	cc804d
average(#ff6600, #cccccc);		
ff6600	cccccc	e69966
average(#ff6600, #ffffff);		
ff6600	ffffff	ffb380
average(#ff6600, #ff0000);		
ff6600	ff0000	ff3300
average(#ff6600, #00ff00);		
ff6600	00ff00	80b300
average(#ff6600, #0000ff);		
ff6600	0000ff	803380

- negation：与 difference() 函数效果相反，输出结果是更亮的颜色。请注意：效果相反不代表做加法运算。
 - 参数：@color1：被减的颜色对象 @color2：减去的颜色对象
 - 例如：

negation(#ff6600, #000000);		
ff6600	000000	ff6600
negation(#ff6600, #333333);		
ff6600	333333	cc9933
negation(#ff6600, #666666);		
ff6600	666666	99cc66
negation(#ff6600, #999999);		
ff6600	999999	66ff99
negation(#ff6600, #cccccc);		
ff6600	cccccc	33cccc
negation(#ff6600, #ffffff);		
ff6600	ffffff	0099ff
negation(#ff6600, #ff0000);		
ff6600	ff0000	006600
negation(#ff6600, #00ff00);		
ff6600	00ff00	ff9900
negation(#ff6600, #0000ff);		
ff6600	0000ff	ff66ff

6、LESS 模块化

- LESS 同样可以使用 模块方式 定义样式，且 LESS 不关心样式在何处导出，最终编译为CSS 时 都会整合为一个CSS文件

```

1 // base/var.less
2 @white: #fff;
3 @light: #f8f9fa;
4 @gray: #6c757d;
5 @secondary: #6c757d;
```

```

6  @gray-dark: #343a40;
7  @dark: #343a40;
8  @blue: #007bff;
9  @indigo: #6610f2;
10 @purple: #6f42c1;
11 @pink: #e83e8c;
12 @red: #dc3545;
13 @orange: #fd7e14;
14 @yellow: #ffc107;
15 @green: #28a745;
16 @teal: #20c997;
17 @cyan: #17a2b8;
18
19 @size: 20px;

```

```

1  // @import "../base/var.less";
2  // @import "../base/var"; //less 导入可以省略扩展名
3  div{
4      color: @red;
5      font-size: @size;
6  }
7  @import "../base/var"; //less 导入可以写在文件的任何位置

```

7、CSS 编写的一点建议

- 该建议是 **笔者** 在实际开发项目，和参考大量 前端框架 开发规范后，总结出得相关编写习惯。

- BEM命名规范

BEM 是由Yandex团队提出的一种CSS Class 命名方法，旨在更好的创建CSS模块。

BEM的意思就是块（block）、元素（element）、修饰符（modifier）。

- block: 可以理解为一个区域、一个组件或者一个块级元素，具体如何区分需要根据实际情况具体分析；
- block_element: 就是一个上面的block里面的元素，比如说导航（nav: block）里面有a标签（a: element）就是一个元素，block与element使用两个下划线链接；
（结合各大框架，建议使用一个 _）
- block_element--modifier: 我的理解是状态或属性。比如element里面的a标签，它有active、hover、normal三种状态，这三种状态就是modifier。midifier是使用两个"--"中横线连接 **（结合各大框架，建议使用一个 _）**

```

1  <!-- HTML结构 -->
2  <nav class="nav">
3      <a href="#" class="nav_item nav_item_active">当前页: active</a>
4      <a href="#" class="nav_item nav_item_hover">假设鼠标在这里要加个hover
      的class</a>
5  </nav>

```

```

1 // LESS写法
2 .nav{
3     &_item{
4         &_active{
5             }
6         &_hover{
7             }
8     }
9 }

```

```

1 /* 编译后的css */
2 .nav{ }
3 .nav_item{ }
4 .nav_item_active{ }
5 .nav_item_hover{ }

```

- 合理的注释：一份可读性的CSS|LESS|SASS 必须有一份说明，一个文件，一个函数都需要一份说明。对于一份LESS|SASS文件，你至少需要说明两点，是公用还是私有、哪个页面哪个部分
- 动态样式语言，首先需要有一份variables（变量表），为提高开发效率打下基础，也是确保页面一致性的基础。
- 动态样式语言 需要合理的模块化，模块化在js中经常听到，对于css来说，模块化对于易读性和可维护性同样重要。
 - 多文件夹：分类存放动态语言文件。例如：将variables、mixin、公共样式、私有样式分成多个文件夹存放
 - 多文件：同一个文件夹的动态样语言文件可以按模块、功能等等分成多个文件，最终用@import 导入

```

1 |LESS
2 |variables           //基本变量
3 |   -colors.less
4 |   .....
5 |mixin               //函数
6 |   .....
7 |common              //公用
8 |   |header
9 |   .....
10 |   |aside
11 |       -list.less
12 |       -nav.less
13 |       -base.less
14 |compoment           //组件样式
15 |   |dropdown
16 |   .....
17 |   |lightbox

```

```
18         .....
19     |page
20     |index          //首页
21     -ad.less        //广告样式
22     -content.less   //内容信息
23     -info.less       //个人信息样式
24     -base.less       //index样式, @import 'ad';@import
25 'content';@import 'info';
26     |write
27     .....
28     |preview
29     -aside.less      //preview页面独有侧边栏
30     |about
31     .....
32 |main.less          //导入所需要的样式, 最终生成一个main.css
```