**一、 css预编译**

常见的css预编译器有三种：less 、 sass 、 stylus。  
Bootstrop使用的是less。  
**在线编译**  
[http://tools.jb51.net/code/less2css](https://links.jianshu.com/go?to=http%3A%2F%2Ftools.jb51.net%2Fcode%2Fless2css)

**二、 Less**

Less是一门css预处理语言，他扩展了css语言，增加了变量、Mixin、函数等特性，使css更容易维护和扩展。他不是一个直接使用的语言，而是一个生成css的语言。Less可以运行在Node或浏览器端。

Css计算：cacl（）

**三、 使用方法**

见参考：LESS安装和使用

**四、Less语法**

**1．变量**

**1) 普通变量**

less 以@开头定义变量，并且使用时直接键入@名称。注意：作为属性值的变量不能添加大括号（{}）

@color:#333;

Div{

Background：@color；

}

Less其变量只能定义一次，不能重复定义，否则后面的会类似与js的变量提升，覆盖前面的变量。

使用变量设置css，也方便代码的维护。

**2) 选择器变量**

选择器变量

让选择器变成一个动态的。

作为选择器的变量在使用的时候需要添加大括号（{}），变量的前面可以添加选择操作符。

/\* Less \*/

@mySelector: #wrap;

@Wrap: wrap;

@{mySelector}{ //变量名 必须使用大括号包裹

color: #999;

width: 50%;

}

.@{Wrap}{

color:#ccc;

}

#@{Wrap}{

color:#666;

}

/\* 生成的 CSS \*/

#wrap{

color: #999;

width: 50%;

}

.wrap{

color:#ccc;

}

#wrap{

color:#666;

}

**3) 属性变量**

可以让我在书写的时候少写很多内容，属性变量使用的时候也需要添加大括号（{}）。

/\* Less \*/

@borderStyle: border-style;

@Soild:solid;

#wrap{

@{borderStyle}: @Soild;//变量名 必须使用大括号包裹

}

/\* 生成的 CSS \*/

#wrap{

border-style:solid;

}

**4) Url变量**

项目结构改变时，可以方便我们修改，目的是为了方便项目的维护。

/\* Less \*/

@images: "../img";//需要加引号

body {

background: url("@{images}/dog.png");//变量名 必须使用大括号包裹

}

/\* 生成的 CSS \*/

body {

background: url("../img/dog.png");

}

**5) 申明变量**

格式：

定义：@name：{

Key：value；

Key：value；

}

使用：@name{}；

案例：

/\* Less \*/

@background: {background:red;};

#main{

@background();

}

@Rules:{

width: 200px;

height: 200px;

border: solid 1px red;

};

#con{

@Rules();

}

/\* 生成的 CSS \*/

#main{

background:red;

}

#con{

width: 200px;

height: 200px;

border: solid 1px red;

}

**6) 变量运算**

规则：

加减法时，以第一个数据的单位为基准。

乘除法时，注意单位一定要统一。

案例：

/\* Less \*/

@width:300px;

@color:#222;

#wrap{

width:@width-20;

height:@width-20\*5;

margin:(@width-20)\*5;

color:@color\*2;

background-color:@color + #111;

}

/\* 生成的 CSS \*/

#wrap{

width:280px;

height:200px;

margin:1400px;

color:#444;

background-color:#333;

}

**7) 变量的作用域**

就近原则，不是指代码的位置，而是指代码的层级结构。

如果是同一级后面的生效，类似于提升。

不同级的变量，距离最近的生效。距离是指定义时的位置。

**8) 用变量的定义变量**

解析的顺序是从后向前逐层解析。

/\* Less \*/

@fnord: "I am fnord.";

@var: "fnord";

#wrap::after{

content: @@var; //将@var替换为其值 content:@fnord;

}

/\* 生成的 CSS \*/

#wrap::after{

content: "I am fnord.";

}

image.png

**2．嵌套**

**1) &符号，表示的是上1级选择器的名字。**

/\* Less \*/

#header{

&:after{ //注意：不能省略&，如果省略会给每一个子元素添加一个after。

content:"Less is more!";

}

.title{

font-weight:bold;

}

&\_content{//理解方式：直接把 & 替换成 #header

margin:20px;

}

}

/\* 生成的 CSS \*/

#header::after{

content:"Less is more!";

}

#header .title{ //嵌套了

font-weight:bold;

}

#header\_content{//没有嵌套！

margin:20px;

}

image.png

**2) 嵌套覆盖原有样式**

div{

width:200px;

height:200px;

background:red;

.show{

display: none;

}

}

.show{

display: block; //被覆盖，只有在div使用show时才覆盖

}

**3．媒体查询**

之前媒体查询的格式是：先分屏然后在每一个媒体查询中设置样式。

@media only screen and {max-width：1200px；}{

Div{}

}

@media only screen and {min-width:992px;}{

Div{}

}

@media only screen and {min-width:768px}{

Div{}

}

less提供给我们更加方便的方式：

/\* Less \*/

#main{

//something...

@media screen{

@media (max-width:768px){

width:100px;

}

}

@media tv {

width:2000px;

}

}

/\* 生成的 CSS \*/

@media screen and (max-width:768px){

#main{

width:100px;

}

}

@media tv{

#main{

width:2000px;

}

}

/\* 生成的 CSS \*/

@media screen and (max-width:768px){

#main{

width:100px;

}

}

@media tv{

#main{

width:2000px;

}

}

image.png

* 唯一缺点：每一个元素都会编译出自己@media声明，并不会合并

**4．混合方法**

**1) 无参数方法 方法如声明的集合，使用值直接键入名称即可。**

要点：“.”与“#”都可以作为方法前缀。可以不适用小括号，但是为了避免css格式混淆，建议加上小括号“（）”。

.card(){

//something....

}

#wrap{

.card();

}

**2) 具体参数方法**

Less可以传递参数

.setSize(@width\_size,@height\_size){

width:@width\_size;

height:@height\_size;

}

**3) 默认参数方法**

Less 可以使用默认参数，如果没有传参数，那么将使用默认参数。 @arguments 犹如 JS 中的 arguments 指代的是全部参数。 传的参数中 必须带着单位。

/\* Less \*/

.border(@a:10px,@b:50px,@c:30px,@color:#000){

border:solid 1px @color;

box-shadow: @arguments;//指代的是 全部参数

}

#main{

.border(0px,5px,30px,red);//必须带着单位

}

#wrap{

.border(0px);

}

#content{

.border;//等价于 .border()

}

/\* 生成的 CSS \*/

#main{

border:solid 1px red;

box-shadow:0px,5px,30px,red;

}

#wrap{

border:solid 1px #000;

box-shadow: 0px 50px 30px #000;

}

#content{

border:solid 1px #000;

box-shadow: 10px 50px 30px #000;

}

**4) 不定参**

不确定参数的个数

案例：

/\* Less \*/

.boxShadow(...){

box-shadow: @arguments;

}

.textShadow(@a,...){

text-shadow: @arguments;

}

#main{

.boxShadow(1px,4px,30px,red);

.textShadow(1px,4px,30px,red);

}

/\* 生成后的 CSS \*/

#main{

box-shadow: 1px 4px 30px red;

text-shadow: 1px 4px 30px red;

}

**5) 方法的匹配模式**

类似于多态。有点类似于switch case

同一个方法名的多个方法，由于传入的参数不同而实现不同的功能。

案例：

/\* Less \*/

.triangle(top,@width:20px,@color:#000){

border-color:transparent transparent @color transparent ;

}

.triangle(right,@width:20px,@color:#000){

border-color:transparent @color transparent transparent ;

}

.triangle(bottom,@width:20px,@color:#000){

border-color:@color transparent transparent transparent ;

}

.triangle(left,@width:20px,@color:#000){

border-color:transparent transparent transparent @color;

}

.triangle(@\_,@width:20px,@color:#000){

border-style: solid;

border-width: @width;

}

#main{

.triangle(left, 50px, #999)

}

/\* 生成的 CSS \*/

#main{

border-color:transparent transparent transparent #999;

border-style: solid;

border-width: 50px;

}

image.png

**6) 方法的命名空间**

有一个嵌套关系

/\* Less \*/

#card(){

background: #723232;

.d(@w:300px){

width: @w;

#a(@h:300px){

height: @h;//可以使用上一层传进来的方法

width: @w;

}

}

}

#wrap{

#card > .d > #a(100px); // 父元素不能加 括号

}

#main{

#card .d();

}

#con{

//不得单独使用命名空间的方法

//.d() 如果前面没有引入命名空间 #card ，将会报错

#card; // 等价于 #card();

.d(20px); //必须先引入 #card

}

/\* 生成的 CSS \*/

#wrap{

height:100px;

width:300px;

}

#main{

width:300px;

}

#con{

width:20px;

}

案例2

.div{

background:red;

.size(@width){

width:@width;

height:200px;

border:1px solid #ccc;

.a(@height){

height:@height;

width:@width;

}

}

}

a{

display: block;

.div>.size(200px);

.a(200px);

}

在 CSS 中> 选择器，选择的是儿子元素，就是必须与父元素有直接血源的元素。 - 在引入命令空间时，如使用 > 选择器，父元素不能加括号。 - 不得单独使用命名空间的方法必须先引入命名空间，才能使用其中方法。 - 子方法可以使用上一层传进来的方法

**7) 条件语句**

Less没有if / else 但是less具有一个when，and，not与“，”语法。

/\* Less \*/

#card{

// and 运算符 ，相当于 与运算 &&，必须条件全部符合才会执行

.border(@width,@color,@style) when (@width>100px) and(@color=#999){

border:@style @color @width;

}

// not 运算符，相当于 非运算 !，条件为 不符合才会执行

.background(@color) when not (@color>=#222){

background:@color;

}

// , 逗号分隔符：相当于 或运算 ||，只要有一个符合条件就会执行

.font(@size:20px) when (@size>50px) , (@size<100px){

font-size: @size;

}

}

#main{

#card>.border(200px,#999,solid);

#card .background(#111);

#card > .font(40px);

}

/\* 生成后的 CSS \*/

#main{

border:solid #999 200px;

background:#111;

font-size:40px;

}

**8) 条件运算符**

比较运算有：> >= = =< <

=代表是等于

除去关键字true以外的值其他都会被默认为fales

**9) 循环语法**

Less并没有提供一个for等循环的方法但是可以使用递归的方法实现。

案例：

/\* Less \*/

.generate-columns(4);

.generate-columns(@n, @i: 1) when (@i =< @n) {

.column-@{i} {

width: (@i \* 100% / @n);

}

.generate-columns(@n, (@i + 1));

}

/\* 生成后的 CSS \*/

.column-1 {

width: 25%;

}

.column-2 {

width: 50%;

}

.column-3 {

width: 75%;

}

.column-4 {

width: 100%;

}

**10) 方法中的important**

方法使用了important，相当于这个方法中的每一个属性都设置了一遍important，不允许覆盖。

/\* Less \*/

/\* Less \*/

.border{

border: solid 1px red;

margin: 50px;

}

#main{

.border() !important;

}

/\* 生成后的 CSS \*/

#main {

border: solid 1px red !important;

margin: 50px !important;

}

**5．Less注释**

Less的注释语法类似于js

//单行注释

/\* ---\*/多行注释

**6．属性的拼接语法**

+代表的是逗号，+\_代表的是空格。

**1) 空格**

案例：

/\* Less \*/

.Animation() {

transform+\_: scale(2);

}

.main {

.Animation();

transform+\_: rotate(15deg);

}

/\* 生成的 CSS \*/

.main {

transform: scale(2) rotate(15deg);

}

**2) 逗号**

/\* Less \*/

.boxShadow() {

box-shadow+: inset 0 0 10px #555;

}

.main {

.boxShadow();

box-shadow+: 0 0 20px black;

}

/\* 生成后的 CSS \*/

.main {

box-shadow: inset 0 0 10px #555, 0 0 20px black;

}

**7． 继承(扩展)**

extend是less的一个**伪类**。它可以继承所匹配声明中的全部样式。

/\* Less \*/

.animation{

transition: all .3s ease-out;

.hide{

transform:scale(0);

}

}

#main{

&:extend(.animation);

}

#con{

&:extend(.animation .hide);

}

/\* 生成后的 CSS \*/

.animation,#main{

transition: all .3s ease-out;

}

.animation .hide , #con{

transform:scale(0);

}

**1) all全局搜索替换**

匹配所有的声明

/\* Less \*/

#main{

width: 200px;

}

#main {

&:after {

content:"Less is good!";

}

}

#wrap:extend(#main all) {}

/\* 生成的 CSS \*/

#main,#wrap{

width: 200px;

}

#main:after, #wrap:after {

content: "Less is good!";

}

**2) 扩展的注意事项**

选择器和扩展之间 是允许有空格的：pre:hover :extend(div pre).

可以有多个扩展: pre:hover:extend(div pre):extend(.bucket tr) - 注意这 与 pre:hover:extend(div pre, .bucket tr)一样。

扩展必须在最后 : pre:hover:extend(div pre).nth-child(odd)。 如果一个规则集包含多个选择器，所有选择器都可以使用extend关键字。

**8. 导入**

在less文件中可以引入其他的less文件。使用关键字import。

①　导入less文件，可以省略后缀。

import “index.less”;

import “index”;

②　@import可以放在任何地方

③　reference

引入一个less文件，但不会编译它。

@import (reference) “bootstrap.less”;

#wrap:extend(.navbar all){}

@import (reference)使用导入外部文件，只能在less中使用，主要是为了继承和方法，最终编译的时候里面的内容不会产生css。

④　once

@import语句的默认行为。表明相同的文件只会被导入一次，而随后的导入文件的重复代码都不会解析。

@import (once) "foo.less";

@import (once) "foo.less"; // this statement will be ignored

⑤　multiple

使用@import (multiple)允许导入多个同名文件。

\* Less \*/

// file: foo.less

.a {

color: green;

}

// file: main.less

@import (multiple) "foo.less";

@import (multiple) "foo.less";

/\* 生成后的 CSS \*/

.a {

color: green;

}

.a {

color: green;

}

**9. 避免编译**

#main{

width:~'calc(300px-30px)';

}

/\* 生成后的 CSS \*/

#main{

width:calc(300px-30px);

}

**10. less中可以使用js**

less本身是使用js实现的，所以在less中可以使用js。

Js的代码写在字符串模板里。

/\* Less \*/

@content:`"aaa".toUpperCase()`;

#randomColor{

@randomColor: ~"rgb(`Math.round(Math.random() \* 256)`,`Math.round(Math.random() \* 256)`,`Math.round(Math.random() \* 256)`)";

}

#wrap{

width: ~"`Math.round(Math.random() \* 100)`px";

&:after{

content:@content;

}

height: ~"`window.innerHeight`px";

alert:~"`alert(1)`";

#randomColor();

background-color: @randomColor;

}

/\* 生成后的 CSS \*/

// 弹出 1

#wrap{

width: 随机值（0~100）px;

height: 743px;//由电脑而异

background: 随机颜色;

}

#wrap::after{

content:"AAA";

}

**六、Less的函数**

参考手册：

[http://lesscss.cn/functions/](https://links.jianshu.com/go?to=http%3A%2F%2Flesscss.cn%2Ffunctions%2F)

[https://www.w3cschool.cn/less/namespaces\_accessors.html](https://links.jianshu.com/go?to=https%3A%2F%2Fwww.w3cschool.cn%2Fless%2Fnamespaces_accessors.html)

**1. 判断类型**

Isnumber():判断是否为数字

Iscolor():判断是否是颜色

Isurl():判断是否是路径

**2. 颜色**

saturate:增加一定数值的颜色饱和度

lighten:增加一定数值的颜色亮度

darken:降低一定数值的颜色亮度

fade:给颜色设置一定数值的透明度

mix:根据比例混合两种颜色

**3. 数学函数**

ceil

floor

round

sqrt

abs

pow