爱创课堂前端培训

# CSS预编译语言

第1天课堂笔记（本课程共1天）

班级：北京前端训练营19期

讲师：李兰波

日期：2019年1月31日

爱创课堂官网 ：[www.icketang.com](http://www.icketang.com)

# 复习：

工程化：打包、压缩、编译等一系列的事情。

GULP：这是一个NodeJS第三方模块。可以定义任务，通过任务实现工程化的内容。

安装：

第一次全局安装：npm install gulp@3.9.1 -g 负责提供命令行的命令 如果安装完毕之后，依旧无法使用gulp命令，此时要去查看全局的安装目录路径是否在环境变量中配置过。

第二次本地安装：npm install gulp@3.9.1 负责提供模块文件 这一次的安装是要安装到工程目录下。

使用：

var gulp = require("gulp");

gulp.task("default", function() {} || [])

命令调用： gulp

注意：在命令调用的时候，一定要先切换到该文件所在的目录 并且一定要将代码放入一个叫: gulpfile.js的文件内

gulp自带的功能：

task: 用于定义任务 第一个参数是任务名称 第二个参数是任务执行内容 第二个参数可以是函数体 也可以是其它任务的名称组成的数组 注：必须要有一个入口任务，也就是default任务

src：用于定位资源 参数是字符串或者是数组

字符串：纯文件路径 此时，只能够指定一个文件

字符串：globs 此时，通过glob表达式去定位文件

pipe: 用于定义操作

dest: 用于发布文件到磁盘上 将文件的内存状态转换为物理磁盘状态

watch: 用于监听一个、些文件的保存行为。一旦保存，立即执行一个、些任务。

glob：一种定位文件的字符串表示法 类似于正则表达式

\*： 单层级全部

\*\*： 多层级全部

[]: 范围 位于范围中的会被选中 只选中一个字符

[^]: 范围取反 位于范围中的都不要

？: 任意字符

插件：gulp自身只能够将文件转换状态，其它功能要依赖插件。

压缩JS:

插件名称： gulp-uglify

使用: gulp.src(xxx).pipe(uglify()).pipe(gulp.dest(xxx));

压缩CSS：

插件名称： gulp-clean-css

使用: gulp.src(xxx).pipe(clean()).pipe(gulp.dest(xxx));

压缩HTML:

插件名称： gulp-minify-html

使用: gulp.src(xxx).pipe(minify()).pipe(gulp.dest(xxx));

压缩图片：

插件名称：gulp-imagemin

使用：gulp.src(xxx).pipe(imagemin()).pipe(gulp.dest(xxx));

重命名：

插件名称：gulp-rename

使用：gulp.src(xxx).pipe(rename({dirname：路径名, prefix: 前缀, basename: 文件名, suffix: 后缀, extname: 拓展名})).pipe(gulp.dest(xxx))

合并：

插件名称： gulp-concat

使用：gulp.src(xxx).pipe(concat(合并之后的文件名)).pipe(gulp.dest(xxx))

语法校验：

插件名称：gulp-jslint

使用：

gulp.src(xxx).pipe(jsLint()).pipe(jslint.reporter("default"));

Webpack: 这是一个NodeJS第三方模块，也是一个模块化打包工具。

安装：

第一次安装 全局安装 负责提供命令

npm install webpack webpack-cli -g

第二次安装 本地安装 负责提供模块文件

npm install webpack webpack-cli

配置文件： webpack.config.js

配置：

因为是模块化开发，所以该js文件要向外暴露一个配置对象

module.exports = {

// 第一种配置方式：单页面应用程序

entry: "入口模块的地址",

// 第二种配置方式：多页面应用程序

entry: {

最终打包的文件名: 打包该文件的入口

}，

output: {

path: 打包之后的文件的目录必须是绝对路径

filename: 打包之后的文件名

}，

module: {

// rules 用于配置加载机 2.x版本之前 叫loaders

rules: [

// 数组中的每一项都是一个加载机配置项

{

test: // 这是一个正则表达式 用于匹配处理的文件

loaders: 字符串 用于书写加载文件时 所需要的模块名称

}

]

}

}  
CSS加载机：

{

test: /\.css$/,

loaders: "style-loader!css-loader"

}

图片加载机:

{

test: /\.(png|jpg|gif)$/,

loaders: "file-loader?limit=4096"

}

webpack-dev-server: 这是一个可以开启服务器的工具

安装需要两次 : npm install webpack-dev-server -g 提供全局命令

npm install webpack-dev-server 提供本地模块文件

当安装完毕之后，就可以使用webpack-dev-server来代替webpack命令

不同的是，webpack-dev-server会开启一个服务器 而且不会生成打包文件。

# CSS预编译语言

简单来说，编译之前不是CSS，编译之后就是CSS。

# Less

这是一个CSS预编译语言。

## 2.1 体验Less

|  |
| --- |
| 1. .box1 { 2. width: 400px; 3. height: 400px; 4. border: 1px solid #ccc; 5. margin: 0 auto; 6. .box2 { 7. width: 300px; 8. height: 300px; 9. border: 1px solid #ddd; 10. margin: 0 auto; 11. .box3 { 12. width: 200px; 13. height: 200px; 14. border: 1px solid #eee; 15. margin: 0 auto; 16. } 17. } 18. } |

我们发现，子元素的选择器与样式都写在父元素的选择器大括号内。这样，就可以解决选择器的权重问题。但是它不被浏览器支持。所以，这就不是CSS。而是Less。只要我们能够将Less转换成CSS，就可以了。

这个转换过程，叫做“编译”。

Less是编译之前发生的。所以叫做预编译语言。

## 2.2 转换方式

前端编译1：转换过程发生在前端。

|  |
| --- |
| 1. <style type="text/less"> 2. .box1 { 3. width: 400px; 4. height: 400px; 5. border: 1px solid #ccc; 6. margin: 0 auto; 7. .box2 { 8. width: 300px; 9. height: 300px; 10. border: 1px solid #ddd; 11. margin: 0 auto; 12. .box3 { 13. width: 200px; 14. height: 200px; 15. border: 1px solid #eee; 16. margin: 0 auto; 17. } 18. } 19. } 20. </style> 21. <script type="text/javascript" src="./less.js"></script> |

前端编译2：外链式

此时有两个要求

第一个是必须将link标签的rel从原来的stylesheet修改为stylesheet/less

第二个是必须处于服务器环境下

|  |
| --- |
| 1. <link rel="stylesheet/less" href="./reset.less"> 2. <script type="text/javascript" src="./less.js"></script> |

后端编译：

node编译 需要下载less模块 也需要安装到全局

下载: 会向全局添加一个lessc命令 该命令可以将less文件转换成css文件

|  |
| --- |
| 1. npm install less -g |

转换：lessc less文件名

|  |
| --- |
|  |

转换：lessc less文件名 css文件名

|  |
| --- |
|  |

此时，会真的生成css文件

|  |
| --- |
|  |

第二种后端转换方式：工程化工具去转换 我们可以使用gulp去转换 也可以使用webpack去转换 我们选择的是webpack

需要的模块： less-loader less

|  |
| --- |
| 1. module.exports = { 2. entry: "./main.js", 3. output: { 4. path: \_\_dirname, 5. filename: "./bundle.js" 6. }, 7. module: { 8. rules: [ 9. { 10. test: /\.less$/, 11. loaders: "style-loader!css-loader!less-loader" 12. } 13. ] 14. } 15. } |

## 2.3 语法

### 2.3.1 变量

Less中允许定义变量。

语法： @variable: value;

例如： @a: 10;

等价于JS中的: var a = 10;

|  |
| --- |
| 1. /\* 整个Less空间 在被执行的时候是一个JS环境 可以定义变量 \*/ 2. @a: 10px; 3. /\*等价于 var a = 10;\*/ 4. div { 5. width: @a; 6. } |

编译之后：

|  |
| --- |
| 1. /\* 整个Less空间 在被执行的时候是一个JS环境 可以定义变量 \*/ 2. /\*等价于 var a = 10;\*/ 3. div { 4. width: 10px; 5. } |

### 2.3.2 混合

Less中允许混合语法。意思就是将一个选择器的样式继承给另外一个选择器。

demo:

|  |
| --- |
| 1. .box { 2. width: 100px; 3. height: 100px; 4. background-color: red; 5. } 6. p { 7. .box; 8. } |

编译之后：

|  |
| --- |
| 1. .box { 2. width: 100px; 3. height: 100px; 4. background-color: red; 5. } 6. p { 7. width: 100px; 8. height: 100px; 9. background-color: red; 10. } |

### 2.3.3 方法

混合是全部继承，包括属性名，包括属性值。

有些时候，我们希望只拥有相同的属性名 但是属性值不同。

这时候我们可以使用方法。

demo:

|  |
| --- |
| 1. // 定义方法 2. .box(@w, @h, @c) { 3. width: @w; 4. height: @h; 5. background-color: @c; 6. } 7. p { 8. // 调用方法 9. .box(200px, 100px, red); 10. } 11. div { 12. // 调用方法 13. .box(100px, 200px, blue); 14. } |

编译之后：

|  |
| --- |
| 1. p { 2. width: 200px; 3. height: 100px; 4. background-color: red; 5. } 6. div { 7. width: 100px; 8. height: 200px; 9. background-color: blue; 10. } |

### 2.3.4 方法的参数默认值

Less中允许给方法定义默认值 也就是当调用方法的时候不传递参数时，也会有一个值。

|  |
| --- |
| 1. .box(@w:100px, @h:100px, @c:orange) { 2. width: @w; 3. height: @h; 4. background-color: @c; 5. } 6. p { 7. .box(); 8. } 9. div { 10. .box(100px, 200px, blue); 11. } |

生成：

|  |
| --- |
| 1. p { 2. width: 100px; 3. height: 100px; 4. background-color: orange; 5. } 6. div { 7. width: 100px; 8. height: 200px; 9. background-color: blue; 10. } |

### 2.3.5 判定语法

Less出现的比较早，所以没有实现if语句 但是有类似的语法

.box (@w, @h, @c) when(statement) {

}

.box(@w, @h, @c) when not(statement) {

}

.box (@w, @h, @c) when (statement) and(statement) {

}

demo:

|  |
| --- |
| 1. .box(@w, @h, @c) { 2. width: @w; 3. height: @h; 4. background-color: @c; 5. } 6. /\*当宽度超过200px时，颜色使用红色 当宽度超过300px时 颜色使用蓝色\*/ 7. .box(@w, @h, @c) when(@w > 200px) { 8. background-color: red; 9. } 10. .box(@w, @h, @c) when(@w < 200px) { 11. background-color: green; 12. } 13. .box(@w, @h, @c) when(@w > 300px) { 14. background-color: blue; 15. } 16. p { 17. .box(100px, 200px, orange); 18. } 19. div { 20. .box(201px, 200px, pink); 21. } 22. section { 23. .box(301px, 200px, purple); 24. } |

编译之后：

|  |
| --- |
| 1. /\*当宽度超过200px时，颜色使用红色 当宽度超过300px时 颜色使用蓝色\*/ 2. p { 3. width: 100px; 4. height: 200px; 5. background-color: orange; 6. background-color: green; 7. } 8. div { 9. width: 201px; 10. height: 200px; 11. background-color: pink; 12. background-color: red; 13. } 14. section { 15. width: 301px; 16. height: 200px; 17. background-color: purple; 18. background-color: red; 19. background-color: blue; 20. } |

### 2.3.6 运算

|  |
| --- |
| 1. @w: 100px; 2. p { 3. /\*乘法运算\*/ 4. width: @w \* 10; /\* 1000px \*/ 5. width: @w \* 100px; /\* 10000px \*/ 6. width: @w \* 9in; /\* 900px \*/ 7. width: @w / 10; /\* 10px \*/ 8. width: @w / 100px; /\* 1px \*/ 9. width: @w / 9in; /\* 11.111111111px \*/ 10. /\*乘法总结：当两个运算数有一个有单位时 以该单位为准 如果都有单位 以第一个运算数的单位为准\*/ 11. /\*除法总结：当两个运算数有一个有单位时 以该单位为准 如果都有单位 以第一个运算数的单位为准\*/ 13. /\*加法与减法\*/ 14. width: @w + 1; /\* 101px \*/ 15. width: @w + 11px; /\* 111px \*/ 16. width: @w + 1in; /\* 196px \*/ 17. width: 1in + @w; /\* 2.xxxxin \*/ 18. width: @w - 1; /\* 99px \*/ 19. width: @w - 11px; /\* 89px \*/ 20. width: @w - 1in; /\* 4px \*/ 21. width: 1in - @w; /\* -0.04166667in \*/ 22. /\*加法总结： 当两个运算数有一个有单位时 以该单位为准 如果都有单位 会向第一个运算数的单位进行转换\*/ 23. /\*减法总结： 当两个运算数有一个有单位时 以该单位为准 如果都有单位 会向第一个运算数的单位进行转换\*/ 24. } |

### 2.3.7 内置函数

内置数学函数 其实就是JS中的Math对象的一些方法。

|  |
| --- |
| 1. .box { 2. width: ceil(208.9px); /\* 209px \*/ 3. width: floor(208.9px); /\* 208px \*/ 4. width: round(109.1px); /\* 109px \*/ 5. width: round(109.5px); /\* 110px \*/ 6. width: percentage(1/2); /\* 50% \*/ 7. } |

内置字符串函数

|  |
| --- |
| 1. .box:after { 2. content: e("ceil()"); 3. content: escape("a=1"); 4. content: replace("今天很热", "热", "冷"); 5. } 6. .box:after { 7. content: ceil(); 8. content: a%3D1; 9. content: "今天很冷"; 10. } |

注：&在less中表示当前所在的选择器

|  |
| --- |
| 1. .box { 2. width: 100px; 3. height: 100px; 4. &:hover { 5. color: white; 6. } 7. .box1 { 8. &:hover { 9. color: orange; 10. } 11. } 12. } |

&：等价于 .box

&: 等价于 .box .box1

### 2.3.8 定义色彩与色彩通道函数

定义色彩：

|  |
| --- |
| 1. /\*色彩的定义\*/ 2. .box1 { 3. background-color: pink; 4. } 5. .box2 { 6. background-color: #abc; 7. } 8. .box3 { 9. background-color: #abc7; 10. } 11. .box4 { 12. background-color: rgb(0, 0, 33); 13. } 14. .box5 { 15. background-color: rgba(88, 123, 255, .5); 16. } 17. .box6 { 18. background-color: hsl(180, .8, .6); 19. } 20. .box7 { 21. background-color: hsla(180, .8, .6, .6); 22. } |

色彩通道函数：

|  |
| --- |
| 1. /\*色彩的定义\*/ 2. @color1: #abc7; 3. @color2: #abc; 4. @color3: rgb(0, 0, 33); 5. @color4: rgba(88, 123, 255, .5); 6. /\*操作色彩\*/ 7. /\*获取某一个颜色的对应通道\*/ 8. @colorMix: rgb(red(@color1), green(@color3), blue(@color4)); 9. body { 10. background-color: @colorMix; 11. } |

### 2.3.9 色彩操作函数

可以调高某一个颜色的透明度、色相、饱和度、明度

fadeIn(color, 10%); 将color的透明度值调高百分之十 如果原来是50% 现在会变为60%

fadeOut(color, 10%); 将color的透明度值降低百分之十 如果原来是50% 现在会变为40%

fade(color, 50%); 将color的透明度设置为多少 不论原来的透明度是多少 经过设置之后就变为50%的透明度

demo：

|  |
| --- |
| 1. .box2 { 2. background-color: fadeIn(@color1, 10%); 3. background-color: fadeOut(@color1, 10%); 4. background-color: fade(@color1, 50%); 5. } |

结果:

|  |
| --- |
| 1. /\*将color1的透明度调整\*/ 2. .box2 { 3. background-color: rgba(170, 187, 204, 0.56666667); 4. background-color: rgba(170, 187, 204, 0.36666667); 5. background-color: rgba(170, 187, 204, 0.5); 6. } |

saturate(color, 10%); 将颜色的饱和度上升10%

desaturate(color, 10%); 将颜色的饱和度下降10%

|  |
| --- |
| 1. background-color: @color1; 2. background-color: saturate(@color1, 10%); 3. background-color: desaturate(@color1, 10%); |

结果:

|  |
| --- |
| 1. background-color: #abc7; 2. background-color: rgba(163, 187, 211, 0.46666667); 3. background-color: rgba(177, 187, 197, 0.46666667); |

lighten(color, 10%); 将颜色的亮度上升10%

darken(color, 10%); 将颜色的亮度下降10%

注：以上的所有方法都不会改变原颜色的值，而是返回新的颜色。

|  |
| --- |
| 1. /\*将color2的亮度调高\*/ 2. .box4 { 3. background-color: @color2; 4. background-color: lighten(@color2, 20%); 5. background-color: darken(@color2, 20%); 6. } |

结果：

|  |
| --- |
| 1. /\*将color2的亮度调高\*/ 2. .box4 { 3. background-color: #aba51f; 4. background-color: #dfd951; 5. background-color: #55520f; 6. } |

### 2.3.10 引入文件

语法： @import url(path);

less2.less：

|  |
| --- |
| 1. @import url("./less1.less"); 2. .box { 3. width: @key; 4. } |

less1.less

|  |
| --- |
| 1. @key: 123px; |

编译之后：

|  |
| --- |
| 1. .box { 2. width: 123px; 3. } |

### 2.3.11 插值语法

语法： @{variable}

demo:

|  |
| --- |
| 1. /\*定义变量\*/ 2. @key: left; 3. .@{key} { 4. width: 100px; 5. float: e("@{key}"); 6. border-@{key}: 1px solid #ccc; 7. } |

编译之后：

|  |
| --- |
| 1. /\*定义变量\*/ 2. .left { 3. width: 100px; 4. float: left; 5. border-left: 1px solid #ccc; 6. } |

# Sass

这也是一个CSS预编译语言。

由日本工程师开发的。基于Ruby。

所以，想要使用Sass，必须安装Ruby。

## 3.1 体验

index.scss

|  |
| --- |
| 1. div { 2. width: 100px; 3. height: 100px; 4. background-color: red; 5. p { 6. width: 50px; 7. height: 50px; 8. background-color: orange; 9. } 10. } |

与Less一样，也是嵌套式写法。

编译方式：1 安装ruby 2 安装sass模块 3 书写代码 4 编译

|  |
| --- |
|  |

编译结果：

|  |
| --- |
| 1. div { 2. width: 100px; 3. height: 100px; 4. background-color: red; 5. } 6. div p { 7. width: 50px; 8. height: 50px; 9. background-color: orange; 10. } 11. /\*# sourceMappingURL=index.css.map \*/ |

## 3.2 webpack编译

所需模块： style-loader css-loader sass-loader node-sass

|  |
| --- |
| 1. module.exports = { 2. entry: "./main.js", 3. output: { 4. path: \_\_dirname, 5. filename: "./bundle.js" 6. }, 7. module: { 8. rules: [ 9. { 10. test: /\.(scss|sass)$/, 11. loaders: "style-loader!css-loader!sass-loader" 12. } 13. ] 14. } 15. } |

## 3.3 语法

### 3.3.1 变量

语法规则： $key: value;

|  |
| --- |
| 1. /\*less中定义变量 @key: value;\*/ 2. /\*sass中定义变量 $key: value;\*/ 3. $w: 100px; 4. .box1 { 5. width: $w \* 2; 6. height: $w \* 2; 7. background-color: red; 8. .box2 { 9. width: $w; 10. height: $w; 11. background-color: blue; 12. } 13. } |

### 3.3.2 混合

与Less不同的是，Less中的混合可以直接使用

Sass中的混合分两步：定义、引用

定义：

|  |
| --- |
| 1. @mixin hunhe { 3. } |

引用:

|  |
| --- |
| 1. @include hunhe; |

demo:

|  |
| --- |
| 1. @mixin hunhe { 2. width: 100px; 3. height: 100px; 4. background-color: red; 5. } 6. div { 7. @include hunhe; 8. } |

### 3.3.3 方法

方法与混合的区别就是是否可以带参数

|  |
| --- |
| 1. @mixin hunhe($w:200px, $h:100px) { 2. width: $w; 3. height: $h; 4. background-color: red; 5. } 6. div { 7. @include hunhe(400px, 400px) 8. } 9. p { 10. @include hunhe(); 11. } |

方法的参数的默认值与less保持一致，都是用:定义的。

### 3.3.4 继承

@extend 被继承的类名

|  |
| --- |
| 1. .box1 { 2. width: 400px; 3. height: 400px; 4. background-color: red; 5. .box2 { 6. width: 300px; 7. height: 300px; 8. background-color: blue; 9. } 10. } 11. /\*unit1想要使用box1的样式\*/ 12. .unit1 { 13. @extend .box1; 14. } |

注：继承时，不仅仅会把被继承者自身的样式引入，还会将被继承者内部的子元素样式引入。

### 3.3.5 if语句

语法： @if statement {} @else if statement {} @else {}

注：表达式的两侧并没有圆括号

|  |
| --- |
| 1. /\* if语句 \*/ 2. @mixin box($w, $h) { 3. width: $w; 4. height: $h; 5. /\*判断 $w的宽度和$h的高度 从而决定color的颜色\*/ 6. @if $w > 100px { 7. background-color: red; 8. } @else if $w > 50px { 9. background-color: blue; 10. } @else { 11. background-color: purple; 12. } 13. } 14. .box { 15. @include box(200px, 200px); 16. } 17. .box1 { 18. @include box(75px, 100px); 19. } 20. .box2 { 21. @include box(25px, 25px); 22. } |

### 3.3.6 for语句

语法1： @for $variable from start to end start是一个数字 end也是数字 表示开始与结束 to不会执行最后一次

语法2： @for $variable from start through end through 会执行最后一次

demo1: to

|  |
| --- |
| 1. @for $i from 1 to 10 { 2. .box#{$i} { 3. width: 50px + $i \* 100; 4. height: 100px; 5. background-color: red; 6. } 7. } |

result1:

|  |
| --- |
| 1. .box1 { 2. width: 150px; 3. height: 100px; 4. background-color: red; } 5. .box2 { 6. width: 250px; 7. height: 100px; 8. background-color: red; } 9. .box3 { 10. width: 350px; 11. height: 100px; 12. background-color: red; } 13. .box4 { 14. width: 450px; 15. height: 100px; 16. background-color: red; } 17. .box5 { 18. width: 550px; 19. height: 100px; 20. background-color: red; } 21. .box6 { 22. width: 650px; 23. height: 100px; 24. background-color: red; } 25. .box7 { 26. width: 750px; 27. height: 100px; 28. background-color: red; } 29. .box8 { 30. width: 850px; 31. height: 100px; 32. background-color: red; } 33. .box9 { 34. width: 950px; 35. height: 100px; 36. background-color: red; } |

demo2:

|  |
| --- |
| 1. @for $i from 1 through 10 { 2. .box#{$i} { 3. width: 50px + $i \* 100; 4. height: 100px; 5. background-color: red; 6. } 7. } |

result2:

|  |
| --- |
| 1. .box1 { 2. width: 150px; 3. height: 100px; 4. background-color: red; } 5. .box2 { 6. width: 250px; 7. height: 100px; 8. background-color: red; } 9. .box3 { 10. width: 350px; 11. height: 100px; 12. background-color: red; } 13. .box4 { 14. width: 450px; 15. height: 100px; 16. background-color: red; } 17. .box5 { 18. width: 550px; 19. height: 100px; 20. background-color: red; } 21. .box6 { 22. width: 650px; 23. height: 100px; 24. background-color: red; } 25. .box7 { 26. width: 750px; 27. height: 100px; 28. background-color: red; } 29. .box8 { 30. width: 850px; 31. height: 100px; 32. background-color: red; } 33. .box9 { 34. width: 950px; 35. height: 100px; 36. background-color: red; } 37. .box10 { 38. width: 1050px; 39. height: 100px; 40. background-color: red; } |

### 3.3.7 while语句

语法： @while statement {}

demo:

|  |
| --- |
| 1. $a: 1; 2. @while $a < 13 { 3. .col-lg-#{$a} { 4. width: percentage($a/12); 5. } 6. $a: $a + 1; 7. } |

结果：

|  |
| --- |
| 1. .col-lg-1 { 2. width: 8.33333%; } 3. .col-lg-2 { 4. width: 16.66667%; } 5. .col-lg-3 { 6. width: 25%; } 7. .col-lg-4 { 8. width: 33.33333%; } 9. .col-lg-5 { 10. width: 41.66667%; } 11. .col-lg-6 { 12. width: 50%; } 13. .col-lg-7 { 14. width: 58.33333%; } 15. .col-lg-8 { 16. width: 66.66667%; } 17. .col-lg-9 { 18. width: 75%; } 19. .col-lg-10 { 20. width: 83.33333%; } 21. .col-lg-11 { 22. width: 91.66667%; } 23. .col-lg-12 { 24. width: 100%; } |

### 3.3.8 each语句

语法： @each $variable in enum

demo:

|  |
| --- |
| 1. $enum: "a", "b", "c", "d"; 2. @each $index in $enum { 3. .box-#{$index} { 4. width: 100%; 5. } 6. } |

result:

|  |
| --- |
| 1. .box-a { 2. width: 100%; } 3. .box-b { 4. width: 100%; } 5. .box-c { 6. width: 100%; } 7. .box-d { 8. width: 100%; } |

### 3.3.9 插值

插值语法： #{}

其它的插值语法：

<%%> 、 @{}

demo: 见3.3.7、 3.3.8

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |