Less

Less （Leaner Style Sheets 的缩写） 是一门向后兼容的 CSS 扩展语言

因为 Less 和 CSS 非常像，因此很容易学习。而且 Less 仅对 CSS 样式增加了少许方便的扩展，这就是 Less 如此易学的原因之一。

15.1.1 Less环境构建

在 Node.js 环境中使用 Less

```
npm install -g less
> lessc styles.less styles.css
```

在浏览器环境中使用 Less

在使用以下效果中，必须基于服务器运行

http-server

http-server 是一个简单的零配置命令行HTTP服务器, 基于 nodeJs

```
npm install http-server -g
```

运行,在项目的根目录下执行`http-server`

```
<link rel="stylesheet/less" type="text/css" href="./style.less" />
<script src="//cdn.jsdelivr.net/npm/less@3.13"></script>
```

考拉转换工具

koala是一个前端预处理器语言图形编译工具，支持Less、Sass、Compass、CoffeeScript，帮助web开发者更高效地使用它们进行开发。跨平台运行，完美兼容windows、linux、mac。

工程化转换

以下方案，我们会在接下来的课程中讲解到

1. less
2. webpack

开发工具中直接转换

1. vsCode（直接搜索Less插件，安装即可）

2. webStorm

15.1.2 Less语法

变量（Variables）

```
@width: 10px;
@height: @width + 10px;

#header {
  width: @width;
  height: @height;
}
```

编译为：

```
#header {
  width: 10px;
  height: 20px;
}
```

混合（Mixins）

混合（Mixin）是一种将一组属性从一个规则集包含（或混入）到另一个规则集的方法。假设我们定义了一个类（class）如下：

```
.bordered {
  border-top: dotted 1px black;
  border-bottom: solid 2px black;
}
```

如果我们希望在其它规则集中使用这些属性呢？没问题，我们只需像下面这样输入所需属性的类（class）名称即可，如下所示：

```
#menu a {
  color: #111;
  .bordered();
}

.post a {
  color: red;
  .bordered();
}
```

嵌套（Nesting）

Less 提供了使用嵌套（nesting）代替层叠或与层叠结合使用的能力。假设我们有以下 CSS 代码：

```
#header {
  color: black;
}
#header .navigation {
  font-size: 12px;
}
#header .logo {
  width: 300px;
}
```

用 Less 语言我们可以这样书写代码：

```
#header {
  color: black;
  .navigation {
    font-size: 12px;
  }
  .logo {
    width: 300px;
  }
}
```

用 Less 书写的代码更加简洁，并且模仿了 HTML 的组织结构。

你还可以使用此方法将伪选择器（pseudo-selectors）与混合（mixins）一同使用。下面是一个经典的 clearfix 技巧，重写为一个混合（mixin）（& 表示当前选择器的父级）：

```
.clearfix {
  display: block;
  zoom: 1;

  &:after {
    content: " ";
    display: block;
    font-size: 0;
    height: 0;
    clear: both;
    visibility: hidden;
  }
}
```

运算（Operations）

算术运算符 +、-、*、/ 可以对任何数字、颜色或变量进行运算。如果可能的话，算术运算符在加、减或比较之前会进行单位换算。计算的结果以最左侧操作数的单位类型为准。如果单位换算无效或失去意义，则忽略单位。无效的单位换算例如：px 到 cm 或 rad 到 % 的转换。

```
// 所有操作数被转换成相同的单位
@conversion-1: 5cm + 10mm; // 结果是 6cm
@conversion-2: 2 - 3cm - 5mm; // 结果是 -1.5cm

// conversion is impossible
@incompatible-units: 2 + 5px - 3cm; // 结果是 4px

// example with variables
@base: 5%;
@filler: @base * 2; // 结果是 10%
@other: @base + @filler; // 结果是 15%
```

乘法和除法不作转换。因为这两种运算在大多数情况下都没有意义，一个长度乘以一个长度就得到一个区域，而 CSS 是不支持指定区域的。Less 将按数字的原样进行操作，并将为计算结果指定明确的单位类型。

```
@base: 2cm * 3mm; // 结果是 6cm
```

你还可以对颜色进行算术运算：

```
@color: #224488 / 2; //结果是 #112244
background-color: #112244 + #111; // 结果是 #223355
```

转义（Escaping）

转义（Escaping）允许你使用任意字符串作为属性或变量值。任何 ~"anything" 或 ~'anything' 形式的内容都将按原样输出，除非 interpolation。

```
@min768: ~"(min-width: 768px)";
.element {
  @media @min768 {
    font-size: 1.2rem;
  }
}
```

编译为：

```
@media (min-width: 768px) {
  .element {
    font-size: 1.2rem;
```

```
    }
}
```

函数（Functions）

Less 内置了多种函数用于转换颜色、处理字符串、算术运算等。函数手册

函数的用法非常简单。下面这个例子将介绍如何利用 percentage 函数将 0.5 转换为 50%，将颜色饱和度增加 5%，以及颜色亮度降低 25% 并且色相值增加 8 等用法：

```
@width: 0.5;

.class {
  width: percentage(@width); // returns `50%`
}
```

映射（Maps）

从 Less 3.5 版本开始，你还可以将混合（mixins）和规则集（rulesets）作为一组值的映射（map）使用。

```
#colors() {
  primary: blue;
  secondary: green;
}

.button {
  color: #colors[primary];
  border: 1px solid #colors[secondary];
}
```

输出符合预期：

```
.button {
  color: blue;
  border: 1px solid green;
}
```

作用域（Scope）

Less 中的作用域与 CSS 中的作用域非常类似。首先在本地查找变量和混合（mixins），如果找不到，则从“父”级作用域继承。

```
@var: red;
```

```
#page {
  @var: white;
  #header {
    color: @var; // white
  }
}
```

与 CSS 自定义属性一样，混合（mixin）和变量的定义不必在引用之前事先定义。因此，下面的 Less 代码示例和上面的代码示例是相同的：

```
@var: red;

#page {
  #header {
    color: @var; // white
  }
  @var: white;
}
```

注释（Comments）

块注释和行注释都可以使用：

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

// 这一行被注释掉了!
@var: white;
```

导入（Importing）

“导入”的工作方式和你预期的一样。你可以导入一个 .less 文件，此文件中的所有变量就可以全部使用了。如果导入的文件是 .less 扩展名，则可以将扩展名省略掉：

```
@import "library"; // library.less
@import "typo.css";
```