你知道什么是CSS预编译器么?

为什么会出现CSS预编译器这个东西呢？这就要谈到CSS的不足了:没有变量(新的规范已经支持了)，不支持嵌套，编程能力较弱，代码复用性差。这些不足导致写出来的CSS维护性极差，同时包含大量重复性的代码；为了弥补这些不足之处,CSS预编译器应运而生。而谈到CSS预编译器,就离不开这三剑客Sass、Less、Stylus。历史上，最先登场的是Sass，因为出现最早，所以也是最完善的，有各种丰富的功能；Less的出现伴随着Bootstrap的流行，因此也获得大量用户；最后是Stylus，由TJ大神开发(敬大神)，由于其简洁的语法，更像是一门编程语言，写起来非常Cool。所以下面我们来做一个简单的对比。

```
Less & SCSS:
.wrap {
  display: block;
}

Sass:
.wrap
  display: block

Stylus:
.wrap
  display block
```

Sass最开始通过缩进，空格，换行的形式来控制层级关系，写过Python的同学一定不会陌生，后来又支持了传统的类CSS语法的Scss。Less中规中矩，使用CSS的风格，对新手非常友好，也利于现有项目的迁移。Stylus既可以使用Sass风格的语法来编写，也兼容CSS的风格。

这三者都支持变量的定义，而定义方式又各不相同:

```
Less:
@smallFont: 12px;
small {
  font-size: @smallFont;
}

Sass:
$smallFont: 12px;
small {
  font-size: $smallFont;
}

Stylus:
```

```
smallFont = 12px
small
  font-size smallFont
```

需要注意的是：Stylus 中声明的变量，如果变量名跟 CSS 中的字面值相同时，会覆盖 CSS 中的字面值。

有一个好玩的东西：相信大家平时都或多或少会接触到组件库，组件库的编写中会使用到 CSS 的预编译器来编写样式，通过外部传入变量的方式来提供自定义样式的功能，就像 Ant.design 和 ElementUI 那样。这就涉及到 CSS 变量的作用范围了，在这三种中预编译中，Sass/Stylus 是相同的，Less 是另一种情况，下面看实例：

Less:

```
@color: red;
.content-1 {
    color: @color;
}

@color: black;
.content-2 {
    color: @color;
}

/* 编译出来的 CSS*/
.content-1 {
  color: black;
}

.content-2 {
  color: black;
}
```

Less 中的变量，在声明中使用时，如果出现多次赋值的情况，其会取最后一次赋值的值，这也就是上面的.content-1， content-2 中的 color 都是 black 的原因。基于 Less 中变量的这个特性，我们可以很轻易的改变原有组件库或者类库中变量的值，只需要在引入 Less 文件后，对特定的变量赋值即可。同时也会带来一定的隐患：如果不同的组件库或类库使用了相同的变量名，那么就会出现覆盖的情况，所以应该采用模块化的方式。

Sass:

```
$color: red;
.content-1 {
    color: $color;
}
```

```
$color: black;
.content-2 {
    color: $color;
}

Stylus:
color = red;
.content-1
    color color

color = black;
.content-2
    color color;


/* 编译出来的CSS*/
.content-1 {
  color: red;
}

.content-2 {
  color: black;
}
```

上面我们可以看到，Sass/Stylus 中的变量，如果出现多次赋值的情况，其会取声明前面最近的一次赋值的值，这就是为什么.content-1 的 color 为 red，.content-2 的 color 为 black 的原因。同时，在 Sass/Stylus 编写的不同组件库或类库中的变量，不会出现冲突，但是这就为通过覆盖变量的值来自定义样式提出了挑战，我们应该怎么做呢？考点来了，Sass/Stylus 中提供了"不存在即赋值"的变量声明：

```
Sass:
$a: 1;
$a: 5 !default;
$b: 3 !default;
// $a = 1, $b = 3

Stylus:
a = 1
a := 5
b = 3
// a = 1, b = 3
```

细看代码，你一定会明白：使用"不存在即赋值"语法，编译器会检查前面是否定义了同名变量，如果没有定义，执行变量定义；如果前面定义了同名变量，则不会执行这个赋值的操作。因此，我们可以在使用了 Sass/Stylus 的组件库或类库

中使用"不存在即赋值"的方式来定义变量，在引入之前定义好需要同名变量，就能达到目的了。

同时，我们定义的变量不仅能用作 CSS 声明的值，同时也能嵌套到选择器及属性中：

```
Less:
@prefix: ui;
@prop:  background;
.@{prefix}-button {
      @{prop}-color: red;
}

Sass:
$prefix: ui
$prop:  background;
.#{$prefix}-button{
     #{$prop}-color: red;
}

Stylus:
prefix = ui
prop = background
.{prefix}-button
     {prop}-color red
```


变量相关的内容基本上覆盖到了，接下来就是样式的复用了。提到样式复用，必定会提到 mixin 和继承了。对于 mixin，这三个预编译器也都有不同的实现方式：

```
Less:

.mixin-style(@color: red) {
  color: @color;
}

.content {
  .mixin-style(red);
}

Sass:
@mixin mixin-style {
    color: red;
}

.content {
```

```
    @include mixin-style;
}

Stylus:
  mixin-style(color)
    color color

  .content
    mixin-style(red)   // or mixin-styls red  透明mixin
```

在Less中，可以直接引入一个CSS的class作为mixin(这种方式非常不推荐)，同时也提供上面的能够传入参数的mixin；Sass比较中规中矩，通过@mixin和@include的方式定义和引入mixin；Stylus不需要显示的声明mixin，同时还提供透明mixin的功能，就像属性一样引入。

接下来就会讲到继承了，这其中，Sass/Stylus通过@extend关键字来继承样式，而Less通过伪类的方式来继承：

Sass/Stylus:

```
.message{
    padding: 10px;
    border: 1px solid #eee;
}

.warning{
    @extend .message;
    color: #e2e21e;
}
```

Less:

```
.message {
  padding: 10px;
  border: 1px solid #eee;
}

.warning {
  &:extend(.message);
  color: #e2e21e;
}
```

个中优劣，大家可以自己评判。具体的细节在这里就不过多展开了。