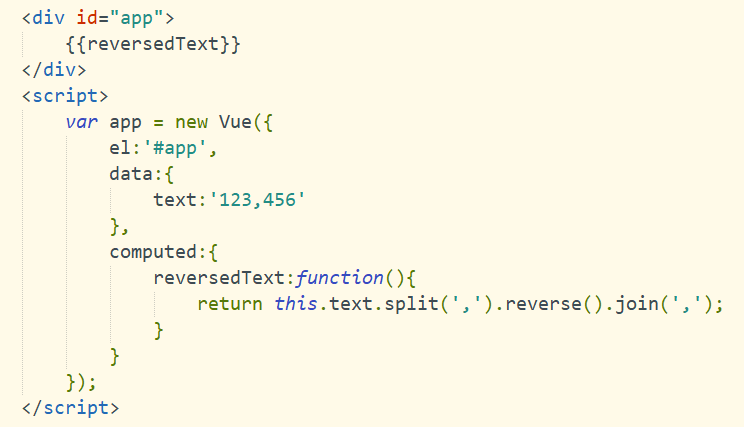
# 什么是计算属性

模板内的表达式常用于简单的运算，如果表达式过长或逻辑复杂时，会变得臃肿、难以阅读、难以维护，比如：



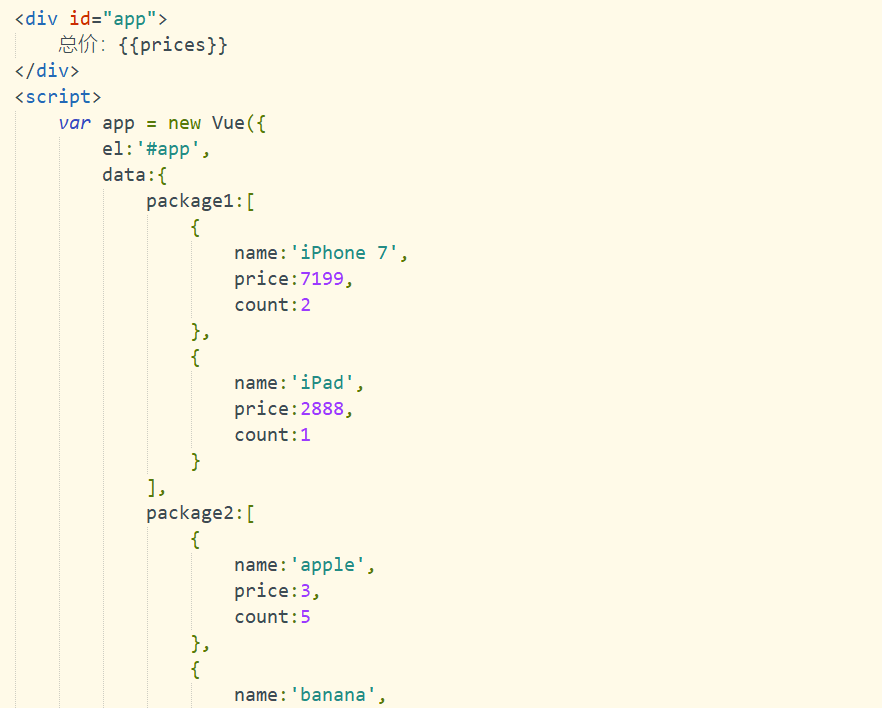
这里的插值表达式包含了3个操作，并不是很清晰，所以在遇到复杂的逻辑时应该使用计算属性。对以上表达式使用计算属性进行改写：

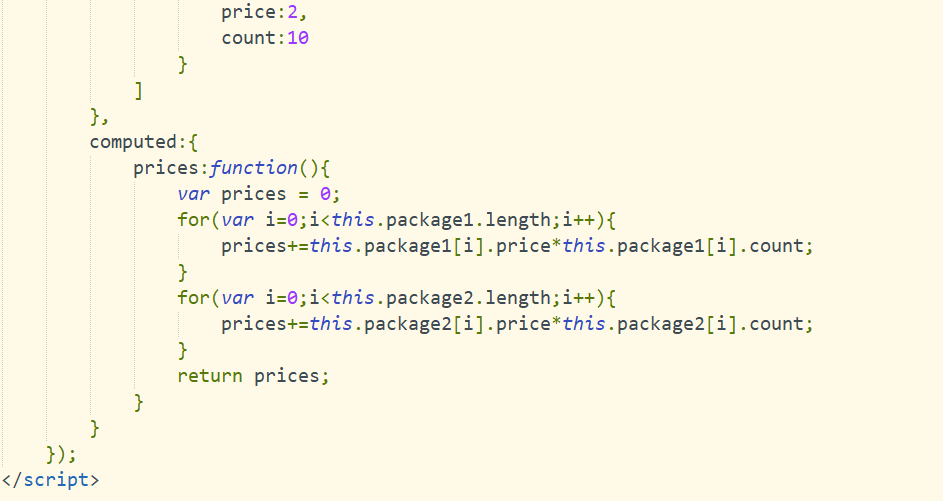


# 计算属性用法

在一个计算属性里可以完成各种复杂的逻辑，包括运算、函数调用等，只要返回一个结果就可以。

1. 除了上例简单的用法，计算属性还可以依赖Vue实例中的多个数据，只要其中任一数据变化，计算属性就会重新执行，视图也会更新。





当package1或package2中的商品数量或商品价格发送变化后，计算属性prices也会发生变化，并实时更新到视图中。

1. 每一个计算属性都包含一个getter和一个setter，我们上面的两个示例都是计算属性的默认用法，只是利用了getter来读取。在你需要时，也可以提供一个setter函数，当手动修改计算属性的值就像修改一个普通数据那样时，就会触发setter函数，执行一些自定义的操作。如：



当执行 app.fullName=’John Doe’ 时，setter就会被调用，firstName和lastName都会相对更新，视图同样也会更新。

但在绝大数情况下，我们只会用默认的getter方法来读取一个计算属性，在业务中很少用到setter，所以在声明一个计算属性时，可以直接使用默认的写法，不必将getter和getter都声明。

1. 计算属性不仅可以依赖当前Vue实例中的数据，还可以依赖其他Vue实例中的数据。如：

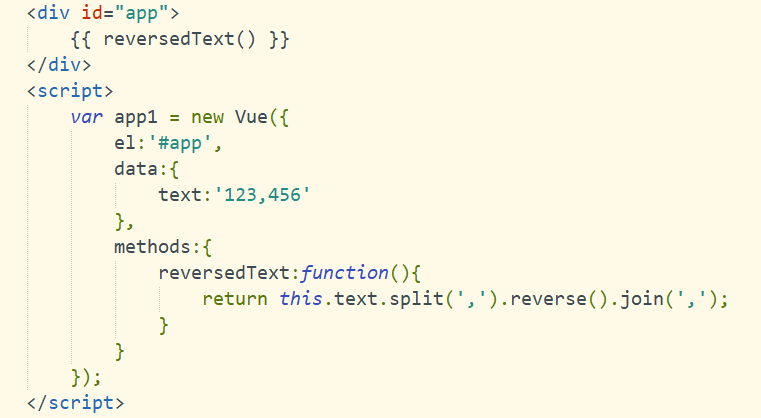


上面代码创建了两个Vue实例app1和app2，在app2的计算属性reversedText中，依赖了实例app1中的数据text，所以当text变化时，实例app2的计算属性也会发生变化。

这样的用法在后面章节介绍的组件和组件哈里会用到，尤其是多人协同开发时很常用，因为你写的一个组件所用得到的数据需要依赖他人的组件提供。

# 计算属性缓存

在前面学习v-on指令时，可以将Vue实例的methods参数中定义的方法绑定给某元素，但methods参数中定义的方法也可以与计算属性起到同样的作用。如：



在插值表达式中，放置的是reversedText( )方法，这个方法是在Vue实例methods参数中定义的，执行该方法，将返回值渲染到视图中；这里没有使用计算属性，却实现了计算属性的功能，该方法还可以接收参数，使用起来更灵活；那既然这样，为什么还需要计算属性呢？原因是因为计算属性是基于依赖缓存的，当一个计算属性所依赖的数据发生变化时，它也会发生变化，并从重新渲染到视图中。