# 自定义指令

在前面的学习中已经介绍过许多Vue内置指令，比如v-if、v-for、v-show等，这些丰富的内置指令能满足绝大部分业务需求，不过在需要一些特殊功能时，希望对DOM进行底层的操作，这时就需要用到自定义指令。

## 注册指令

自定义指令的注册方法和组件很像，也分为全局注册和局部注册，全局注册可以在已被挂载的DOM中使用，局部注册只能在当前Vue实例的挂载范围内使用。比如注册一个v-focos的指令，用于在<input>、<textarea>元素初始化时自动获得焦点，两种写法分别如下：



上例中只是注册了自定义指令v-focus，还没有实现具体实现。

## 定义指令的功能

注册指令后，通过指令的各个选项来定义该指令要实现的功能。指令的选项是由几个钩子函数组成的，每个都是可选的。

·bind：指定第一次绑定到元素时调用，只调用一次

·inserted：被绑定元素插入父节点时调用（父节点存在即可，不必存在documen

t中）

·update：被绑定元素所在的模板更新时调用，而不论绑定值是否变化。通过比较

更新前后绑定值，可以忽略不必要的模板更新。

·componentUpdated：被绑定元素所在模板完成一次更新周期时调用。

·unbind：指令与元素解绑时调用，只调用一次。

可以根据需求在不同的钩子函数内完成逻辑代码，比如注册一个v-focos的指令，用于在<input>、<textarea>元素初始化时自动获得焦点。我们希望在元素插入父节点时就调用，那用到的做好是inserted。示例如下：



当运行该页面，文本框将自动获取焦点。其中inserted函数的第一个参数表示该指令绑定的元素。

## 可用参数

每个钩子函数都有几个参数可用，比如上节示例中的el。它们的含义如下：

·el：指令所绑定的元素，可以用来直接操作DOM

·binding：一个对象，包含以下属性：

name：指令名称，不包括v-前缀

value：指令绑定的值，例如v-my-directive=”1+1”，value的值为2

oldValue：当指令绑定的值改变后，保存指令上一个旧的值，仅在update和

componentUpdated钩子函数中可用（无论value是否改变都可用）。

expression：指定绑定的值的字符串表现形式。例如v-my-directive=”1+1”，

expression的指是”1+1”

arg：传给指令的参数。例如v-my-directve:foo，argu的值是foo

modifiers：一个包含指令修饰符的对象。例如v-my-directive.foo.bar，修饰符

对象modifiers的值是{foo:true,bar:true}

·vnode：Vue编译生成的虚拟节点（进阶篇中介绍）

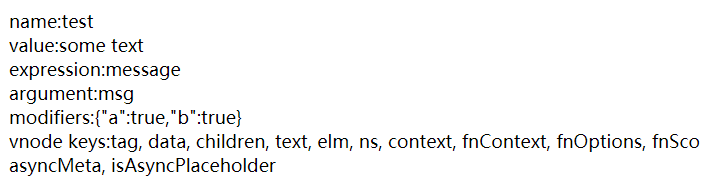
·oldValue：当虚拟节点改变后，保存上一个旧的虚拟节点，仅在update和

componentUpdated钩子函数中可用

通过一个示例来验证以上参数的使用：



运行后，<div>的内容会还有innerHTML重置，结果为：



在大多数使用场景，我们会在bind函数里绑定一些事件，比如document上用addEventListener绑定，在unbind里用removeEventListener解绑，比较典型的示例就是让这个元素随着鼠标拖拽。

如果指令需要多个值，自定义指令也可以传入一个JavaScript字面量对象，只要是合法类型的JavaScript都是可以的。示例如下：

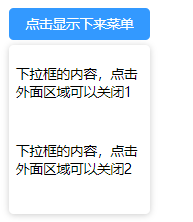


在使用自定义指令时，应该充分理解业务需求，因为很多时候你需要的可能并不是自定义指令，而是组件。

# 实战

## 开发一个可从外部关闭的下拉菜单

网页中有很多下拉菜单，如下：

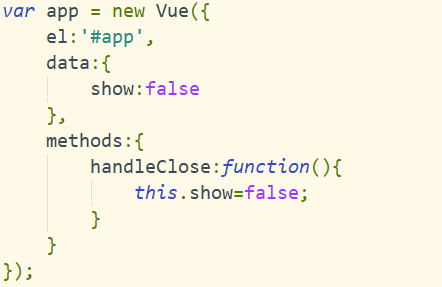


该菜单有两个特点，一是点击下拉菜单本身是不会关闭的，而是点击下来菜单以外的所以区域都会关闭此下来菜单。点击所有区域可以在document上绑定click事件来实现，同时只要过滤出是否点击的是目标元素内部的元素即可。下面通过自定义指令来实现这样的功能。

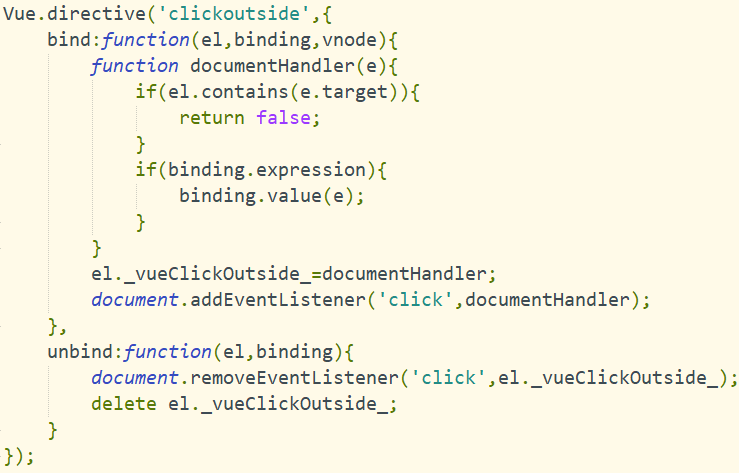
**index.html**



**index.js**



**clickoutside.js**



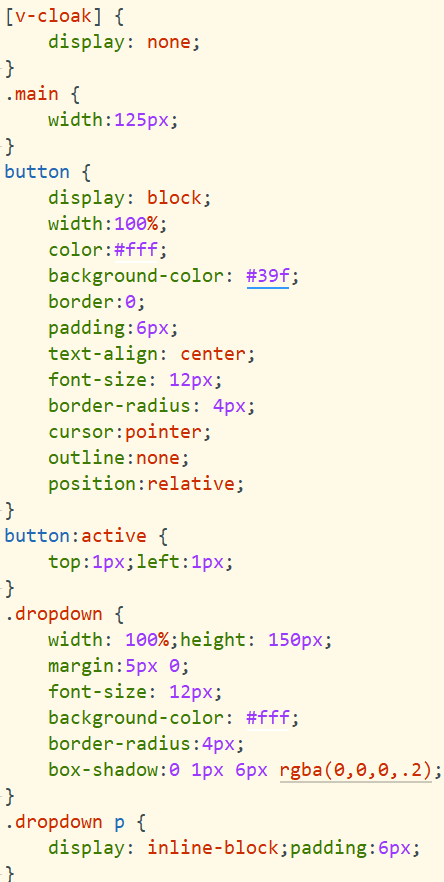
之前分析过，要在document上绑定click事件，所以在bind钩子内声明了一个函数documentHandler，并将它作为句柄绑定在document的click事件上documentHandler函数做了两个判断，第一个判断点击的区域是否是指令所在的元素内部，如果是，就跳出函数，不往下继续执行。contains方法是用来判断A是否包含了B元素，包含返回true，不包含返回false。使用示例如下：



第二个判断的是当前的指令v-clickoutside有没有写表达式，在该自定义指令中，表达式应该是一个函数，在过滤了内部元素后，点击外面任何区域应该执行用户表达式中的函数，所有binding.value()就是用来执行当前上下文methods中指定的函数 的。由于在定义指令中，不能使用this.xxx的形式在上下文声明一个变量，所以用了el.\_vueClickOutside\_引用了documentHandler，这样就可以在unbind钩子里移除对document的click事件监听了。如果不移除，当组件或元素销毁时，它仍然存在于内存中。

**2.css**

美化样式文件内容如下：



## 开发一个实时时间转换指令

在一些社区，比如微博，朋友圈等，发布的动态会有一个相对本机时间转换后的相对时间。如下：



一般在服务端的存储时间格式是Unix时间戳，比如2017-01-01 00:00:00的时间戳是1483200000，前端拿到后端存储的时间后，将它转换为可读的时间格式再显示出来。为了突出实时性，在一些社交类产品中，甚至会实时转换为几秒钟前、几分钟前、几小时前等不同的格式，这样比直接转换为年、月、日、时、分、秒更友好。本示例就来实现这样一个自定义指令v-time，将表达式传入的时间戳实时转换为相对时间。经过分析后，时间转换的逻辑如下：

·1分钟以前，显示“刚刚”

·1分钟~1小时之间，显示“xx分钟前”

·1小时~1天之间，显示“xx小时前”

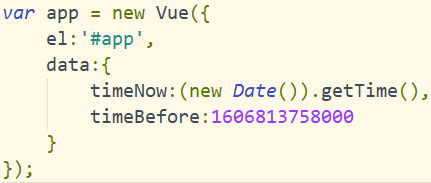
·1天~1个月（31天）之间，显示“xx天前”

·大于1个月，显示“xx年xx月xx日”

**index.html**



**index1.js**

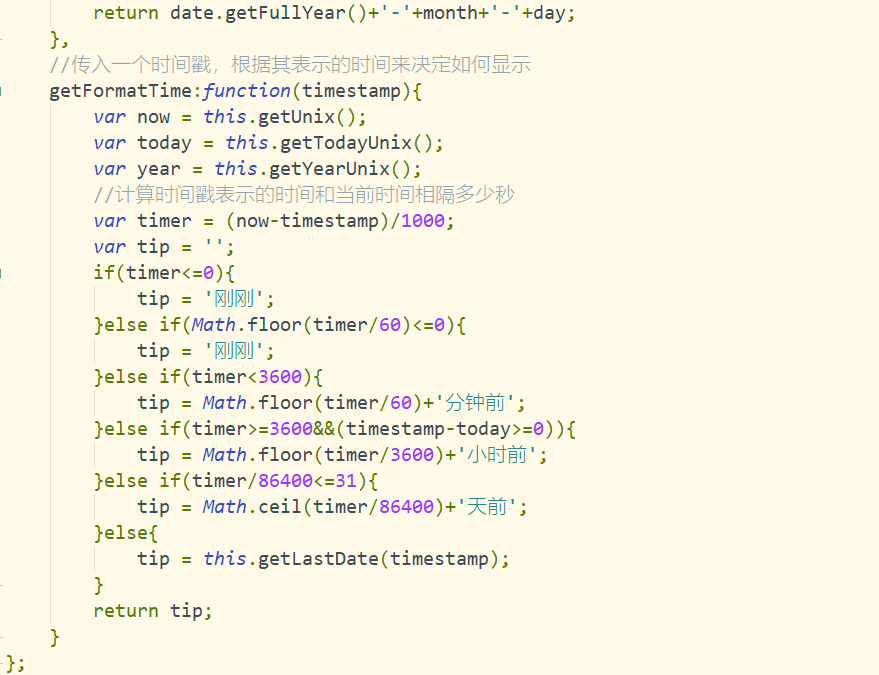


为了演示效果，初始化时定义了两个时间。timeNow是当前时间，timeBefore是一个较早的时间。

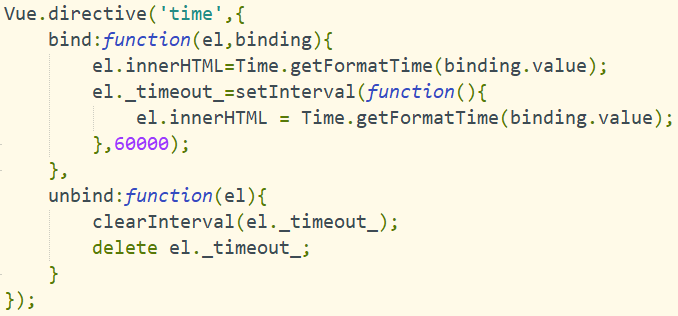
**time.js**

首先声明一个对象Time，将需要常用的方法封装在该对象中。





关于Date对象和Math对象不太了解，可以回顾JavaScript。其中getFormatTime( )方法就是自定义指令v-time所需要的，入参为毫秒级类型的时间戳，返回最终要显示的字符串。最后在time.js里补全剩余的代码：



在bind函数里，将指令v-time表达式的值binding.value（一个毫秒级类型的时间戳）作为参数传入Time.getFormatTime( )函数中得到格式化时间，再通过el.innerHTML写入指令所绑定的元素内。定时器el.\_timeout\_每分钟触发一次，更新时间，并且在unbind钩子里清除掉。

总结：在编写自定义指令时，给DOM绑定一次性事件等初始动作，建议在bind钩子内完成，同时要在unbind内解除相关绑定。在定义指令里，理论上可以任意操作DOM，但这有违背了Vue.js的初衷，所以对于大幅度的DOM变动，应该使用组件。