## 第3章 Vue事件处理

作为响应式前端，事件主要分为：与人交互事件，请求数据的异步事件，以及模板加载的生命周期事件等。如果想处理事件，就得去监听事件。

### 3.1 事件监听

事件监听可以用 v-on 指令监听 DOM 事件，并在触发时运行一些 JavaScript 代码，示例：

<div id**="example-1"**>  
 <button v-on:click**="counter += 1"**>**Add 1**</button>  
 <p>**The button above has been clicked {{ counter }} times.**</p>  
</div>   
<script>  
 var *example1* = new Vue({  
 el: '#example-1',  
 data: {  
 counter: 0  
 }  
 })  
</script>

**v-on:click**，监听的是click事件，即鼠标点击事件，鼠标每点击一次，{{ counter }}增加1。

### 3.2 事件处理方法

对于复杂的逻辑，需要写一个方法函数，赋值给v-on:click，避免模板过于复杂难以维护。如下例所示，把事件处理方法写在methods里：

模板：

<div id**="app"**>  
 <button v-on:click**="greet"**>**Greet**</button>  
</div>

JS:

var *example2* = new Vue({  
 el: '#app',  
 data: {  
 name: 'Vue.js'  
 },  
 methods: {  
 greet: function (event) {  
 *alert*('Hello ' + this.name + '!')  
 // `event` 是原生 DOM 事件  
 if (event) {  
 *alert*(event.target.tagName)  
 }  
 }  
 }  
})

greet是方法函数，必须写在methods对象里。greet方法有一个参数event，在调用时并没有传递，依然可以访问到event，这个是默认的DOM事件，可以不传递。