# 事件处理

Montage在浏览器原生事件处理上包装实现了自身的事件处理模块，但是包装对用户如何使用事件是透明的。Montage事件处理提供几个特性，包括让事件处理代码更简单、属性改变监听器和高性能。

## 事件委托

Montage使用事件委托来管理事件处理和分发。使用事件委托不需要在每一个元素上面定义监听，只需要在父元素上绑定一个监听， 然后所有的子元素都使用它。 这样的事件处理方式依赖于[DOM Level 3 Event Specification](http://www.w3.org/TR/DOM-Level-3-Events/#event-flow)定义的标准事件“流"。

事件委托方式有很多好处。比如，因为减少事件监听器的定义，所以性能更高。在Montage应用中只有一个“原生”事件监听器，它负责所有其它事件的接收和分发，可以把它看做是一个主事件监听器。委托也可以让Montage应用可以监听到对象属性值和数组改变事件。

## 创建事件处理

使用标准的addEventListener()函数在目标对象上面注册一个事件处理，在Montage事件处理中，目标对象可以是任意的JavaScript对象，而不是像JavaScript事件处理只可以是DOM元素。

target.addEventListener(eventType, listener[, useCapture]);

* eventType 事件名字的字符串。
* listener 事件处理对象（实现Montage事件监听器接口），或者直接就是一个函数
* useCapture是一个布尔参数；如果值为true，监听器对象的处理函数会被第一个调用，然后监听器对象子组件对应的监听函数被调用。useCapture默认值false,也就是冒泡方式(**bubble**)传递事件, 如果是true就是捕获方式(**capture**)。

### Montage事件监听器接口

Montage事件监听接口实现[DOM Level 3 EventListener interface](http://dev.w3.org/2006/webapi/DOM-Level-3-Events/html/DOM3-Events.html#interface-EventListener)标准定义，这个定义被当前全部浏览器支持。第一步在目标对象上设置一个事件监听器对象，当相应的事件被触发后，浏览器会调用监听器对象的handleEvent()方法。

// DOM Level 3 EventListener interface  
var listenerObj = {};  
listenerObj.handleEvent = function(event) {  
 alert("Got 'mousedown' event.");  
}  
var loginBtn = document.querySelector("#loginBtn");  
loginBtn.addEventListener("mousedown", listenerObj);

Montage增强了这个接口让开发者更容易使用。在Montage中，不是调用监听器对象的handleEvent()方法，而是调用事件对象中与当前事件相应的方法。在事件处理方法的传入参数中包括以下三个数据：

* 事件阶段（冒泡或者捕获）
* 事件名字
* 缺省的，字符串类型标识符，标识目标DOM元素或者Javascript对象。

下面的伪代码展示事件模块是如何确定调用监听器对象的什么方法：

methodToInvoke = "";  
identifier = eventTarget.identifier;  
if (event.phase == "bubble" ) {  
 methodToInvoke = "handle" +   
 (identifier ? identifier.toCapitalized() : "") +  
 eventType.toCapitalized();  
} else {  
 methodtoInvoke = "capture" +   
 (identifier ? identifier.toCapitalized() : "") +   
 eventType.toCapitalized();  
}

理解Montage事件处理最好的方法是通过例子来了解。

## 例子

下面的代码跟上面的例子一样，只是事件处理方法名是handleMousedown()而不是handleEvent()（Montage事件处理）。当loginBtn触发mousedown事件之后这个方法会被自动调用，注意只是在事件冒泡阶段。

// Listening for mousedown event during bubble phase  
var listenerObj = {};  
listenerObj.handleMousedown = function(event) {  
 alert("Got 'mousedown' event.");  
}  
var loginBtn = document.querySelector("#loginBtn");  
loginBtn.addEventListener("mousedown", listenerObj);

如果想监听事件的捕获阶段，你需要设置addEventListener()第三个参数为true，然后修改处理函数handleMousedown()名为captureMousedown()。

// Listening for capture events on same element   
var listenerObj = {};  
listenerObj.captureMousedown = function(event) {  
 alert("Got 'mousedown' event during bubble phase.");  
}  
var loginBtn = document.querySelector("#loginBtn");  
loginBtn.addEventListener("mousedown", listenerObj, true); // useCapture = true

你还可以指定事件目标对象的identifier属性，这样只有相应的事件处理函数会被调用。事件处理模块会调用identifier首字面大写的函数。 在下面的例子中，loginBtn的identifier值为\_\_login\_\_，所以对应的事件处理函数名为handleLoginMousedown()。

// Using identifier strings on target elements  
var listenerObj = {};  
// Listener for loginBtn  
listenerObj.handleLoginMousedown = function(event) {  
 console.log("mousedown on loginBtn");  
}  
var loginBtn = document.querySelector("#loginBtn");  
// Assign string identifier to button  
loginBtn.identifier = "login";  
loginBtn.addEventListener("mousedown", listenerObj);

### 事件处理优先级

事件处理模块会调用更具体的事件处理。比如下面的例子中，定义了两个事件处理函数，一个包含目标的标识符(handleLoginMousedown())， 而另外一个不包含(handleMousedown())。 Montage会调用handleLoginMousedown()，因为它比另外一个更具体。

// Event handler precedence  
var listenerObj = {};  
listenerObj.handleMousedown = function(event) {  
 // This won't get called.  
 alert("Got 'mousedown' event.");  
}  
listenerObj.handleLoginMousedown = function (event) {  
 alert("Got 'mousedown' event on event.target");   
}  
var loginBtn = document.querySelector("#loginBtn");  
loginBtn.identifier = "login";  
loginBtn.addEventListener("mousedown", listenerObj);

注意如果loginBtn没有定义identifier属性，handleMousedown()会被调用。

当然如果监听器对象里面没有定义具体的事件处理函数，Montage会调用handleEvent()方法。这样就提供了一种对“普通”事件的处理机制。

// Using default handleEvent() handler  
var listenerObj = {};  
listenerObj.captureClickEvent = function(event) {  
 alert("Got click event");  
}  
listenerObj.handleEvent = function(event) {  
 alert("No specific handler for " + event.type);  
}  
loginBtn.addEventListener("mousedown", listenerObj);  
loginBtn.addEventListener("click", listenerObj, true);

## 在组件模板中定义事件监听器

在模板中每个对象可以包含一个"listeners"数组指定监听的事件名字和监听器对象，事件是否是捕获方式（可选）。

首先在组件JS文件中新建handleAction()方法，这个方法的功能是改变触发该事件按钮的标签。

// controller.js  
var Component = require("montage/ui/component").Component;  
  
exports.Controller = Component.specialize({  
 handleAction: {  
 value: function(event) {  
 event.target.value = "Well done";  
 }  
 }  
})

接下来在组件模板中定义按钮组件。按钮组件的listeners属性包含事件名字(action)和用来处理事件的监听器对象。

<html>  
 ...  
<script type="text/montage-serialization">  
{  
 "button" : {  
 "name": "Button",  
 "module": "montage/ui/button.reel",  
 "properties": {  
 "element": {"#": "btn"}  
 },  
 "listeners": [  
 {  
 "type": "action",  
 "listener": {"@": "owner"}  
 }  
 ]  
 }  
}  
</script>  
 ...  
</html>

你也可以像下面这样指定identifier属性：

{  
 "button" : {  
 "name": "Button",  
 "module": "montage/ui/button.reel",  
 "properties": {  
 "element": {"#": "btn"},  
 "identifier": "purchase"  
 },  
 "listeners": [  
 {  
 "type": "action",  
 "listener": {"@": "owner"}  
 }  
 ]  
 }  
}