

开发了一个电子书分享小网站，目前分享了 1000 多本技术类电子书，欢迎围观：GeekGist.com。



精致码农 · 王亮

Be humble, communicate clearly, and respect others.

[博客园](#)[首页](#)[新随笔](#)[联系](#)[订阅](#)[管理](#)

随笔 - 96 评论 - 2058 阅读 - 113万

ExtJS框架基础：事件模型及其常用功能

前言

工作中用ExtJS有一段时间了，Ext丰富的UI组件大大的提高了开发B/S应用的效率。虽然近期工作中天天都用到ExtJS，但很少对ExtJS框架原理性的东西进行过深入学习，这两天花了些时间学习了下。我并不推荐大家去研究ExtJS框架的源码，虽然可以学习其中的思想和原理，但太浪费精力了，除非你要自己写框架。

对于ExtJS这种框架，非遇到“杂症”的时候我觉得也没必要去研究其源码和底层的原理，对一些机制大致有个概念，懂得怎么用就行，这也是本篇博文的主要目的。

Ext自己的事件机制

Ext中的事件遵循树状模型，和事件相关的类主要有这么几个：Ext.util.Observable、Ext.lib.Event、Ext.EventManager和Ext.EventObject。

Ext使用Ext.lib.Event、Ext.EventManager和Ext.EventObject对原生浏览器事件进行了封装，最后给我们用的是一套统一的跨浏览器的通用事件接口。HTML元素本身已经支持事件，为什么基本上所有的主流JS框架都要实现自己的事件机制呢？一个最主要的原因是HTML元素对事件的处理是通过简单的单一绑定实现的，如果不进行封装，事件只能绑定到一个事件处理句柄。如下面代码所示：

```
var e = document.getElementById("test");
e.onclick = function() { alert("handler1") };
e.onclick = function() { alert("handler2") };
```

单击test按钮后会发现只会弹出一个显示“handler2”的提示框，因为第一个被覆盖。而使用像Ext、jQuery这样的框架就不用担心这个问题，同一个事件可以依次绑定多个事件处理句柄，如下代码所示：

```
Ext.onReady(function () {
    var test = Ext.get("test");
    test.on("click", function () {
```

专注技术 · 分享干货 · 精进思想

微信扫码关注『精致码农』公众号



关注有惊喜

申请加入 [全球.NET技术交流微信群](#)，
加 Willick 好友，备注：.NET

.NET Core 技术交流QQ群：

239152943



微信号：[Willick](#)（请备注博客园）

头条号：[精致码农](#)（不仅仅是编程）

联系我：[给我写信](#)

所在地：上海 闵行区

昵称：精致码农

园龄：10年7个月

粉丝：2582

关注：13

+加关注

最新随笔

- 1..NET 6 Preview 3 中 ASP.NET Core 的更新和改进
- 2.[C#.NET 拾遗补漏]14：使用结构体实现共用体
- 3.深入解析 C# 的 String.Create 方法
- 4.Visual Studio 调试技巧之即时断点
- 5.审计系统的一剂良方——事件溯源

8

```
        alert("handler1");
    });
    test.on("click", function () {
        alert("handler2");
    });
});
```



Ext实现自己的事件机制，原因很多，比如为了兼容不同浏览器之间的差异等。Ext对原生浏览器事件的封装都在上面所说的几个类中，如果在项目中要熟练应用Ext，是非常有必要了解一下和事件相关的类和常用函数的。下面开始介绍这些类和它们的功能。

Ext.util.Observable

Ext.util.Observable在Ext事件模型中有着举足轻重的地位，位于Ext组件的顶端，为Ext组件提供处理事件的最基本的功能。所有继承自Ext.util.Observable类的控件都可以支持事件。可以为这些继承了Ext.util.Observable的对象定义一些事件，然后为这此事件配置监听器。当某个事件触发时，Ext会自动调用对应的监听器，这些就是Ext的事件模型。

下面通过继承Ext.util.Observable来实现一个支持事件的对象：



```
Ext.onReady(function () {
    //定义一个Person类。
    function Person(name) {
        this.name = name;
        this.addEvents("walk", "eat");
        this.superclass.constructor.call(this);
    }

    //1、让Person继承Ext.util.Observable的所有属性，
    // 这样Person类构造器中的addEvents和Person.superclass.constructor.call()在实例创建时才会
    //  Person的实例就可以应用Ext的事件相关的on、un等方法和在Person类构造器中的addEvents和Person
    //2、添加一个info()函数，让它返回Person信息。
    Ext.extend(Person, Ext.util.Observable, {
        info: function (event) {
            return this.name + " is " + (event ? "ing" : "doing nothing") + ".";
        }
    });

    //1、创建一个Person实例，然后为它的事件配置好监听器。
    //2、on是addListener的简写，un是removeListener简写
    var person = new Person("Liam");
    person.on("walk", function () {
        this.state = "walk";
        Ext.Msg.alert("event", this.name + " is walking.");
    });
    person.on("eat", function (meal) {
        this.state = "eat";
        Ext.Msg.alert("event", this.name + " is eating " + meal + ".");
    });

    //测试效果
    Ext.get("btnWalk").on("click", function () {
        person.fireEvent("walk");
    });
    Ext.get("btnEat").on("click", function () {
        person.fireEvent("eat", "breakfast");
    });
    Ext.get("btnInfo").on("click", function () {
        Ext.Msg.alert("info", person.info(person.state));
    });
});
```



- 6.[C#.NET 拾遗补漏]13：动态构建LINQ查询表达式
- 7.再聊 Blazor，它是否值得你花时间学习
- 8.使用 .NET 5 体验大数据和机器学习
- 9.[C#.NET 拾遗补漏]12：死锁和活锁的发生及避免
- 10.如约而至，.NET 5.0 正式版发布

积分与排名

积分 - 314937

排名 - 1958

随笔分类

- [01] .NET(73)
- [02] 前端(7)
- [03] 算法(7)
- [04] 数据库(2)
- [05] 工具(5)
- [06] 软技能(5)

最新评论

- 1. Re:.NET 开源项目 Polly 介绍
从知乎到这,感谢大佬分享
--AhF阿锋
- 2. Re:使用 xUnit 编写 ASP.NET Core 单元测试
@.NET开发菜鸟 @Liam Wang 这个解决了吗?...
--Flare_Moon
- 3. Re:再聊 Blazor，它是否值得你花时间学习
十分严重的觉得，这玩艺就是当年的 lightswitch!
--奇微
- 4. Re:在 CentOS 7 中安装 MySQL 8
很详细，多谢。
--樵夫学编程
- 5. Re:.NET 开源项目 Polly 介绍
mark
--吃饭是一件很重要的事
- 6. Re:[C#.NET 拾遗补漏]12：死锁和活锁的发生及避免
学习了，多线程真是大活。
--阿布523
- 7. Re:几个超级实用但很少人知道的 VS 技巧[更新]
点赞！
--咖啡不会醉
- 8. Re:[C#.NET 拾遗补漏]14：使用结构体实现共用体
乍一看，有点像fsharp里的Discriminated Unions
--
- 9. Re:深入解析 C# 的 String.Crea

以上代码展示了在Ext中如何通过继承Ext.util.Observable给一个类自定义事件，到这里，我们大概也了解了addListener/on、addEvents和fireEvent这些函数的基本用法，removeListener/un函数相关内容还会在本文后面介绍。如果要了解Ext.util.Observable的其他细节，可看看[Ext官方API文档](#)的介绍。

Ext.lib.Event

Ext.lib.Event是一个工具类，它封装了不同浏览器的事件处理函数，为上层组件提供了统一功能接口。

对于这个工具类，Ext自带的文档中没有关于这个类的说明，实际中也很少直接用到这个类，只是与事件相关的那些操作最后都会归结为对这些底层函数的调用。

Ext.lib.Event中定义了以下几个主要函数。

getX()、getY()、getXY()，获得发生的事件在页面中的坐标位置：

```
Ext.get("test").on("click", function () {
    alert(this.getX() + "," + this.getY());
});
```

getTarget()，返回事件的目标元素，该函数用来统一IE和其他浏览器使用的e.target和e.srcElement：

```
Ext.get("test").on("click", function (e) {
    var test = e.getTarget();
    alert(test.value);
});
```

on()和un()，这两个函数就不用多说了。

preventDefault()，用于取消浏览器当前事件所执行的默认操作，比如阻止页面跳转。使用这个函数，我是不是可以阻止弹出浏览器鼠标右键菜单呢？我用下面的代码试了下，结果右键菜单并没有被阻止，谁能告诉我为什么？

```
//鼠标右键事件没有被阻止？
Ext.getDoc().on("mousedown ", function (e) {
    if (e.button == "2")
        e.preventDefault();
});
```

stopPropagation()，停止事件传递。比如divTest元素订阅了click事件，它的子元素btnTest被click时，父元素divTest的click事件也会被触发，stopPropagation()就是用来阻止这种事件冒泡的发生：

```
Ext.get("divTest").on("click", function () {
    alert("divTest clicked!");
});
Ext.get("btnTest").on("click", function (e) {
    alert("btnTest clicked!");
    //阻止事件冒泡
    e.stopPropagation();
});
```

stopEvent()，停止一个事件，相当于调用preventDefault()和stopPropagation()两个函数。

支持作者，评论的大多内容跟文章毛关系没有，什么情况？

--ahjszll

10. Re: .NET 6 Preview 3 中 ASP.NET Core 的更新和改进

@胖子黎 你不用新东西，也不影响你...

--Fan\

推荐排行榜

1. [ASP.NET MVC 小牛之路]04 - 依赖注入(DI)和Ninject(170)
2. [ASP.NET MVC 小牛之路]02 - C#知识点提要(130)
3. 几个超级实用但很少人知道的 VS 技巧 [更新](125)
4. [C#.NET 拾遗补漏]08：强大的LINQ(117)
5. [ASP.NET MVC 小牛之路]01 - 理解MVC模式(96)

另外还有一些几乎用不上的函数onAvailable()、getRelatedTarget()等，就不再一一介绍了。

再次说明一下，Ext.lib.Event这个类实际中很少直接用到，用的只是上面讲的一些底层通用函数，并供一些其它和事件相关的类如Ext.EventManager和Ext.EventObject的底层的调用。

Ext.EventManager

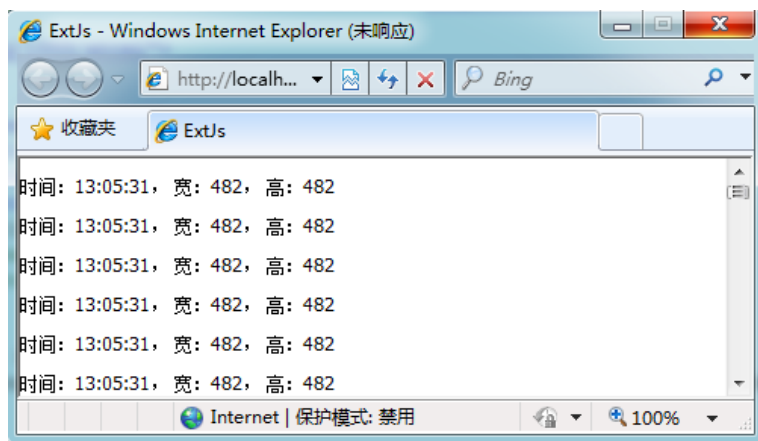
Ext.EventManager，作为事件管理器，定义了一系列事件相关的处理函数。其中最常用的就是onDocumentReady和onWindowResize了。

我们常用的Ext.onReady()就是Ext.EventManager.onDocumentReady()的简写形式，它会在页面文档渲染完毕但图片等资源文件还未下载时调用启动函数。

这里有必要提一下众所周知人人共愤的window.onresize事件：

```
function resizeProcess(width, height) {  
    var p = document.createElement("p");  
    p.innerHTML = "时间：" + new Date().toLocaleTimeString() + "， 宽：" + width + "， 高：" + height;  
    document.body.appendChild(p);  
}  
//原生浏览器resize事件  
window.onresize = function () {  
    resizeProcess(document.documentElement.clientWidth, document.documentElement.clientHeight);  
}
```

当为window.onresize添加了事件处理函数resizeProcess后，会发现resizeProcess会被执行多次，尤其是IE6、IE7、IE8，还会出现假死，动不动就崩掉。



如图，IE8浏览器会直接死掉。真心深恶痛绝IE6、IE7、IE8，要是有朝一日能因为IE11的出现，IE6到IE10都被消灭，那该是多么大快人心的事！

window.onresize事件处理函数被多次乃至无数次触发的问题，网上有不少解决方案，但稍微理想点的方案用起来都挺麻烦。Ext.EventManager下的onWindowResize事件处理函数就非常好的解决了这个问题：

```
Ext.onReady(function () {  
    function resizeProcess(width, height) {  
        var p = document.createElement("p");
```

```
p.innerText = "时间：" + new Date().toLocaleTimeString() + "， 宽：" + width + "，  
document.body.appendChild(p);  
}  
//Ext封装的resize事件  
Ext.EventManager.onWindowResize(function (width, height) {  
    resizeProcess(width, height);  
});  
});
```



如图，每次改变窗口大小，resizeProcess只执行了一次。

Ext.EventManager还有on/addListener、un/removeListener等函数，这些函数都是都过Ext.lib.Event实现的，这里就不再赘述了。

Ext.EventObject

Ext.EventObject是对事件的封装，它提供了丰富的工具函数，帮助我们获得事件相关的信息。通过[Ext.EventObject帮助文档](#)可以了解到，它包含的许多函数都与Ext.lib.Event中的函数功能是相同甚至同名的，如getPageX()、getPageY()、getPageXY()和getTarget()等，这些函数实际上都是通过Ext.lib.Event实现的。

Ext.EventObject对Ext.lib.Event扩展的部分是对鼠标事件和按键事件的增强，定义了一系列按键，可以用来判断某个键是否被按下：

```
Ext.get("text").on("keypress", function (e) {  
    if (e.getKey() == Ext.EventObject.SPACE) {  
        Ext.Msg.alert("提示", "你按了空格键！");  
    }  
});
```

Ext.EventObject将浏览器事件和自定义事件结合在一起使用，是对事件的封装。如果要获得浏览器原始的事件，可通过Ext.EventObject的browserEvent获得。但这种原生事件在不同浏览器中可能会有很大差异，所以Ext.EventObject虽然提供该功能，但一般不建议使用。

给Ext组件添加事件处理函数

添加原生浏览器事件处理函数

我们已经知道可以通过 on/addListener的方式给HTML元素添加事件处理函数，Ext组件也可以通过这种方式添加，如下代码所示：



```
var text = new Ext.form.TextField({  
    id: "text", renderTo: Ext.getBody()  
});  
Ext.get("text").on("mouseover", function (e) {
```

```
        alert("mouse over.");
    });
    //也可以一次添加多个事件处理函数：
    Ext.get("text").on({
        "mouseover": function (e) {
            alert("mouse over.");
        },
        "mouseout": function (e) {
            alert("mouse out.");
        }
    });
```



这种方式可以给任何原生浏览器所支持的事件添加处理函数。但这种方式不能用于容器类的Ext组件，如Ext.form.FieldSet、Ext.form.FormPanel和Ext.Toolbar等。

添加Ext组件事件处理函数

几乎所有Ext组件根据自身的特性对原生事件都行了扩展，另外封装了一套属于自己的事件，这些事件的处理函数会能接收到与该组件相关的事件参数信息。下面代码是给Ext组件添加事件的两种方式：

```


var text1 = new Ext.form.TextField({
    id: "text1", renderTo: Ext.getBody()
});
//任何一个关于导航类键 (arrows、tab、enter、esc等) 被敲击则触发此事件
Ext.getCmp("text1").on("specialkey", function (field,e) {
    alert(field.getValue() + ", " + e.getKey());
});

//也可以在组件创建的时候添加事件处理函数：
var text2 = new Ext.form.TextField({
    id: "text2", renderTo: Ext.getBody(),
    listeners: {
        change: function (field, newValue, oldValue) {
            alert("change:" + newValue);
        },
        blur: function (field) {
            alert("blur:" + field.getValue());
        }
    }
});
```



但这种方式并不支持所有的原生浏览器事件，比如给 Ext.form.TextField 组件通过上面的方式添加 mouseover 事件处理函数是没有效果的。

还有一种通过 handler 属性给 Ext 按钮组件添加事件的方式，这种方式只针对Ext按钮组件，如下：

```

var button = new Ext.Button({
    id: 'button',
    text: '按钮',
    renderTo: Ext.getBody(),
    handler: function () {
        alert("Clicked!!!");
    }
});
```



移除事件处理函数

我们已经知道可通过`un/removeListener`移除某个事件处理函数。值得注意的事，对于原生浏览器事件，用`Ext.fly`获得元素的方式添加的事件处理函数必须用`Ext.fly`获得元素的方式移除，同理，`Ext.get`也是一样。但一般我们用`Ext.fly`而不用`Ext.get`获得元素的方式添加事件处理函数，原因`Ext.fly`更省内存。对于`Ext`组件事件，则必须通过`Ext.getCmp`获得组件的方式移除事件处理函数。如下代码所示：

```
var text = new Ext.form.TextField({
    id: "text", renderTo: Ext.getBody(),
    listeners: {
        change: function (field, n, o) {
            alert("new value : " + n);
        }
    }
});

//事件处理函数
var handlerFn = function (e) {
    alert("mouse over.");
};


//添加mouseover事件处理函数。
Ext.get("text").on("mouseover", handlerFn);
//移除mouseover事件指定引用的处理函数。
Ext.get("text").removeListener("mouseover", handlerFn);
//移除mouseover事件所有的处理函数。
Ext.get("text").removeListener("mouseover");
//用fly获得元素的方式不能移除mouseover处理函数，因为该处理函数是通过get获取元素添加的。
Ext.fly("text").removeListener("mouseover");
//同样，用getCmp获得组件的方式也不能移除mouseover处理函数。
Ext.getCmp("text").removeListener("mouseover");
//移除text元素所有原生浏览器事件的所有处理函数。
Ext.get("text").removeAllListeners();
//获得组件的方式移除change事件所有的处理函数。
Ext.getCmp("text").removeListener("change");
```

对事件的一些额外的控制

事件的额外控制包括让事件只被触发一次、延迟事件处理和控制在多次触发事件的间隔等。通过`on/addListener`函数的第4个参数的属性来实现，让我们通过下面代码来看看常见的几个：

```
var button = new Ext.Button({
    id: 'button',
    text: '按钮',
    renderTo: Ext.getBody()
});

button.on("click",
    function () {
        var el = document.createElement("p");
        el.innerHTML = new Date().toLocaleTimeString();
        document.body.appendChild(el);
    }, this, {
        single: true, //只会执行一次单击事件。
        buffer: 1000, //间隔1秒响应，在响应前点击无效。
        delay: 1000, //从事件触发开始，1后才会执行处理函数。
        stopPropagattion: true, //事件不会向上传递（即停止事件冒泡）。
        preventDefault: true //停止事件默认操作。
```

```
// ...  
}  
);  

```

结束语

ExtJS的事件模型比较复杂，提供的事件处理函数也非常之多，本文短短篇幅不可能面面俱到，只是把常用的做了简单介绍。本人用ExtJS也不久，不免有错差。

希望园友们不吝指教，多多交流，随手点个推荐，以助大家在ExtJS学习之路上快乐进步。

参考：

《深入浅出 Ext JS》

[Ext JS API](#)

作者：精致码农

出处：<http://cnblogs.com/willick>

联系：liam.wang@live.com

本文版权归作者和博客园共有，欢迎转载，但未经作者同意必须保留此段声明，且在文章页面明显位置给出原文连接，否则保留追究法律责任的权利。如有问题或建议，请多多赐教，非常感谢。

分类： [02] 前端

标签： Ext

好文要顶

关注我

收藏该文



精致码农

关注 - 13

粉丝 - 2582

[+加关注](#)

« 上一篇：[ExtJS初探：了解 Ext Core](#)

» 下一篇：[生活沉思录 via 哲理小故事（一）](#)

posted @ 2013-07-06 13:30 精致码农 阅读(17165) 评论(7) 编辑 收藏 举报

[刷新评论](#) [刷新页面](#) [返回顶部](#)

登录后才能查看或发表评论，立即 [登录](#) 或者 [逛逛](#) 博客园首页

编辑推荐：

- 技术管理进阶 —— 技术总监的第一要务
- 带团队后的日常（二）
- WebGIS vs WebGL 图形编程
- .Net Core with 微服务 - Consul 注册中心
- 为什么选择 ASP.NET Core

最新新闻：

- Linux基金会推出开放语音网络（Open Voice Network）
 - 微软公布首批Designed for Xbox显示器名单：普及4K 120Hz与HDR体验
 - 华为程序员频交Linux内核补丁遭质疑，管理员：承认贡献，但请不要琐碎提交
 - 超清动图：原来蚊子吸血这么复杂？才不是一根针在戳你呢
 - 共享充电宝4元/小时成常态？特殊场景涨至6元/小时
- » 更多新闻...