jQuery事件对象及补充点

jQuery的事件对象

只是要是事件就会有事件对象 ，在jQuery里面也是有事件对象的，它的事件对象与原生DOM里面的事件对象是不一样的，有一定的区别

```
<script>
    var btn1 = document.querySelector("#btn1");
    btn1.addEventListener("click", function (event) {
        // 事件对象是有兼容性的
        event = event || window.event;

        console.log(event);
    });

    //------------正同是jQuery的代码 ---------------------------
    $("#btn2").on("click",function(event){
        // 如果要获取事件对象 ,不需要做兼容性处理,jQeury的内部已经做了封装
        console.log(event);
    });
</script>
```

在上面的代码里在，我们可以看到对2个元素同时绑定了事件，一个是通过原生的DOM操作来绑定事件的，一个是通过jQuery框架来绑定的事件的，最后在控制台打印事件对象

上面的打印结果很明显，jQuery的事件对象是一个自定义事件对象，这是因为jQuery把事件做了封装处理，这样事件就没有兼容性了

jQuery的事件对象是一个封装好的自定义对象 ，它屏蔽了不同浏览器之间的兼容性，这样做的到高效快捷的使用

区别

1. jQuery的事件对象当中有一个属性叫which，它代表限鼠标按下了哪一个键，1代表鼠标左键，2代表鼠标中键，3代表鼠标右键，这一点就与原生的事件对象不一样，在原生的事件对象里面，如果我们要判断鼠标按下了哪个键，我们应该用button，但是原生事件的button是有兼容性的
2. jQuery事件里面的stopPropagation()这个方法其实也是后来的封装，它相当于原生事件里面的的2行代码

```
//原生事件里面，如果要阻止事件冒泡
event.cancelBubble = true;
event.stopPropagation();
```

3. 在jQuery的事件对象里面,如果主动的return false则相当于这个事件要阻止默认行为，还要停止事件传播，它相当于下面的三行代码

```
//相当于原生事件对象里在的三行代码
event.cancelBubble = true;
event.stopPropagation();
event.preventDefault();
```

4. jQuery的事件对象是一个封装好的事件对象 ，但是我们仍然可以获取到原生的事件对象

这个地方的originalEvent就可以找到原生的事件对象

jQuery事件委托详解

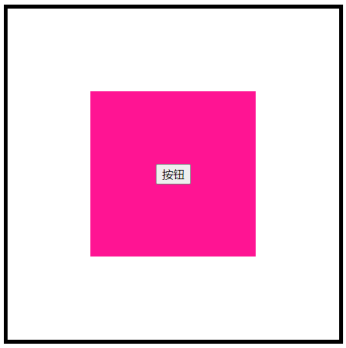
在之前DOM的事件委托里面，我们已经讲到过2个基本的点

1. 事件的绑定者叫event.currentTarget
2. 事件的触发者叫event.target
3. 事件委托者叫event.delegateTarget（这个是jQuery里面独有的）

但是在jQuery的下面，还有一个对象叫事件委托者

我们先看原生的事件委托

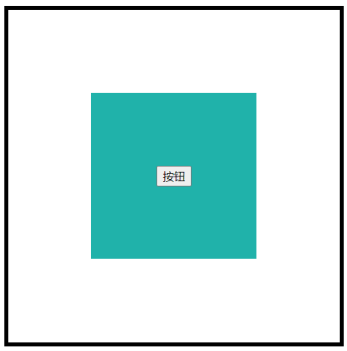
```
<body>
    <div class="big-box">
        <div class="box">
            <button type="button" id="btn1">按钮</button>
        </div>
    </div>
</body>
<script>
    var bigBox = document.querySelector(".big-box");
    var box = document.querySelector(".box");
    var btn1 = document.querySelector("#btn1");
    // 我想绑定box的click事件，但是委托给bigBox，请问，怎么办？
    bigBox.addEventListener("click",function(event){
        //判断事件的触发者
        if(event.target.matches(".box")){
            console.log("事件触发了");
        }
    });
</script>
```



我们点击红色的盒子会产生事件，但是点击红色盒子里面的按钮，事件就没有效果，所以怎么办呢？

jQuery的事件委托

```
<body>
    <div class="big-box">
        <div class="box">
            <button type="button" id="btn1">按钮</button>
        </div>
    </div>
</body>
<script src="./js/jquery-3.6.1.js"></script>
<script>
    $(".big-box").on("click", ".box", function (event) {
        console.log("我是jQuery的事件委托");
    })
</script>
```



在真正的做事件委托的时候，一定是通过下面的代码来实现的

```
page: 217
▼ path: Array(7)
  ▶ 0: button#btn1
  ▶ 1: div.box
  ▶ 2: div.big-box
  ▶ 3: body
  ▶ 4: html
  ▶ 5: document
  ▶ 6: Window {window: Window, self: Window, document: document, name: '', location: Location, …}
    length: 7
  ▶ [[Prototype]]: Array(0)
```

```
<body>
    <div class="big-box">
        <div class="box">
```

```
            <button type="button" id="btn1">按钮</button>
        </div>
    </div>
</body>
<script>
    var bigBox = document.querySelector(".big-box");
    var box = document.querySelector(".box");
    var btn1 = document.querySelector("#btn1");
    // 我想绑定box的click事件，但是委托给bigBox，请问，怎么办？
    bigBox.addEventListener("click",function(event){
        console.log(event);
        for(var i=0;i<event.path.length;i++){
            // console.log(event.path[i]);
            if(event.path[i] instanceof Element &&
event.path[i].matches(".box")){
                console.log("我触发了事件");
            }
        }
    });
</script>
```

jQuery事件绑定的方式

在目前的jQuery的版本里面，如果我们想绑定一个元素的事件，可以通过下面的方式来进行

```
<body>
    <div class="box">
        <button type="button" id="btn1">按钮1</button>
        <button type="button" id="btn2">按钮2</button>
        <button type="button" id="btn3">按钮3</button>
        <button type="button" id="btn4">按钮4</button>
        <button type="button" id="btn5">按钮5</button>
        <button type="button" id="btn6">按钮6</button>
        <button type="button" id="btn7">按钮7</button>
    </div>
</body>
<script src="./js/jquery-3.6.1.js"></script>
<script>
    $(function () {
        $("#btn1").click(function (event) {
            console.log("第一种：事件方法");
        });
        $("#btn2").on("click", function (event) {
            console.log("第二种:on完成");
        });
        $("#btn3").one("click", function (event) {
            console.log("第三种：one方式");
        });
        // 事件委托
```

```
        $(".box").on("click", "#btn4", function (event) {
            console.log("第四种：事件委托");
        });
        // ----------------------------------------------

        //jquery1.5-1.7的版本的事件绑定
        //这种方式，不要用，这是旧版本里面的
        $("#btn5").bind("click",function(event){
            console.log("第五种：bind绑定");
        });
        //移除bind绑定的事件
        $("#btn5").unbind("click");

        //这种是旧版本的事件委托 ，不要用
        $(".box").delegate("#btn6","click",function(event){
            console.log("第六种：委托");
        });
        $(".box").undelegate("#btn6","click");

        //jQuery1.5版本之前
        /*
         $("#btn7").live("click",function(event){

         });

         $("#btn").die("click",function(event){

         })
         */
    })
</script>
```

一定看请楚上面的代码，事件绑定在不同的版本里面有不同的方式 ，终归而言有以下几种

1. 万能的 on/off
2. 单次的 one
3. 旧版本的 bind/unbind
4. 旧版本的事件委托 delegate/undelegate
5. 再旧一点的版本 live/die

jQuery的扩展方法

jQuery是一系列方法的集合，它把我们常用的就去进行了一些封装，所以我们可以在工作开发当中直接使用这些方法，但是如果我们想往上这个上面添加一些方法呢?

```
$.get();
$.ajax();
$.each();
$.makeArray();
$(".div1").toArray()
$("button").show();
```

❓ **思考**：我们能否在这些方法再去扩展一些自己的方法，如下

```
$.biaogege();
$(".div1").hello();
```

这些方法都是jQuery不存在的，所以我们如果想使用就要自己扩展！

jQuery内部提供我们扩展方法的一个操作，可以直接来进行

第一种情况：直接在$上面去扩展方法

```
 $.extend({
     biaogege:function(){
         console.log("我是标哥哥");
     }
 })
$.biaogege();
```

第二种情况：直接在选中的元素上面扩展

```
// 如果想在选择器上面扩展方法
$.fn.extend({
    hello:function(){
        console.log("你好啊，我是hello的方法")
    }
})
// $.hello();        报错
$("#btn1").hello();   //正常
```

有了这个东西，我们就可以结合我们之前所学习的模板引擎来完成相应的操作

```
<body>
    <ul class="ul1">
    </ul>
    <template id="temp1">
        {{each list item index}}
        <li>{{item}}</li>
        {{/each}}
    </template>
</body>
<script src="./js/template-web.js"></script>
<script src="./js/jquery-3.6.1.js"></script>
<script>
    $.fn.extend({
```

```
        render:function(tempid,data){
            var htmlStr = template(tempid,data);
            // 这里的this就是你选择器选中的东西
            this.html(htmlStr);
        }
    })

    $(function () {
        var arr = ["张三", "李四", "王五", "赵六"];
        // var htmlStr = template("temp1",{
        //     list:arr
        // });
        // console.log(htmlStr);
        // $(".ul1").html(htmlStr)

        // 我想在jQuery上面扩展一个方法，直接通过下面的代码就可以完成
        $(".ul1").render("temp1",{
            list:arr
        })
    })
</script>
```

上面的代码就是直接在jQuery的选择器上面扩展了一个render的方法，用于渲染模板