

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

```
1 <script>
2     var btn1 = document.querySelector("#btn1");
3     btn1.addEventListener("click", function (event) {
4         // 事件对象是有兼容性的
5         event = event || window.event;
6
7         console.log(event);
8     });
9
10    //-----正同是jQuery的代码 -----
11    $("#btn2").on("click",function(event){
12        // 如果要获取事件对象，不需要做兼容性处理，jQuery的内部已经
        做了封装
13        console.log(event);
14    });
15 </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

```
▶ PointerEvent {isTrusted: true, pointerId: 1, width: 1, height: 1, pressure: 0, ...} 01事件对象.html:24
▶ jQuery.Event {originalEvent: PointerEvent, type: 'click', target: button#btn2, currentTarget: button#btn2, isDefaultPrevented: f, ...} 01事件对象.html:30
```

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2行代码

```
1 //原生事件里面，如果要阻止事件冒泡
2 event.cancelBubble = true;
3 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

```
1 //相当于原生事件对象里在的三行代码
2 event.cancelBubble = true;
3 event.stopPropagation();
4 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

```
live server connected.
03jQuery事件对象.htm
jQuery.Event {originalEvent: PointerEvent, type: 'click', target: button#btn1, currentTarget: button#btn1,
  faultPrevented: f, ...}
  ▶ currentTarget: button#btn1
  data: undefined
  ▶ delegateTarget: button#btn1
  ▶ handleObj: {type: 'click', origType: 'click', data: undefined, guid: 2, handler: f, ...}
  ▶ isDefaultPrevented: f returnTrue()
  ▶ isPropagationStopped: f returnTrue()
  jQuery361002558946026435227: true
  ▶ originalEvent: PointerEvent 这一个就是原生的事件对象, 如果在开发当中需要使用原生的事件对象, 就找它
    isTrusted: true
    altKey: false
    altitudeAngle: 1.5707963267948966
    azimuthAngle: 0
```

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

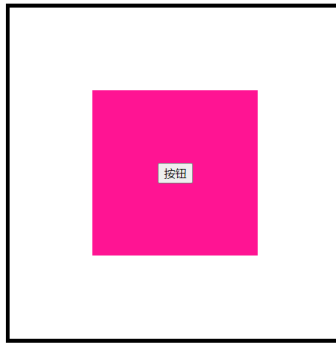
在之前DOM的事件委托里面, 我们已经讲到过2个基本的点

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

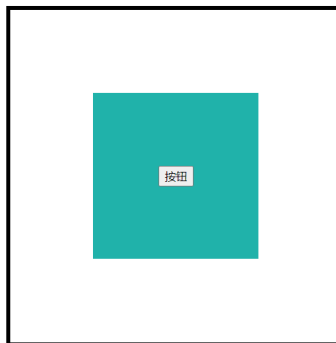
```
1 <body>
2   <div class="big-box">
3     <div class="box">
4       <button type="button" id="btn1">按钮</button>
5     </div>
6   </div>
7 </body>
8 <script>
9   var bigBox = document.querySelector(".big-box");
10  var box = document.querySelector(".box");
11  var btn1 = document.querySelector("#btn1");
12  // 我想绑定box的click事件, 但是委托给bigBox, 请问, 怎么办?
13  bigBox.addEventListener("click", function(event){
14    //判断事件的触发者
15    if(event.target.matches(".box")){
16      console.log("事件触发了");
17    }
18  });
19 </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

```
1 <body>
2   <div class="big-box">
3     <div class="box">
4       <button type="button" id="btn1">按钮</button>
5     </div>
6   </div>
7 </body>
8 <script src="./js/jquery-3.6.1.js"></script>
9 <script>
10   $(".big-box").on("click", ".box", function (event) {
11     console.log("我是jQuery的事件委托");
12   })
13 </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: 21/
▼ 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)
```

```
1 <body>
2   <div class="big-box">
3     <div class="box">
```

```

4         <button type="button" id="btn1">按钮</button>
5     </div>
6 </div>
7 </body>
8 <script>
9     var bigBox = document.querySelector(".big-box");
10    var box = document.querySelector(".box");
11    var btn1 = document.querySelector("#btn1");
12    // 我想绑定box的click事件，但是委托给bigBox，请问，怎么办？
13    bigBox.addEventListener("click",function(event){
14        console.log(event);
15        for(var i=0;i<event.path.length;i++){
16            // console.log(event.path[i]);
17            if(event.path[i] instanceof Element &&
event.path[i].matches(".box")){
18                console.log("我触发了事件");
19            }
20        }
21    });
22 </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

```

1 <body>
2     <div class="box">
3         <button type="button" id="btn1">按钮1</button>
4         <button type="button" id="btn2">按钮2</button>
5         <button type="button" id="btn3">按钮3</button>
6         <button type="button" id="btn4">按钮4</button>
7         <button type="button" id="btn5">按钮5</button>
8         <button type="button" id="btn6">按钮6</button>
9         <button type="button" id="btn7">按钮7</button>
10    </div>
11 </body>
12 <script src="./js/jquery-3.6.1.js"></script>
13 <script>
14     $(function () {
15         $("#btn1").click(function (event) {
16             console.log("第一种：事件方法");
17         });
18         $("#btn2").on("click", function (event) {
19             console.log("第二种: on完成");
20         });
21         $("#btn3").one("click", function (event) {
22             console.log("第三种：one方式");
23         });
24         // 事件委托

```

```

25     $(".box").on("click", "#btn4", function (event) {
26         console.log("第四种：事件委托");
27     });
28     // -----
29
30     //jquery1.5-1.7的版本的事件绑定
31     //这种方式，不要用，这是旧版本里面的
32     $("#btn5").bind("click",function(event){
33         console.log("第五种：bind绑定");
34     });
35     //移除bind绑定的事件
36     $("#btn5").unbind("click");
37
38     //这种是旧版本的事件委托 ， 不要用
39     $(".box").delegate("#btn6","click",function(event){
40         console.log("第六种：委托");
41     });
42     $(".box").undelegate("#btn6","click");
43
44     //jQuery1.5版本之前
45     /*
46     $("#btn7").live("click",function(event){
47
48     });
49
50     $("#btn").die("click",function(event){
51
52     })
53     */
54 })
55 </script>

```

一定看清楚上面的代码，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？

```
1 $.get();
2 $.ajax();
3 $.each();
4 $.makeArray();
5 $(".div1").toArray()
6 $("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

```
1 $.biaogege();
2 $(".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

```
1 $.extend({
2     biaogege:function(){
3         console.log("我是标哥哥");
4     }
5 })
6 $.biaogege();
```

第二种情况：直接在选中的元素上面扩展

```
1 // 如果想在选择器上面扩展方法
2 $.fn.extend({
3     hello:function(){
4         console.log("你好啊，我是hello的方法")
5     }
6 })
7 // $.hello();      报错
8 $("#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

```
1 <body>
2     <ul class="ul1">
3     </ul>
4     <template id="temp1">
5         {{each list item index}}
6         <li>{{item}}</li>
7         {{/each}}
8     </template>
9 </body>
10 <script src="./js/template-web.js"></script>
11 <script src="./js/jquery-3.6.1.js"></script>
12 <script>
13     $.fn.extend({
```

```

14         render:function(tempid,data){
15             var htmlStr = template(tempid,data);
16             // 这里的this就是你选择器选中的东西
17             this.html(htmlStr);
18         }
19     })
20
21     $(function () {
22         var arr = ["张三", "李四", "王五", "赵六"];
23         // var htmlStr = template("temp1",{
24         //     list:arr
25         // });
26         // console.log(htmlStr);
27         // $(".ul1").html(htmlStr)
28
29         // 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
30         $(".ul1").render("temp1",{
31             list:arr
32         })
33     })
34 </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