高级Day 12---Jquery操作和事件

一、DOM操作的分类

一般情况下，DOM操作分三个方面：DOM Core、HTML-DOM和CSS-DOM

1.1 DOM Core

并不是JavaScript所专有，也不仅仅只操作HTML文档。只要文档是DOM结构，都可以使用核心DOM中的方法，如html文档和xml文档都可以使用getElement(s)相关的方法获得元素

1.2 HTML-DOM

专门针对HTML的一种DOM操作。提供了一些专属的方法。例如，直接通过document对象获取标签对象或操作标签的属性。这两种DOM操作都是HTML-DOM。

例如：document.froms element.src

1.3 CSS-DOM

CSS-DOM是针对CSS的操作。主要作用就是获取和数组style对象的各种属性。让网页呈现出不同的效果。

例如：element.style.color = "red";

二、JQuery基础DOM操作

2.1 获取和设置元素的内容

> 三个方法：text()、html()和val()

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <script src="libs/jquery-3.1.1.js" type="text/javascript" charset="utf-8"></script>
        <script type="text/javascript">
            $(function () {
                //获取得到的标签的内的文本，不包括内部的标签。但是内部标签与标签中的空格会获取到
                var text = $(".box").text();    // ele.innerText
                console.log(text);
                var html = $(".box").html();
                //标签内的所有内容，包括子标签本身   ele.innerHTML
                console.log(html);

                //上面的方法不传入参数表示只是获取，如果传入字符串表示设置
//                $(".box").text("哈哈哈");
                //设置html，如果有标签，则会解析出来
                $(".box").html("<button>点我啊</button>")
            })

            $(function () {
                $("div button").click(function () {
                 //获取一个标签的value值(如果有)
                    var inputText = $("div input").val();
                    alert(inputText);

                })
            })
        </script>

    </head>
    <body>
        <div class="box">
            <a href="http://www.yztcedu.com">点我去育知同创</a>
        </div>
       <div>
           <input type="text" name="user" id="user" value="" />
           <br />
           <button>获取输入框中的值</button>
        </div>
    </body>
</html>
```

2.2 获取和修改元素的属性

> attr(属性名)方法操作

```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="UTF-8">
        <title></title>
        <script src="libs/jquery-3.1.1.js" type="text/javascript" charset="utf-8"></script>
        <script type="text/javascript">
            $(function() {
                //获取属性href的值
                var $href = $("div a").attr("href");
                console.log($href);
                //点击按钮的时候更换属性的值
                $("div button").click(function () {
                    $("div a")
                    .attr("href", "http://www.sina.com.cn") //参数1：属性名  参数2：属性值
                    .attr("title", "浪浪")
                    .text("浪一下");
                })
            })
        </script>

    </head>

    <body>
        <div class="box">
            <a href="http://www.yztcedu.com">点我去育知同创</a>
        </div>
        <div><button>更换友情链接</button></div>
    </body>


</html>
```

> 如果需要同时设置多个属性也可以把多个属性封装在一个对象中

```
$("div a")
  .attr({
        href:"http://www.sina.com.cn",
        title:"浪浪"
    })
  .text("浪一下");
```

2.3 元素的样式操作

这里的样式操作主要是针对一个元素的class属性进行操作。一般有一些潜规则：id是给js用的，class是给css用的。所以，大家希望用class去设置样样式。

对样式的操作包括：获取class，设置class，追加class，删除class，切换class，判断是否有class属性

2.3.1 获取和设置样式

> class也是属性，所以获取和设置的方法与获取和设置属性的方法一致。

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
            div.nomal{
                width: 200px;
                height: 200px;
                background: pink;
                border: 1px dashed blue;
            }
            div ul{
                list-style: none;
            }
            .big{
                width: 400px;
                height: 400px;
                border: 1px dashed green;
            }
            .color{
                background: gray;
            }

        </style>
        <script src="libs/jquery-3.1.1.js" type="text/javascript" charset="utf-8"></script>

        <script type="text/javascript">
            $(function () {
                $("li:nth-child(1)").click(function () {
                    $("div#box").attr("class", "big");
                })

                $("li:nth-child(2)").click(function  () {
                    $("div#box").attr("class", "color");

                })
            })
        </script>
    </head>
    <body>
        <div class="nomal" id="box">我是div中的元素</div>

        <div>
            <ul>
                <li><button>点我变大</button></li>
                <li><button>点我变色</button></li>
            </ul>
        </div>
    </body>

</html>
```

2.3.2 追加样式

替换样式会把原来的全部更新，有的时候，我们只想向更新某个样式，则使用追加。

> CSS规定，一个元素的class属性可以多个值，多个值之间用空格隔开。表示合并这些样式的值。如果有重复的，则以后面的为准
>
> JQuery使用addClass方法来追加样式.
>
> addClass方法可以完成追加的效果

```
$(function () {
    $("ul li:first").click(function () {
        //追加样式
        $("#box").addClass("big");
    })

    $("li:eq(1)").click(function  () {
        $("#box").addClass("color");

    })
})
```

2.3.3 移除样式

removeClass(样式名)

2.4.4 切换样式

> toggleClass(样式)
>
> 切换样式，如果类名存在则删除，不存在则添加

```
$(function () {
    $("li:last").click(function () {
        //该方法会自动判断这个class是否存在，如果存在就删除，不存在就添加
        $("#box").toggleClass("color");
    })

})
```

2.4.5 判断是否有指定的class

> hasClass(样式)

2.4 each方法的使用

```
<script type="text/javascript">
    $(function () {
        //变量选择到的所有元素。  传入的参数中有两个参数：参数1：遍历的元素的下标 参数2：遍历的那个元素(dom对象，不是jq对象)
        $("div").each(function (index, ele) {
            $(ele).html("呵呵呵")
        })

    })
</script>
```

三、CSS-DOM操作

CSS-DOM操作就是操作读取和设置sytle对象的各种操作。使用style属性只能获取行内style属性，内部样式和外部样式则无法获取。用JQuery一切都变得简单很多。

3.1 使用css方法获取和设置css属性

css方法既可以获取行间样式属性，也可以获取内部和外部样式的属性

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
            .box{
                width: 300px;
                height: 300px;
                border: 1px dashed red;
                background: pink;
            }
        </style>
        <script src="libs/jquery-3.1.1.js" type="text/javascript" charset="utf-8"></script>
        <script type="text/javascript">
            $(function() {
                $("div button").click(function () {
                    var bg = $(".box").css("backgroundColor"); //获取背景色。返回的是rgb颜色
//                  alert(bg);
                    //设置背景色。
//                  $(".box").css("background-color", "#808080");
                    //也可是使用对象设置多个css属性
                    $(".box").css({"background-color":"#808080", "border":"10px solid green"});
                    //单独设置透明度。第二个参数是number可以不加引号。(加上也可以的)
                    $(".box").css("opacity",0.5);
                    //深圳宽高属性既可以用css方法，也可以使用width方法和height方法
//                  $(".box").css("width", "800px");  //使用css设置宽
                    //使用width方法。只传递数字表示为1200px。 也可以写"1200px"
                    $(".box").width(1200);
                })
            })
        </script>
    </head>
    <body>
        <div id="box" class="box">

        </div>
        <div>
            <button>更换颜色</button>
        </div>
    </body>
</html>
```

3.2 offset方法

获取这个元素在当前文档的相对偏移。其中返回的对象包含两个属性，left和top

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
            *{
                margin: 0;
                padding: 0;
            }
            .box{
                width: 300px;
                height: 300px;
                border: 1px dashed red;
                background: pink;

            }
        </style>
        <script src="libs/jquery-3.1.1.js" type="text/javascript" charset="utf-8"></script>
        <script type="text/javascript">
            $(function() {
                $("button").click(function () {
                    //获取相对文档偏移量
                    var $off = $(".box").offset();
                    console.log($off.left + "  " + $off.top);
                    //设置相对文档偏移量
                    $(".box").offset({top:300, left:300})
                })
            })
        </script>
    </head>
    <body>
        <div id="" style="position:absolute;left: 20px;">
            <div id="box" class="box"></div>
        </div>
        <div>
            <button>获取偏移量</button>
        </div>
    </body>
</html>
```

3.3 position方法

获取标签相对于参照定位标签的偏移量

position（）方法的返回值和offset()一样，有top和left

一般只用来获取而不是用来设置。

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
            *{
                margin: 0;
                padding: 0;
            }
            .box{
                width: 400px;
                height: 400px;
                border: 1px dashed red;
                background: pink;
                position: relative;
                left: 50px;
            }
            .box div{
                width: 300px;
                height: 300px;
                border: 1px solid green;
                background: #808080;
                position: absolute;
                left: 20px;
            }
        </style>
        <script src="libs/jquery-3.1.1.js" type="text/javascript" charset="utf-8"></script>
        <script type="text/javascript">
            $(function() {
                $("button:first").click(function () {
                    var $position = $(".box div").position();
                    console.log($position.left + "  " + $position.top);
                })
            })
        </script>
    </head>
    <body>
        <div class="box">
            <div></div>
        </div>
        <div>
            <button>获取偏移量</button>
            <button>设置偏移量</button>
        </div>
    </body>
</html>
```

3.4 scrollTop()和scrollLeft()方法

> 获取某个元素的滚动条距离上端和左端的滚动的距离。

> 也可以设置让滚动条去滚动

```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
            p {
                width: 400px;
                height: 100px;
                overflow-x: scroll;
                overflow-y:
            }
        </style>
        <script src="libs/jquery-1.12.3.js" type="text/javascript" charset="utf-8"></script>
        <script type="text/javascript">
            $(function() {
                $("button:eq(0)").click(function() {
                    var top = $("p:first").scrollTop();//获取垂直滚动条距离顶端的距离
                    var left = $("p:first").scrollLeft();//获取水平滚动条距离左端的距离
                    console.log(top + "  " + left);
                })
                var per = 1;
                var $p = $("p:first");
                var timerId;
                //自动滚屏
                $("button:eq(1)").click(function() {
                    timerId = setInterval(function() {
                        var top = $p.scrollTop();
                        $p.scrollTop(top + per);//更改垂直滚动条距离顶端的距离，则滚动条滚起来
                    }, 50);

                })
                $("button:eq(2)").click(function() {
                    //取消自动滚屏
                    clearInterval(timerId);

                })

            })
        </script>
    </head>

    <body>




        <p>我要滚动了 1 我要滚动了 1我要滚动了 1 我要滚动了 1我要滚动了 1 我要滚动了 1我要滚动了 1 我要
```

```
滚动了 1我要滚动了 1 我要滚动了 1我要滚动了 1 我要滚动了 1我要滚动了 1 我要滚动了 1我要滚动了 1 我要滚动
了 1我要滚动了 1 我要滚动了 1我要滚动了 1 我要滚动了 1我要滚动了 1 我要滚动了 1我要滚动了 1 我要滚动了 1
我要滚动了 1 我要滚动了 1我要滚动了 1 我要滚动了 1我要滚动了 1 我要滚动了 1我要滚动了 1 我要滚动了 1我要
滚动了 1 我要滚动了 1我要滚动了 1 我要滚动了 1我要滚动了 1 我要滚动了 1我要滚动了 1 我要滚动了 1我要滚动
了 1 我要滚动了 1我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 我要滚动了 1 我要滚动了 1 我要滚动了
1 我要滚动了 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 我要滚动了 1 我要滚动了 1 我要滚动了 1 我
要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我
要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我
要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我
要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我
要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我
要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我
要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我
要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1 我要滚动了 1
         </p>
         <p>我要滚动了 2 </p>
         <button>获取滚动距离 </button>
         <button>自动滚屏 </button>
         <button>取消自动滚屏 </button>
     </body>

</html>
```

四、DOM节点操作

4.1 创建节点

4.1.1 创建元素节点

```
var $li1 =$("<li></li>");    //创建一个li节点
```

4.1.2 创建文本节点

```
var $li1 =$("<li>JavaScript</li>"); //创建一个li节点和一个文本节点。元素节点和文本节点一起创建的
```

4.1.3 创建属性节点

```
var $li1 = $("<li id='js'>javaScrpt</li>"); //创建属性节点同元素节点文本节点一起创建
```

4.2 插入节点

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <script src="libs/jquery-1.12.3.js" type="text/javascript" charset="utf-8"></script>
        <script type="text/javascript">
            $(function () {
                var $li1 = $("<li id='js'>javaScrpt</li>");
                var $ul = $("ul");
//              $ul.append($li1);  //向ul中追加一个li标签。 插入之后 li是ul的子标签
//              $li1.appendTo($ul);  // 将$li1追加到$ul中。  其实是颠倒了append的逻辑
//              $ul.prepend($li1);  //向ul的最前面追加一个li标签
//              $ul.after($li1);  //在匹配的元素之后插入。  插入之后li是ul的兄弟标签
                $ul.before($li1);  ////在匹配的元素之前插入。  插入之后li是ul的兄弟标签
            })
        </script>
    </head>
    <body>
        <ul>
            <li>c++</li>
        </ul>
    </body>
</html>
```

4.3 删除节点

> remove、detach和empty三个方法可以完成删除节点的操作

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <script src="libs/jquery-1.12.3.js" type="text/javascript" charset="utf-8"></script>
        <script type="text/javascript">
            $(function () {
                $(".btns button:eq(0)").click(function () {
                    //删除找到的标签。删除的标签会返回。  绑定的事件会丢失。
                    var $removed = $(".languages li:nth-child(2)").remove();
                    //追加到末尾
                    $("ul.languages").append($removed);

                })
                $(".btns button:eq(1)").click(function () {
                    //与remove不同的是，他绑定的事件、附加的数据还在
                    var $removed = $(".languages li:nth-child(2)").detach();
                    $("ul.languages").append($removed);

                })

                $(".btns button:eq(2)").click(function () {
                    //清空找到的节点的所有内容，包括所有后代节点。节点还在。  这个其实并没有清除节点
                    $(".languages li:nth-child(2)").empty()
                })
            })
        </script>
    </head>
    <body>
        <ul class="languages">
            <li>c++</li>
            <li>javaScript</li>
            <li>java</li>
            <li>php</li>
        </ul>
        <ul class="btns">
            <li><button>使用remove删除第二个li</button></li>
            <li><button>使用detach删除第二个li</button></li>
            <li><button>清空元素内容</button></li>
        </ul>
    </body>
</html>
```

4.4 复制节点

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <script src="libs/jquery-1.12.3.js" type="text/javascript" charset="utf-8"></script>
        <script type="text/javascript">
            $(function () {
                $(".btns button:eq(0)").click(function () {
                   //复制节点，并添加到尾部
                    $(".languages li:first").clone().appendTo($(".languages"));
                })
            })
        </script>
    </head>
    <body>
        <ul class="languages">
            <li>c++</li>
            <li>javaScript</li>
            <li>java</li>
            <li>php</li>
        </ul>
        <ul class="btns">
            <li><button>复制第一个节点，然后追加到最后</button></li>
        </ul>
    </body>
</html>
```

4.5 替换节点

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <script src="libs/jquery-1.12.3.js" type="text/javascript" charset="utf-8"></script>
        <script type="text/javascript">
            $(function () {
                $(".btns button:eq(0)").click(function () {
                    //找到的所有的li用参数中的节点替换
//                  $(".languages li").replaceWith("<li>我要全部干掉你们</li>");
                    //或者用replaceAll,效果同上。
                    $("<li>我要全部干掉你们</li>").replaceAll($(".languages li"));
                })
            })
        </script>
    </head>
    <body>
        <ul class="languages">
            <li>c++</li>
            <li>javaScript</li>
            <li>java</li>
            <li>php</li>
        </ul>
        <ul class="btns">
            <li><button>替换所有的节点</button></li>

        </ul>
    </body>
</html>
```

4.6 包裹节点

> b.wrap(a)　每一个b分别用一个a包裹
>
> b.wrapAll(a)　所有的b用同一个a包过，如果有其他元素则扔到包裹的后面
>
> b.wrapInner(a);　用a去包过b的内容。不包裹b这个标签本身。

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <script src="libs/jquery-1.12.3.js" type="text/javascript" charset="utf-8"></script>
        <script type="text/javascript">
            $(function () {
                $(".btns button:eq(0)").click(function () {
                    //用div把 .languages 包裹起来。  如果多个元素需要包裹，则会有多个div
//                  $(".languages").wrap("<div></div>");
                    //所有的用一个div包裹起来。  如果中间有其他元素，其他元素则放置包裹之后
//                  $("ul").wrapAll("<div></div>");
                    //.languages li 标签中的内容用span包括。  注意：不包裹标签字节，只包括标签的内容
                    $(".languages li").wrapInner("<sapn></span>");
                })
            })
        </script>
    </head>
    <body>
        <ul class="languages">
            <li>c++</li>
            <li>javaScript</li>
            <li>java</li>
            <li>php</li>
        </ul>
        <p>擦，我竟然被扔出来了</p>
        <ul class="btns">
            <li><button>给ul包裹一个div</button></li>

        </ul>
    </body>
</html>
```

五、JQuery中的事件

5.1 基础事件

5.1.1 事件绑定

> 元素对象.bind(事件，[可选的额外参数], 事件处理函数)
>
> 可选参数2一般省略
>
> 参数2的事件类型：
>
> 第 1 个参数是事件类型，类型包括：blur、focus、load、resize、scroll、unload、click、dblclick、mousedown、mouseup、mousemove、mouseover、mouseout、mouseenter、mouseleave、change、select、submit、keydown、keypress、keyup 和 error 等，当然也可以是自定义名称。

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title>Document</title>
    <script src="libs/jquery-1.12.3.js" type="text/javascript" charset="utf-8"></script>
    <script type="text/javascript">
        $(function(){
            //这里绑定的事件是没有on的。
             $("button:first").bind("click", function () {
                 alert("我被点击了");
             })
        })
    </script>
</head>
<body>
    <button>我有事件了</button>
</body>
</html>
```

5.1.2 解除绑定

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title>Document</title>
    <script src="libs/jquery-1.12.3.js" type="text/javascript" charset="utf-8"></script>
    <script type="text/javascript">
        $(function(){
            var firstBtn = $("button:first");
            firstBtn.bind("click", function () {
                alert("我被点击了");
            })

            firstBtn.bind("mouseover", function () {
                $(this).css("background", "red");
            })

            firstBtn.bind("mouseout", function () {
                $(this).css("background", "#FFC0CB");
            })

            $("button:eq(1)").click(function () {
                firstBtn.unbind("click");  //解除click事件的绑定
            })

            $("button:eq(2)").click(function () {
                firstBtn.unbind(); //不传参数表示解除所有的事件绑定
            })
        })
    </script>
</head>
<body>
    <button>我有事件了</button>
    <button>解除点击事件</button>
    <button>解除所有事件</button>
</body>
</html>
```

5.1.3 简写事件

经常用到的事件，jquery也提供了一种简写的方式，效率一样，只是代码写法简化了

```
$("button:eq(1)").click(function () {
    alert("解除")
    firstBtn.unbind("click");  //解除click事件的绑定
})
```

5.1.4 复合事件

JQuery提供了两个合成事件：hover和toggle

1. hover(f1, f2)用于模拟光标悬停事件，当光标移动到元素上的时候，会触发指定的第一个函数，当光标移除这

个元素时，会触发第二个元素

2. 功能1：toggle(f1, f2, ... fn) 用于模拟鼠标连续单击事件，当第一次单击时只需第一个函数，当第二次单击执行第二个函数，...，执行完毕之后再循环来。功能2：不传任何参数，可以让元素隐藏或显示。 但是从1.9版本开始，只剩下了功能2

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <style type="text/css">
            #box{
                width: 200px;
                height: 200px;
                border: 1px solid red;
                text-align: center;
                line-height: 200px;
            }
        </style>
        <script src="libs/jquery-3.1.1.js" type="text/javascript" charset="utf-8"></script>
        <script type="text/javascript">
            $(function () {
                $("div").hover(function () {
                    //鼠标进入元素区域
                    $(this).css("background", "red");
                }, function () {
                    //鼠标离开元素区域
                    $(this).css("background", "pink")
                });

                /*$("div").toggle(function  () {
                    $("p").html("第一次点我")
                }, function () {
                    $("p").html("第二次点我")
                }, function () {
                    $("p").html("第三次点我")
                })*/
                $("button").click(function () {
                    $("div").toggle();  //如果隐藏就显示，如果显示就隐藏。
                })
            })

        </script>
    </head>
    <body>
        <div id="box">
            我是个盒子
        </div>
        <p>哈哈哈</p>
        <button>点我</button>
    </body>
</html>
```

5.2 事件对象

5.2.1 事件对象的获取

获取事件对象非常简单，只需要在绑定事件的时候的事件处理函数添加一个参数即可。例如：

ele.click(function(event){ }); event就是事件对象。函数触发的时候，事件对象创建，函数执行完毕事件对象销毁，只能在函数内部使用。

```
<script type="text/javascript">
    $(function() {
        //匿名函数的参数就是事件对象。执行的时候浏览器会自动创建事件对象，并传入函数中。
        //这里的事件对象，已经经过了JQuery的封装。是JQuery对象
        $("span").click(function(event) {
            console.log(event);
        })
        $("span").bind("click", function(event) {
            console.log(event);
        })
    })
</script>
```

5.2.2 事件对象的常用属性

经过JQuery封装后的事件对象，有如下常用的属性可供使用。

1. type属性 获取事件类型
2. preventDefault()方法 阻止事件的默认行为。兼容各种浏览器。
3. stopPropation()方法 阻止事件的冒泡
4. target属性 获取到触发事件的元素
5. pageX和pageY属性 光标相对于页面的距离。如果有滚动条，还要加上滚动条的距离
6. which属性。 如果是鼠标事件则获取鼠标的左(1)中(2)右(3)键，如果是键盘则获取键盘的按键。
7. metaKey属性。获取ctrl按键。

```
<script type="text/javascript">
  $(function() {
    $("span").click(function(event) {
      console.log("事件类型: " + event.type);
      console.log("targe属性: " + event.target);
      console.log("pageX:" + event.pageX);
      console.log("pageY:" + event.pageY);
      console.log("which:" + event.which)
    })
  })
</script>
```

5.2.3 事件的冒泡以及默认处理

preventDefault()阻止默认行为，stopPropation()阻止冒泡。 事件函数中返回false，可以同时停止默认行为和冒泡。所以，可以直接返回false，而不要调用上述两个方法。

5.3 高级事件

5.3.1 事件模拟操作

1. ele.trigger(事件类型)

```
<script type="text/javascript">
    $(function  () {
        $("button").click(function() {
            alert("我被点击了");
        })
        //模拟用户的点击行为
        $("button").trigger("click");
        //简写
        //$("button").click();
    })
</script>
```

2. trigger也可以触发自定义事件

```
<script type="text/javascript">
    $(function  () {
        //自定义事件
        $("button").bind("myClick", function () {
            alert("这个是我自定义的事件");
        })
        $("button").trigger("myClick");

    })
</script>
```

3. 执行默认操作：trigger触发的事件，不仅仅触发了事件函数，也会触发浏览器的默认操作。比如，给input注册一个focus事件，则不但触发focus事件，还会触发浏览器的默认事件，使input获得焦点。使用trigerHanddler可以避免浏览器的默认操作

```
<script type="text/javascript">
    $(function  () {
        $("input:last").focus(function () {
            console.log("擦，我获得焦点了");
        })
        $("input:last").triggerHandler("focus");
//      $("input:last").trigger("focus");
    })
</script>
```

5.3.2 事件命名空间

JQuery为了方便管理事件，给事件提出了命名空间的概念。 语法：事件.命名空间

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <script src="libs/jquery-1.12.3.js" type="text/javascript" charset="utf-8"></script>
        <script type="text/javascript">
            $(function() {
                //此时绑定的事件就使用了命名空间。
                $("button:eq(0)").bind("click.space", function() {
                    alert("你点击了按钮1");
                }).bind("mouseover", function () {
                    alert("鼠标来了")
                })

                $("button:eq(1)").bind("click.space", function() {
                    alert("你点击了按钮2");
                })

                $("button:eq(2)").bind("click", function() {
                    alert("你点击了按钮3");
                })

                $("button:eq(3)").click(function () {
                    //解除这个命名空间下的所有事件：按钮3的click事件和按钮1mouseover事件没有命名空间，所以不会解绑
                    $("button").unbind(".space")
                })
            })
        </script>
    </head>
    <body>
        <ul>
            <li>
                <button>第1个按钮</button>
            </li>
            <li> <button>第2个按钮</button>
            </li>
            <li><button>第3个按钮</button></li>
            <li><button>解绑space命名空间内的所有事件</button></li>
        </ul>
    </body>
</html>
```

5.3.3 事件委托