jQuery特殊属性操作

val方法

> val方法用于设置和获取表单元素的值，例如input、textarea的值

```
//设置值
$("#name").val(“张三”);
//获取值
$("#name").val();
```

> val方法

```
<body>
  <input type="button" value="呵呵" id="btn">
  <input type="text" value="洋酒" id="txt">
  <script src="jquery-1.12.4.js"></script>
  <script>
    $(function () {
      //console.log($("#btn").val());
      //$("#btn").val("哈哈");
      //console.log($("#btn").attr("value"));
      //$("#txt").val("123");
      $("#txt").focus(function () {
        //如果是默认值，清空内容
        if ($(this).val() === "洋酒") {
          $(this).val("");
        }
      });
      $("#txt").blur(function () {
        if ($(this).val() === "") {
          $(this).val("洋酒");
        }
      });
    });
  </script>
</body>
```

html方法与text方法

> html方法相当于innerHTML text方法相当于innerText

```
//设置内容
$(“div”).html(“<span>这是一段内容</span>”);
//获取内容
$(“div”).html()

//设置内容
$(“div”).text(“<span>这是一段内容</span>”);
//获取内容
$(“div”).text()
```

区别：html方法会识别html标签，text方法会将内容直接当成字符串，并不会识别html标签。

> html方法与text方法

```
<body>
<div><h3>我是标题</h3></div>
<script src="jquery-1.12.4.js"></script>
<script>
  $(function () {
    //html:innerHTML  text:innerText
    //console.log($("div").html());//<h3>我是标题</h3>
    //console.log($("div").text());//我是标题
    $("div").html("<p>我是文本</p>");
    $("div").text("<p>我是文本</p>");
  });
</script>
```

width方法与height方法

> 设置或者获取高度

```
//带参数表示设置高度
$(“img”).height(200);
//不带参数获取高度
$(“img”).height();
```

获取网页的可视区宽高

```
//获取可视区宽度
$(window).width();
//获取可视区高度
$(window).height();
```

width方法与height方法

```
<body>
  <div></div>

  <script src="jquery-1.12.4.js"></script>
```

```
  <script>
    $(function () {
      //获取div的宽度
      //console.log($("div").css("width"));
      //$("div").css("width", "400px");
      //直接获取到的是数字
      //就是获取的width的值
      console.log($("div").width());//width
      //console.log($("div").innerWidth());//padding+width
      //console.log($("div").outerWidth());//padding+width+border
      //console.log($("div").outerWidth(true));//padding+width+border+margin
      //$("div").width(400);
      //需要获取页面可视区的宽度和高度
      //    $(window).resize(function () {
      //      console.log($(window).width());
      //      console.log($(window).height());
      //    });
    });
  </script>
</body>
```

scrollTop与scrollLeft

> 设置或者获取垂直滚动条的位置

```
//获取页面被卷曲的高度
$(window).scrollTop();
//获取页面被卷曲的宽度
$(window).scrollLeft();
```

scrollTop与scrollLeft

```
<style>
    body {
      height: 4000px;
      width: 4000px;
    }
  </style>
</head>
<body>
  <script src="jquery-1.12.4.js"></script>
  <script>
    $(function () {
      $(window).scroll(function () {
        console.log($(window).scrollTop());
        console.log($(window).scrollLeft());
      });
    });
  </script>
</body>
```

> 小火箭返回顶部案例

```
<style>
    body {
      height: 8000px;
    }
    a {
      color: #FFF;
    }
    .actGotop {
      position: fixed;
      bottom: 50px;
      right: 50px;
      width: 150px;
      height: 195px;
      display: none;
      z-index: 100;
    }
    .actGotop a,
    .actGotop a:link {
      width: 150px;
      height: 195px;
      display: inline-block;
      background: url(images/gotop.png) no-repeat;
      outline: none;
    }
    .actGotop a:hover {
      width: 150px;
      height: 195px;
      background: url(images/gotop.gif) no-repeat;
      outline: none;
    }
  </style>
</head>
<body>
  <!-- 返回顶部小火箭 -->
  <div class="actGotop"><a href="javascript:;" title="Top"></a></div>
  <script src="jquery-1.12.4.js"></script>
  <script>
    $(function () {
      //当页面超出去1000px的时候，让小火箭显示出来,如果小于1000，就让小火箭隐藏
      $(window).scroll(function () {
        if ($(window).scrollTop() >= 1000) {
          $(".actGotop").stop().fadeIn(1000);
        } else {
          $(".actGotop").stop().fadeOut(1000);
        }
      });
      function getScroll() {
        return {

          left: window.pageYOffset || document.documentElement.scrollLeft ||
```

```
document.body.scrollLeft || 0,
          top: window.pageYOffset || document.documentElement.scrollTop ||
document.body.scrollTop || 0
        }
      }
      //在外面注册
      $(".actGotop").click(function () {
        $("html,body").stop().animate({ scrollTop: 0 }, 3000);
      })
    });
  </script>
</body>
```

offset方法与position方法

> offset方法获取元素距离document的位置，position方法获取的是元素距离有定位的父元素的位置。

```
//获取元素距离document的位置,返回值为对象：{left:100, top:100}
$(selector).offset();
//获取相对于其最近的有定位的父元素的位置。
$(selector).position();
```

offset方法与position

```
 <style>
    * {
      margin: 0;
      padding: 0;
    }
    .father {
      width: 400px;
      height: 400px;
      background-color: pink;
      position: relative;
      margin: 100px;
    }
    .son {
      width: 200px;
      height: 200px;
      background-color: red;
      position: absolute;
      top: 100px;
      left: 100px;
    }
  </style>
</head>
<body>

  <div class="father">
```

```
    <div class="son"></div>
  </div>
  <script src="jquery-1.12.4.js"></script>
  <script>
    $(function () {
      //获取元素的相对于document的位置
      console.log($(".son").offset());
      //获取元素相对于有定位的父元素的位置
      console.log($(".son").position());
    });
  </script>
</body>
```

jQuery事件机制

> JavaScript中已经学习过了事件，但是jQuery对JavaScript事件进行了封装，增加并扩展了事件处理机制。jQuery不仅提供了更加优雅的事件处理语法，而且极大的增强了事件的处理能力。

jQuery事件发展历程(了解)

简单事件绑定>>bind事件绑定>>delegate事件绑定>>on事件绑定(推荐)

> 简单事件注册

```
click(handler)          单击事件
mouseenter(handler)     鼠标进入事件
mouseleave(handler)     鼠标离开事件
```

缺点：不能同时注册多个事件

> bind方式注册事件

```
//第一个参数：事件类型
//第二个参数：事件处理程序
$("p").bind("click mouseenter", function(){
    //事件响应方法
});
```

缺点：不支持动态事件绑定

> delegate注册委托事件

```
// 第一个参数：selector，要绑定事件的元素
// 第二个参数：事件类型
// 第三个参数：事件处理函数
$(".parentBox").delegate("p", "click", function(){
    //为 .parentBox下面的所有的p标签绑定事件
});
```

缺点：只能注册委托事件，因此注册时间需要记得方法太多了

> on注册事件

> jquery事件机制的发展历程

```
  <style>
    #box {
      width: 500px;
      height: 500px;
      background-color: pink;
    }
  </style>
</head>
<body>
<!--点击按钮，在div里面创建一个新的p元素-->
<input type="button" value="按钮" id="btn">
<div id="box">
  <div>
    <span>呵呵</span>
    <p>11111</p>
    <p>22222</p>
    <p>33333</p>
    <p>44444</p>
  </div>
</div>
<script src="jquery-1.12.4.js"></script>
<script>
  //$("div").children("p").click(function(){})
  $(function () {
//
    //bind方式
//    $("p").bind({
//      click:function () {
//        alert("呵呵")
//      },
//      mouseenter:function () {
//        alert("哈哈")
//      }
//    });
    $("#btn").click(function () {
      $("<p>我是新增加的p元素</p>").appendTo("div");
    });
    //简单事件，给自己注册的事件
//    $("div").click(function () {
//      alert("哈哈");
//    });
    //delegate:代理，委托
    //1. 给父元素注册委托事件，最终还是有子元素来执行。
    //要给div注册一个委托事件,但是最终不是由执行，而是有p执行
    //第一个参数：selector:事件最终由谁来执行。
    //第二个参数：事件的类型
    //第三个参数：函数，要做什么
```

```
    //1. 动态创建的也能有事件 :缺点：只能注册委托事件
    $("#box").delegate("p", "click", function () {
      //alert("呵呵");
      console.log(this);
    });
    //注册简单事件，缺点：一次只能注册一个事件
//    $("p").click(function () {
//      alert("呵呵");
//    });
  });
</script>
</body>
```

on注册事件(重点)

> jQuery1.7之后，jQuery用on统一了所有事件的处理方法。
>
> 最现代的方式，强烈建议使用。

on注册简单事件

```
// 表示给$(selector)绑定事件，并且由自己触发，不支持动态绑定。
$(selector).on( "click", function() {});
```

on注册委托事件

```
// 表示给$(selector)绑定代理事件，当必须是它的内部元素span才能触发这个事件，支持动态绑定
$(selector).on( "click",“span”, function() {});
```

on注册事件的语法：

```
// 第一个参数：events，绑定事件的名称可以是由空格分隔的多个事件（标准事件或者自定义事件）
// 第二个参数：selector，执行事件的后代元素（可选），如果没有后代元素，那么事件将有自己执行。
// 第三个参数：data，传递给处理函数的数据，事件触发的时候通过event.data来使用（不常使用）
// 第四个参数：handler，事件处理函数
$(selector).on(events[,selector][,data],handler);
```

> on注册事件的两种方式

```
<body>
<input type="button" value="增加" id="btn">
<div>
  <p>111111</p>
  <p>111111</p>
  <p>111111</p>
  <p>111111</p>
```

```
</div>
<script src="jquery-1.12.4.js"></script>
<script>
  $(function () {
    // 这个是p自己注册的事件（简单事件）
    /*$("p").on("click", function () {
      alert("呵呵");
    });*/
    $("div").on("click", "p", function () {
      alert("呵呵")
    });//事件委托
    $("#btn").on("click", function () {
      $("<p>我是新建的p元素</p>").appendTo("div");
    });
  });
</script>
</body>
```

事件的执行顺序

```
  <style>
    div {
      height: 500px;
      width: 500px;
      background-color: pink;
    }
  </style>
</head>
<body>
  <input type="button" value="增加" id="btn">
  <div>
    <p>这是自带的p元素</p>
    <p>这是自带的p元素</p>
    <p>这是自带的p元素</p>
    <p>这是自带的p元素</p>
  </div>
  <script src="jquery-1.12.4.js"></script>
  <script>
    $(function () {
      //给div里面的p执行 委托事件
      $("div").on("click", "p", function () {
        alert("333")
      });
      // 这个是p自己注册的事件（简单事件）
      $("p").on("click", function () {
        alert("111");
      });
      //给div自己执行的
      $("div").on("click", function () {
        alert("222");
      });
      $("#btn").on("click", function () {
        $("<p>我是新建的p元素</p>").appendTo("div");
```

```
      });
    });
  </script>
</body>
```

> 表格删除功能

```
  <style>
        * {
            padding: 0;
            margin: 0;
        }
        .wrap {
            width: 410px;
            margin: 100px auto 0;
        }
        table {
            border-collapse: collapse;
            border-spacing: 0;
            border: 1px solid #c0c0c0;
        }
        th,
        td {
            border: 1px solid #d0d0d0;
            color: #404060;
            padding: 10px;
        }
        th {
            background-color: #09c;
            font: bold 16px "Î¢ÈíÑÅºÚ";
            color: #fff;
        }
        td {
            font: 14px "Î¢ÈíÑÅºÚ";
        }
        td a.get {
            text-decoration: none;
        }
        a.del:hover {
            text-decoration: underline;
        }
        tbody tr {
            background-color: #f0f0f0;
        }
        tbody tr:hover {
            cursor: pointer;
            background-color: #fafafa;
        }
        .btnAdd {
            width: 110px;
            height: 30px;
            font-size: 20px;
            font-weight: bold;
```

```
        }
        .form-item {
            height: 100%;
            position: relative;
            padding-left: 100px;
            padding-right: 20px;
            margin-bottom: 34px;
            line-height: 36px;
        }
        .form-item>.lb {
            position: absolute;
            left: 0;
            top: 0;
            display: block;
            width: 100px;
            text-align: right;
        }
        .form-item>.txt {
            width: 300px;
            height: 32px;
        }
        .mask {
            position: absolute;
            top: 0px;
            left: 0px;
            width: 100%;
            height: 100%;
            background: #000;
            opacity: 0.15;
            display: none;
        }
        .form-add {
            position: fixed;
            top: 30%;
            left: 50%;
            margin-left: -197px;
            padding-bottom: 20px;
            background: #fff;
            display: none;
        }
        .form-add-title {
            background-color: #f7f7f7;
            border-width: 1px 1px 0 1px;
            border-bottom: 0;
            margin-bottom: 15px;
            position: relative;
        }
        .form-add-title span {
            width: auto;
            height: 18px;
            font-size: 16px;
            font-family: ËÎÌå;

            font-weight: bold;
```

```
            color: rgb(102, 102, 102);
            text-indent: 12px;
            padding: 8px 0px 10px;
            margin-right: 10px;
            display: block;
            overflow: hidden;
            text-align: left;
        }
        .form-add-title div {
            width: 16px;
            height: 20px;
            position: absolute;
            right: 10px;
            top: 6px;
            font-size: 30px;
            line-height: 16px;
            cursor: pointer;
        }
        .form-submit {
            text-align: center;
        }
        .form-submit input {
            width: 170px;
            height: 32px;
        }
    </style>
</head>
<body>
    <div class="wrap">
        <input type="button" value="清空内容" id="btn">
        <input type="button" value="添加" id="btnAdd">
        <table>
            <thead>
                <tr>
                    <th>课程名称</th>
                    <th>所属学院</th>
                    <th>操作</th>
                </tr>
            </thead>
            <tbody id="j_tb">
                <tr>
                    <!-- <td><input type="checkbox"/></td> -->
                    <td>JavaScript</td>
                    <td>美国</td>
                    <td><a href="javascript:;" class="get">删除</a></td>
                </tr>
                <tr>
                    <!-- <td><input type="checkbox"/></td> -->
                    <td>css</td>
                    <td>英国</td>
                    <td><a href="javascript:;" class="get">删除</a></td>
                </tr>

                <tr>
```

```
                    <!-- <td><input type="checkbox"/></td> -->
                    <td>html</td>
                    <td>中国</td>
                    <td><a href="javascript:;" class="get">删除</a></td>
                </tr>
                <tr>
                    <td>jQuery</td>
                    <td>日本</td>
                    <td><a href="javascript:;" class="get">删除</a></td>
                </tr>
            </tbody>
        </table>
    </div>
    <script src="jquery-1.12.4.js"></script>
    <script>
        $(function() {
            //1. 找到清空按钮，注册点击事件，清空tbody
            $("#btn").on("click", function() {
                $("#j_tb").empty();
            });
            //2. 找到delete，注册点击事件
            //    $(".get").on("click", function () {
            //      $(this).parent().parent().remove();
            //    });
            $("#j_tb").on("click", ".get", function() {
                $(this).parent().parent().remove();
            });
            //3. 找到添加按钮,注册点击事件
            $("#btnAdd").on("click", function() {
                $('<tr> <td>jQuery</td> <td>韩国</td> <td><a href="javascript:;" class="get">删除
</a></td> </tr>').appendTo("#j_tb");
            });

        });
    </script>
</body>
```

事件解绑

> unbind方式（不用）

```
$(selector).unbind(); //解绑所有的事件
$(selector).unbind("click"); //解绑指定的事件
```

> undelegate方式（不用）

```
$( selector ).undelegate(); //解绑所有的delegate事件
$( selector).undelegate( “click” ); //解绑所有的click事件
```

> off方式（推荐）

```
// 解绑匹配元素的所有事件
$(selector).off();
// 解绑匹配元素的所有click事件
$(selector).off("click");
```

> 移除事件绑定

```
<body>
  <input type="button" value="触发" id="btn">
  <p>我是一个p元素</p>
  <script src="jquery-1.12.4.js"></script>
  <script>
    $(function () {
      $("p").on("click", function () {
        alert("呵呵");
      })
      // toggle：切换  trigger：触发
      $("#btn").on("click", function () {
        //触发p元素的点击事件
        //$("p").click();
        //$("p").trigger("click");
      });
    });
  </script>
</body>
```

触发事件

```
$(selector).click(); //触发 click事件
$(selector).trigger("click");
```

jQuery事件对象

jQuery事件对象其实就是js事件对象的一个封装，处理了兼容性。

```
//screenX和screenY    对应屏幕最左上角的值
//clientX和clientY    距离页面左上角的位置（忽视滚动条）
//pageX和pageY    距离页面最顶部的左上角的位置（会计算滚动条的距离）

//event.keyCode 按下的键盘代码
//event.data     存储绑定事件时传递的附加数据

//event.stopPropagation()    阻止事件冒泡行为
//event.preventDefault()     阻止浏览器默认行为
//return false 既能阻止事件冒泡，又能阻止浏览器默认行为。
```

> 事件对象

```
<body>
  <div>这是一个div</div>
  <p>这是一个p元素</p>
  <script src="jquery-1.12.4.js"></script>
  <script>
    $(function () {
      //100，注册的时候的时候，把100传到事件里面去。
      var money = 100;
      //on(types, selector, data, callback)
      //使用on方法的时候，可以给data参数传一个值，可以在事件里面通过e.data获取到。
      $("div").on("click", 100, function (e) {
        console.log(e);
        console.log("哈哈，我有" + e.data);
      });
      money = 0;
      $("p").on("click", function () {
        console.log("呜呜，我有" + 0);
      })
    });
  </script>
</body>
```

阻止冒泡和阻止浏览器的默认行为

```
<body>
    <a href="http://www.baidu.com" id="link">呵呵</a>
    <script src="jquery-1.12.4.js"></script>
    <script>
        $(function() {
            $("#link").on("click", function(e) {
                //阻止 默认
                //e.preventDefault();
                //e.stopPropagation();
                //alert("呵呵");
                //return false;//既能阻止事件冒泡，也能阻止浏览器的默认行为。
                //console.log(e.cancelBubble);
                //alert("呵呵");
            });
            $("body").on("click", function() {
                alert("嘻嘻");
            })
        });
    </script>
</body>
```

delay的用法

```
  <style>
    div {
      width: 400px;
```

```
      height: 60px;
      background-color: pink;
      text-align: center;
      line-height: 60px;
      font-size: 30px;
      /* margin: 100px auto; */
      display: none;

    }
  </style>
</head>
<body>
  <div>这个一个提示消息</div>
  <script src="jquery-1.12.4.js"></script>
  <script>
    $(function () {
      // delay(duration);  ==> 推迟动画队列中的动画

      $("div").fadeIn(1000).delay(2000).fadeOut(1000).slideDown(3000);
      // $("div").fadeIn(1000).delay(2000).css("fontSize", 10); // 只能用于动画队列中的动画
    });
  </script>
</body>
```

> jQuery对象拷贝

```
  <script>
        $(function() {
            // var targetObj = {};
            // var obj = {
            //     id: 1,
            //     name: "andy"
            // };
            // // $.extend(target, obj);
            // $.extend(targetObj, obj);
            // console.log(targetObj);
            // var targetObj = {
            //     id: 0
            // };
            // var obj = {
            //     id: 1,
            //     name: "andy"
            // };
            // // $.extend(target, obj);
            // $.extend(targetObj, obj);
            // console.log(targetObj); // 会覆盖targetObj 里面原来的数据
            var targetObj = {
                id: 0,
                msg: {
                    sex: '男'
                }
            };
            var obj = {
```

```
                id: 1,
                name: "andy",
                msg: {
                    age: 18
                }
            };
            // // $.extend(target, obj);
            // $.extend(targetObj, obj);
            // console.log(targetObj); // 会覆盖targetObj 里面原来的数据
            // // 1. 浅拷贝把原来对象里面的复杂数据类型地址拷贝给目标对象
            // targetObj.msg.age = 20;
            // console.log(targetObj);
            // console.log(obj);
            // 2. 深拷贝把里面的数据完全复制一份给目标对象 如果里面有不冲突的属性,会合并到一起
            $.extend(true, targetObj, obj);
            // console.log(targetObj); // 会覆盖targetObj 里面原来的数据
            targetObj.msg.age = 20;
            console.log(targetObj); // msg :{sex: "男", age: 20}
            console.log(obj);
        })
    </script>
```

隐式迭代

基本概念

> 隐式迭代：jQuery在设置属性时会自动的遍历，因此我们不需要再遍历

1. jQuery在执行设置性操作时，会给所有的元素都设置上相同的值。
2. jQuery在执行获取性操作时，只会返回第一个元素对应的值。
3. 如果想要给每一个元素都设置不同的值，需要手动进行遍历jQuery对象。

each方法

> 遍历jQuery对象集合，为每个匹配的元素执行一个函数

语法：

```
// 参数一表示当前元素在所有匹配元素中的索引号
// 参数二表示当前元素，在function中this也表示当前元素。
$(selector).each(function(index,element){});
```

each方法

```
<style>
    ul {
      list-style: none;
    }

    li {
```

```
      float: left;
      width: 200px;
      height: 200px;
      background: pink;
      text-align: center;
      line-height: 200px;
      margin: 0 20px 20px 0;
    }
  </style>

  <script src="jquery-1.12.4.js"></script>
  <script>
    $(function () {

      // for (var i = 0; i < $("li").length; i++) {
      //   $("li").eq(i).css("opacity", (i + 1) / 10);
      // }

      //each方法
      $("li").each(function (index, element) {
        $(element).css("opacity", (index + 1) / 10);
      })


    });
  </script>

</head>

<body>
  <ul id="ulList">
    <li>什么都看不见</li>
    <li>什么都看不见</li>
    <li>什么都看不见</li>
    <li>什么都看不见</li>
    <li>什么都看不见</li>
    <li>什么都看不见</li>
    <li>什么都看不见</li>
    <li>什么都看不见</li>
    <li>什么都看不见</li>
    <li>什么都看不见</li>
  </ul>
</body>
```

链式编程

> 链式编程的原理：设置性操作会返回一个jQuery对象，因此可以继续调用jQuery的方法。

1. 设置操作的时候，可以使用链式编程。
2. 获取操作的时候，无法使用链式编程。

```
end(); // 上一次返回的jq对象
```

> 链式编程

```
<body>
  <div></div>
  <script src="jquery-1.12.4.js"></script>
  <script>
    $(function () {
      //设置性操作：可以链式编程
      //获取性操作，不能链式，因为获取性操作，数值，字符串，
      //返回值是不是一个jq对象。
      console.log($("div").width(200).height(200).css("backgroundColor", "pink").width());
    });
  </script>
</body>
```

> 五星评分案例

```
<style>
    * {
      padding: 0;
      margin: 0;
    }

    .comment {
      font-size: 40px;
      color: #ff16cf;
    }

    .comment li {
      float: left;
    }

    ul {
      list-style: none;
    }
  </style>
  <script src="jquery-1.12.4.js"></script>
  <script>
    $(function () {
      //1. 给li注册鼠标经过事件，让自己和前面所有的兄弟都变成实心
      var wjx_k = "☆";
      var wjx_s = "★";
      $(".comment>li").on("mouseenter", function () {
        $(this).text(wjx_s).prevAll().text(wjx_s);
        $(this).nextAll().text(wjx_k);
      });
      //2. 给ul注册鼠标离开时间，让所有的li都变成空心
      $(".comment").on("mouseleave", function () {
        $(this).children().text(wjx_k);
        //再做一件事件，找到current，让current和current前面的变成实心就行。
```

```
        $("li.current").text(wjx_s).prevAll().text(wjx_s);
      });
      //3. 给li注册点击事件
      $(".comment>li").on("click", function () {
        $(this).addClass("current").siblings().removeClass("current");
      });
    });
  </script>
</head>
<body>
  <ul class="comment">
    <li>☆</li>
    <li>☆</li>
    <li>☆</li>
    <li>☆</li>
    <li>☆</li>
  </ul>
</body>
```

```
prevAll();//获取前面所有的兄弟元素
nextAll();//获取后面所有的兄弟元素
siblings();//获取所有的兄弟元素
prev();//获取前一个兄弟
next();//获取后一个兄弟。
```

多库共存

> jQuery使用`$`作为标示符，但是如果与其他框架中的冲突时，*jQuery*可以释放``符的控制权.

```
var c = $.noConflict();//释放$的控制权,并且把$的能力给了c
```

```
    <script>
        $(function() {
            function $(ele) {
                return document.querySelector(ele);
            }
            //console.log($("div"));报错，符号冲突
            $.each();
            // 1. 如果$ 符号冲突 我们就使用 jQuery
            jQuery.each();
            // 2. 让jquery 释放对$ 控制权 让用自己决定
            var suibian = jQuery.noConflict();
            console.log(suibian("span"));
            suibian.each();
        })
    </script>
</head>

<body>
```

```
    <div></div>
    <span></span>
</body>
```