**day02 - jQuery**

学习目标：

能够操作 jQuery 属性

能够操作 jQuery 元素

能够操作 jQuery 元素尺寸、位置

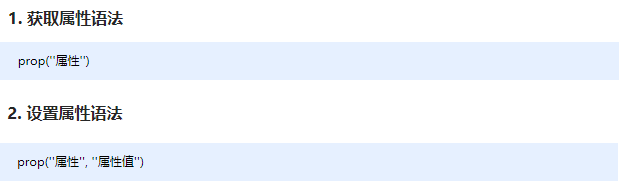
**1.1. jQuery 属性操作**

jQuery 常用属性操作有三种：prop() / attr() / data() ;

**1.1.1 元素固有属性值 prop()**

所谓元素固有属性就是元素本身自带的属性，比如 <a> 元素里面的 href ，比如 <input> 元素里面的 type。

**语法**



注意：prop() 除了普通属性操作，更适合操作表单属性：disabled / checked / selected 等。

当元素的值改变时发生 change 事件（仅适用于表单字段）

<a href="http://www.itcast.cn" title="都挺好">都挺好</a>

<input type="checkbox" name="" id="" checked>

//1. element.prop("属性名") 获取元素固有的属性值

            console.log($("a").prop("href"));

            $("a").prop("title", "我们都挺好");

            $("input").change(function() {

                console.log($(this).prop("checked"));

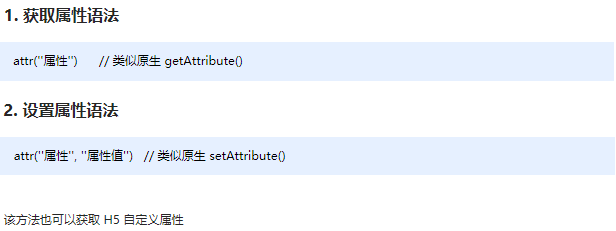
            });

            // console.log($("div").prop("index"));

**1.1.2 元素自定义属性值 attr()**

用户自己给元素添加的属性，我们称为自定义属性。 比如给 div 添加 index =“1”。

**语法**



注意：attr() 除了普通属性操作，更适合操作自定义属性。（该方法也可以获取 H5 自定义属性）

<div index="1" data-index="2">我是div</div>

// 2. 元素的自定义属性 我们通过 attr()

            console.log($("div").attr("index"));

            $("div").attr("index", 4);

            console.log($("div").attr("data-index"));

**1.1.3 数据缓存 data()**

data() 方法可以在指定的元素上存取数据，并不会修改 DOM 元素结构。一旦页面刷新，之前存放的数据都将被移除。

**语法**

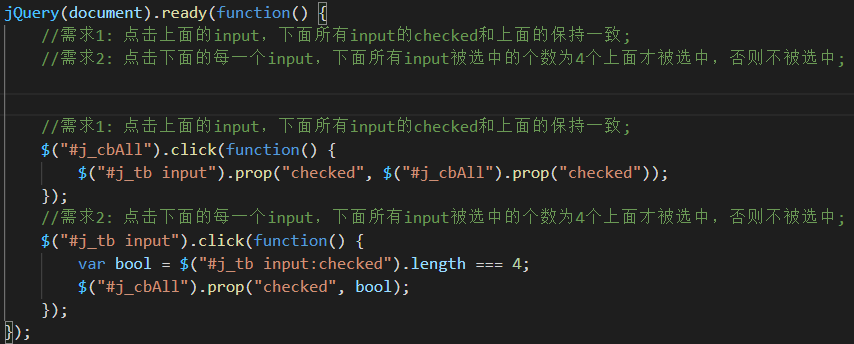


注意：同时，还可以读取 HTML5 自定义属性 data-index ，得到的是数字型。

**演示代码**

<body>  
   <a href="http://www.itcast.cn" title="都挺好">都挺好</a>  
   <input type="checkbox" name="" id="" checked>  
   <div index="1" data-index="2">我是div</div>  
   <span>123</span>  
   <script>  
       $(function() {  
           //1. element.prop("属性名") 获取元素固有的属性值  
           console.log($("a").prop("href"));  
           $("a").prop("title", "我们都挺好");  
           $("input").change(function() {  
               console.log($(this).prop("checked"));  
          });  
           // console.log($("div").prop("index"));  
           // 2. 元素的自定义属性 我们通过 attr()  
           console.log($("div").attr("index"));  
           $("div").attr("index", 4);  
           console.log($("div").attr("data-index"));  
           // 3. 数据缓存 data() 这个里面的数据是存放在元素的内存里面  
           $("span").data("uname", "andy");  
           console.log($("span").data("uname"));  
           // 这个方法获取data-index h5自定义属性 第一个 不用写data- 而且返回的是数字型  
           console.log($("div").data("index"));  
      })  
   </script>  
</body>

**练习1：**



**1.1.4 案例：购物车案例模块-全选**

1.全选思路：里面3个小的复选框按钮（j-checkbox）选中状态（checked）跟着全选按钮（checkall）走。

2.因为checked 是复选框的固有属性，此时我们需要利用prop()方法获取和设置该属性。

3.把全选按钮状态赋值给3小复选框就可以了。

4.当我们每次点击小的复选框按钮，就来判断：

5.如果小复选框被选中的个数等于3 就应该把全选按钮选上，否则全选按钮不选。

6.checked 选择器 :checked 查找被选中的表单元素。

$(function() {

    // 1. 全选 全不选功能模块

    // 就是把全选按钮（checkall）的状态赋值给 三个小的按钮（j-checkbox）就可以了

    // 事件可以使用change

    $(".checkall").change(function() {

        // console.log($(this).prop("checked"));

        $(".j-checkbox, .checkall").prop("checked", $(this).prop("checked"));

});

// 2. 如果小复选框被选中的个数等于3 就应该把全选按钮选上，否则全选按钮不选。

    $(".j-checkbox").change(function() {

        // if(被选中的小的复选框的个数 === 3) {

        //     就要选中全选按钮

        // } else {

        //     不要选中全选按钮

        // }

        // console.log($(".j-checkbox:checked").length);

        // $(".j-checkbox").length 这个是所有的小复选框的个数

        if ($(".j-checkbox:checked").length === $(".j-checkbox").length) {

            $(".checkall").prop("checked", true);

        } else {

            $(".checkall").prop("checked", false);

        }

    });

}

**1.2. jQuery 文本属性值**

jQuery的文本属性值常见操作有三种：html() / text() / val() ; 分别对应JS中的 innerHTML 、innerText 和 value 属性。

**1.2.1 jQuery内容文本值**

常见操作有三种：html() / text() / val() ; 分别对应JS中的 innerHTML 、innerText 和 value 属性，主要针对元素的内容还有表单的值操作。

**语法**



注意：html() 可识别标签，text() 不识别标签。

**演示代码**

<body>  
   <div>  
       <span>我是内容</span>  
   </div>  
   <input type="text" value="请输入内容">  
   <script>  
       // 1. 获取设置元素内容 html()  
       console.log($("div").html());  
       // $("div").html("123");  
       // 2. 获取设置元素文本内容 text()  
       console.log($("div").text());  
       $("div").text("123");  
       // 3. 获取设置表单值 val()  
       console.log($("input").val());  
       $("input").val("123");  
   </script>  
</body>

**1.2.2. 案例：购物车案例模块-增减商品数量**

1.核心思路：首先声明一个变量，当我们点击+号（increment），就让这个值++，然后赋值给文本框。

2.注意1： 只能增加本商品的数量， 就是当前+号的兄弟文本框（itxt）的值。 3.修改表单的值是val() 方法

4.注意2： 这个变量初始值应该是这个文本框的值，在这个值的基础上++。要获取表单的值

5.减号（decrement）思路同理，但是如果文本框的值是1，就不能再减了。

// 3. 增减商品数量模块 首先声明一个变量，当我们点击+号（increment），就让这个值++，然后赋值给文本框。

//增加商品数量

    $(".increment").click(function() {

        // 得到当前兄弟文本框的值

        var n = $(this).siblings(".itxt").val();

        // console.log(n);

        n++;

        $(this).siblings(".itxt").val(n);

});

//减少商品数量

    $(".decrement").click(function() {

        // 得到当前兄弟文本框的值

        var n = $(this).siblings(".itxt").val();

        if (n == 1) {

            return false;

        }

        // console.log(n);

        n--;

        $(this).siblings(".itxt").val(n);

    });

**1.2.3. 案例：购物车案例模块-修改商品小计**

1.核心思路：每次点击+号或者-号，根据文本框的值 乘以 当前商品的价格 就是 商品的小计

2.注意1： 只能增加本商品的小计， 就是当前商品的小计模块（p-sum）

3.修改普通元素的内容是text() 方法

4.注意2： 当前商品的价格，要把￥符号去掉再相乘 截取字符串 substr(1)

5.parents(‘选择器’) 可以返回指定祖先元素

6.最后计算的结果如果想要保留2位小数 通过 toFixed(2) 方法

7.用户也可以直接修改表单里面的值，同样要计算小计。 用表单change事件

8.用最新的表单内的值 乘以 单价即可 但是还是当前商品小计

// 3. 增减商品数量模块 首先声明一个变量，当我们点击+号（increment），就让这个值++，然后赋值给文本框。

    $(".increment").click(function() {

        // 得到当前兄弟文本框的值

        var n = $(this).siblings(".itxt").val();

        // console.log(n);

        n++;

        $(this).siblings(".itxt").val(n);

        // 3. 计算小计模块 根据文本框的值 乘以 当前商品的价格  就是 商品的小计

        // 当前商品的价格 p

        var p = $(this).parents(".p-num").siblings(".p-price").html();

        // console.log(p);

        p = p.substr(1);

        console.log(p);

        var price = (p \* n).toFixed(2);

        // 小计模块

        // toFixed(2) 可以让我们保留2位小数

        $(this).parents(".p-num").siblings(".p-sum").html("￥" + price);

        getSum();

    });

    $(".decrement").click(function() {

        // 得到当前兄弟文本框的值

        var n = $(this).siblings(".itxt").val();

        if (n == 1) {

            return false;

        }

        // console.log(n);

        n--;

        $(this).siblings(".itxt").val(n);

        // var p = $(this).parent().parent().siblings(".p-price").html();

        // parents(".p-num") 返回指定的祖先元素

        var p = $(this).parents(".p-num").siblings(".p-price").html();

        // console.log(p);

        p = p.substr(1);

        console.log(p);

        // 小计模块

        $(this).parents(".p-num").siblings(".p-sum").html("￥" + (p \* n).toFixed(2));

        getSum();

});

//  4. 用户修改文本框的值 计算 小计模块

    $(".itxt").change(function() {

        // 先得到文本框的里面的值 乘以 当前商品的单价

        var n = $(this).val();

        // 当前商品的单价

        var p = $(this).parents(".p-num").siblings(".p-price").html();

        // console.log(p);

        p = p.substr(1);

        $(this).parents(".p-num").siblings(".p-sum").html("￥" + (p \* n).toFixed(2));

        getSum();

    });

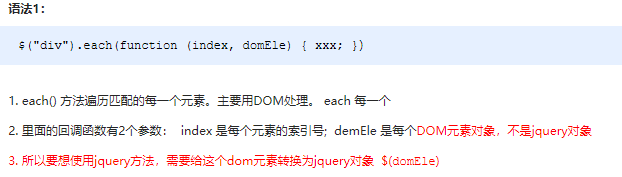
**1.3. jQuery 元素操作**

jQuery 元素操作主要讲的是用jQuery方法，操作标签的遍历、创建、添加、删除等操作。

**1.3.1. 遍历元素**

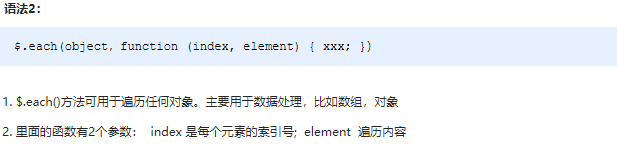
jQuery 隐式迭代是对同一类元素做了同样的操作。 如果想要给同一类元素做不同操作，就需要用到遍历。

**语法1**



注意：此方法用于遍历 jQuery 对象中的每一项，回调函数中元素为 DOM 对象，想要使用 jQuery 方法需要转换。

**语法2**



注意：此方法用于遍历 jQuery 对象中的每一项，回调函数中元素为 DOM 对象，想要使用 jQuery 方法需要转换。

**演示代码**

<body>  
   <div>1</div>  
   <div>2</div>  
   <div>3</div>  
   <script>  
       $(function() {  
           // 如果针对于同一类元素做不同操作，需要用到遍历元素（类似for，但是比for强大）  
           var sum = 0;  
           var arr = ["red", "green", "blue"];  
           // 1. each() 方法遍历元素   
           $("div").each(function(i, domEle) {  
               // 回调函数第一个参数一定是索引号 可以自己指定索引号号名称  
               // console.log(i);  
               // 回调函数第二个参数一定是 dom 元素对象，也是自己命名  
               // console.log(domEle); // 使用jQuery方法需要转换 $(domEle)  
               $(domEle).css("color", arr[i]);  
               sum += parseInt($(domEle).text());  
          })  
           console.log(sum);  
           // 2. $.each() 方法遍历元素 主要用于遍历数据，处理数据  
           // $.each($("div"), function(i, ele) {  
           //     console.log(i);  
           //     console.log(ele);  
           // });  
           // $.each(arr, function(i, ele) {  
           //     console.log(i);  
           //     console.log(ele);  
           // })  
           $.each({  
               name: "andy",  
               age: 18  
          }, function(i, ele) {  
               console.log(i); // 输出的是 name age 属性名  
               console.log(ele); // 输出的是 andy 18 属性值  
          })  
      })  
   </script>  
</body>

**1.3.2. 案例：购物车案例模块-计算总计和总额**

1.把所有文本框中的值相加就是总额数量，总计同理。

2.文本框里面的值不同，如果想要相加需要用 each() 遍历，声明一个变量做计数器，累加即可。

// 5. 计算总计和总额模块

    getSum();

    function getSum() {

        var count = 0; // 计算总件数

        var money = 0; // 计算总价钱

        $(".itxt").each(function(i, ele) {

            count += parseInt($(ele).val());

        });

        $(".amount-sum em").text(count);

        $(".p-sum").each(function(i, ele) {

            money += parseFloat($(ele).text().substr(1));

        });

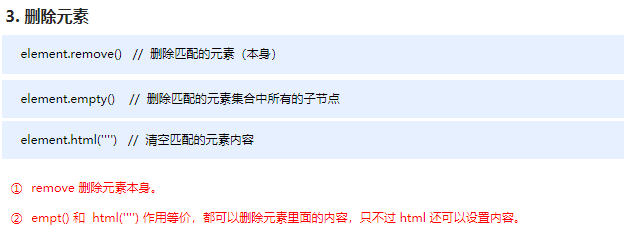
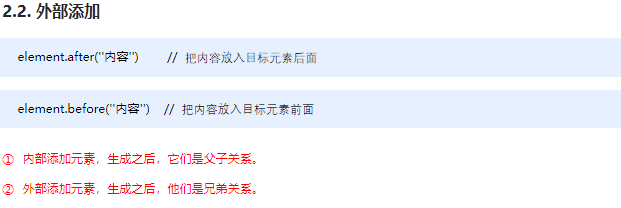
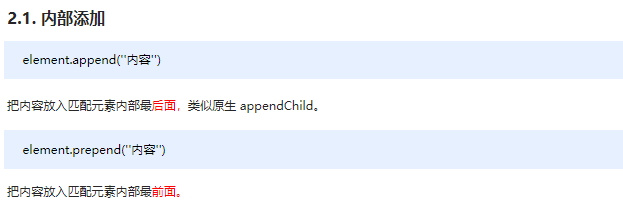
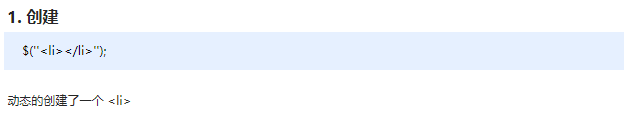
        $(".price-sum em").text("￥" + money.toFixed(2));

    }

**1.3.3. 创建、添加、删除**

jQuery方法操作元素的创建、添加、删除方法很多，则重点使用部分，如下：

**语法总和**



注意：以上只是元素的创建、添加、删除方法的常用方法，其他方法请参详API。

**案例代码**

<body>  
   <ul>  
       <li>原先的li</li>  
   </ul>  
   <div class="test">我是原先的div</div>  
   <script>  
       $(function() {  
           // 1. 创建元素  
           var li = $("<li>我是后来创建的li</li>");  
       
           // 2. 添加元素  
           // 2.1 内部添加  
           // $("ul").append(li); 内部添加并且放到内容的最后面   
           $("ul").prepend(li); // 内部添加并且放到内容的最前面  
           // 2.2 外部添加  
           var div = $("<div>我是后妈生的</div>");  
           // $(".test").after(div);  
           $(".test").before(div);  
       
           // 3. 删除元素  
           // $("ul").remove(); 可以删除匹配的元素 自杀  
           // $("ul").empty(); // 可以删除匹配的元素里面的子节点 孩子  
           $("ul").html(""); // 可以删除匹配的元素里面的子节点 孩子  
      })  
   </script>  
</body>

**练习2：**

jQuery(window).ready(function() {

//需求1：点击前两个按钮，一侧的所有option移动到另一侧的select标签中;

//需求2：点击后两个按钮，一侧的所有option中"被选中"的全部移动到另一侧的select标签中;

/以前我们要一把一把操作需要借助for循环;有了jQuery之后，就不用for循环;

//需求1：点击前两个按钮，一侧的所有option移动到另一侧的select标签中;

            $("button").eq(0).click(function() {

                //左侧所有option全部移动到右侧select中(不用for循环)

                $("#sel2").append($("#sel1>option"));

            });

            $("button").eq(1).click(function() {

                //右侧所有option全部移动到左侧select中(不用for循环)

                $("#sel1").append($("#sel2>option"));

            });

//需求2：点击后两个按钮，一侧的所有option中"被选中"的全部移动到另一侧的select标签中;

            //(:selected)

            $("button").eq(2).click(function() {

                //左侧所有option全部移动到右侧select中(不用for循环)

                $("#sel2").append($("#sel1>option:selected"));

            });

            $("button").eq(3).click(function() {

                //右侧所有option全部移动到左侧select中(不用for循环)

                $("#sel1").append($("#sel2>option:selected"));

            });

            //优化封装;

            //            function fn(str1,str2) {

            //右侧所有option全部移动到左侧select中(不用for循环)

            //                $(str1).append($(str2));

            //            }

        });

**练习3：**

$(function() {

            //需求1：点击j\_btnAddData，显示mask和form表单;

            //需求2：点击j\_hideFormAdd，隐藏mask和form表单;

            //需求3：点击j\_btnAdd，添加一个tr到tbody中;

            //需求4：点击a.get，删除他所在的tr;

            console.log($("#j\_btnAddData"))

                //需求1：点击j\_btnAddData，显示mask和form表单;

            $("#j\_btnAddData").click(showOrHide);

            //需求2：点击j\_hideFormAdd，隐藏mask和form表单;

            $("#j\_hideFormAdd").click(showOrHide);

            //需求3：点击j\_btnAdd，添加一个tr到tbody中;

            $("#j\_btnAdd").click(function() {

                //获取两个input里面的值，往tr里面的td中设置内容;

                var val1 = $("#j\_txtLesson").val(); //val()可以获取input的value属性值;

                var val2 = $("#j\_txtBelSch").val(); //val()可以获取input的value属性值;

                //bug3: 如果两个值为空字符串或者空格，不能添加;

                if (val1.trim() == "" || val2.trim() == "") {

                    alert("内容不能为空！");

                    $("#j\_txtLesson").val(""); //设置参数就是赋值，和html()一样;

                    $("#j\_txtBelSch").val("清华学院"); //设置参数就是赋值，和html()一样;

                    return;

                }

                var tr = $("<tr></tr>");

                var str = "<td>" + val1 + "</td>" +

                    "<td>" + val2 + "</td>" +

                    "<td><a href='javascript:;' class='get'>GET</a></td>";

                tr.html(str);

                //添加到tbody中

                $("#j\_tb").append(tr);

                //bug1: 添加完数据，隐藏两个盒子;

                showOrHide();

                //bug2: 初始化值，第一个清空，第二个默认值;

                $("#j\_txtLesson").val(""); //设置参数就是赋值，和html()一样;

                $("#j\_txtBelSch").val("清华学院"); //设置参数就是赋值，和html()一样;

                //bug4: 新添加的元素没有老事件;

                //每次创建一个，绑定一次

                tr.find(".get").click(function() {

                    //删除他所在的tr;

                    $(this).parent().parent().remove();

                });

            });

            //需求4：点击a.get，删除他所在的tr;

            //新添加的元素没有老事件：1.事件委托;  2.每次创建绑定一次;  3.jQ事件绑定;

            $(".get").click(function() {

                //删除他所在的tr;

                $(this).parent().parent().remove();

            });

            //每天讲解(以事件委托的形式进行事件绑定)

            //            $("#j\_tb").on("click",".get", function () {

            //                $(this).parent().parent().remove();

            //            })

            //封装,显示隐藏盒子

            function showOrHide() {

                $("#j\_mask").toggle();

                $("#j\_formAdd").toggle();

            }

        })

**1.3.4 案例：购物车案例模块-删除商品模块**

1.核心思路：把商品remove() 删除元素即可

2.有三个地方需要删除：

1. 商品后面的删除按钮

2. 删除选中的商品

3. 清理购物车

3.商品后面的删除按钮： 一定是删除当前的商品，所以从 $(this) 出发

4.删除选中的商品： 先判断小的复选框按钮是否选中状态，如果是选中，则删除对应的商品

5.清理购物车： 则是把所有的商品全部删掉

// 6. 删除商品模块

    // (1) 商品后面的删除按钮

    $(".p-action a").click(function() {

        // 删除的是当前的商品

        $(this).parents(".cart-item").remove();

        getSum();

    });

    // (2) 删除选中的商品

    $(".remove-batch").click(function() {

        // 删除的是小的复选框选中的商品

        $(".j-checkbox:checked").parents(".cart-item").remove();

        getSum();

    });

    // (3) 清空购物车 删除全部商品

    $(".clear-all").click(function() {

        $(".cart-item").remove();

        getSum();

    })

**1.3.5 案例：购物车案例模块-选中商品添加背景**

1.核心思路：选中的商品添加背景，不选中移除背景即可

2.全选按钮点击：如果全选是选中的，则所有的商品添加背景，否则移除背景

3.小的复选框点击： 如果是选中状态，则当前商品添加背景，否则移除背景

4.这个背景，可以通过类名修改，添加类和删除类

$(".checkall").change(function() {

        // console.log($(this).prop("checked"));

        $(".j-checkbox, .checkall").prop("checked", $(this).prop("checked"));

//添加背景

        if ($(this).prop("checked")) {

            // 让所有的商品添加 check-cart-item 类名

            $(".cart-item").addClass("check-cart-item");

        } else {

            // check-cart-item 移除

            $(".cart-item").removeClass("check-cart-item");

        }

});

$(".j-checkbox").change(function() {

        // if(被选中的小的复选框的个数 === 3) {

        //     就要选中全选按钮

        // } else {

        //     不要选中全选按钮

        // }

        // console.log($(".j-checkbox:checked").length);

        // $(".j-checkbox").length 这个是所有的小复选框的个数

        if ($(".j-checkbox:checked").length === $(".j-checkbox").length) {

            $(".checkall").prop("checked", true);

        } else {

            $(".checkall").prop("checked", false);

        }

//添加背景

        if ($(this).prop("checked")) {

            // 让当前的商品添加 check-cart-item 类名

            $(this).parents(".cart-item").addClass("check-cart-item");

        } else {

            // check-cart-item 移除

            $(this).parents(".cart-item").removeClass("check-cart-item");

        }

    });

**1.4. jQuery 尺寸、位置操作**

jQuery中分别为我们提供了两套快速获取和设置元素尺寸和位置的API，方便易用，内容如下。

**1.4.1. jQuery 尺寸操作**

jQuery 尺寸操作包括元素宽高的获取和设置，且不一样的API对应不一样的盒子模型。

**语法**



**代码演示**

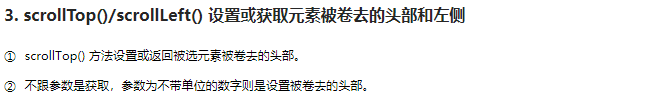
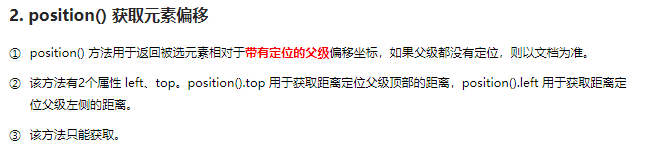
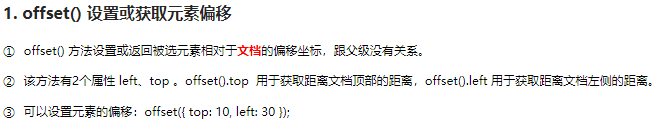
<body>  
   <div></div>  
   <script>  
       $(function() {  
           // 1. width() / height() 获取设置元素 width和height大小   
           console.log($("div").width());  
           // $("div").width(300);  
​  
           // 2. innerWidth() / innerHeight() 获取设置元素 width和height + padding 大小   
           console.log($("div").innerWidth());  
​  
           // 3. outerWidth() / outerHeight() 获取设置元素 width和height + padding + border 大小   
           console.log($("div").outerWidth());  
​  
           // 4. outerWidth(true) / outerHeight(true) 获取设置 width和height + padding + border + margin  
           console.log($("div").outerWidth(true));  
      })  
   </script>  
</body>

注意：有了这套 API 我们将可以快速获取和子的宽高，至于其他属性想要获取和设置，还要使用 css() 等方法配合。

**1.4.2. jQuery 位置操作**

jQuery的位置操作主要有三个： offset()、position()、scrollTop()/scrollLeft() , 具体介绍如下:

**语法**



**代码演示**

.father {

            width: 400px;

            height: 400px;

            background-color: pink;

            margin: 100px;

            overflow: hidden;

            position: relative;

        }

     .son {

            width: 150px;

            height: 150px;

            background-color: purple;

            position: absolute;

            left: 10px;

            top: 10px;

        }

<div class="father">

        <div class="son"></div>

    </div>

    <script>

        $(function() {

            // 1. 获取设置距离文档的位置（偏移） offset

            console.log($(".son").offset());

            console.log($(".son").offset().top);

            // $(".son").offset({

            //     top: 200,

            //     left: 200

            // });

            // 2. 获取距离带有定位父级位置（偏移） position   如果没有带有定位的父级，则以文档为准

            // 这个方法只能获取不能设置偏移

            console.log($(".son").position());

            // $(".son").position({

            //     top: 200,

            //     left: 200

            // });

        })

</script>

**1.4.3. 案例：带有动画的返回顶部**

1.核心原理： 使用animate动画返回顶部。

2.animate动画函数里面有个scrollTop 属性，可以设置位置

3.但是是元素做动画，因此 $(“body,html”).animate({scrollTop: 0})

<style>

        body {

            height: 2000px;

        }

        .back {

            position: fixed;

            width: 50px;

            height: 50px;

            background-color: pink;

            right: 30px;

            bottom: 100px;

            display: none;

        }

        .container {

            width: 900px;

            height: 500px;

            background-color: skyblue;

            margin: 400px auto;

        }

</style>

<div class="back">返回顶部</div>

    <div class="container">

    </div>

    <script>

        $(function() {

            $(document).scrollTop(100);

            // 被卷去的头部 scrollTop()  / 被卷去的左侧 scrollLeft()

            // 页面滚动事件

            var boxTop = $(".container").offset().top;

            $(window).scroll(function() {

                // console.log(11);

                console.log($(document).scrollTop());

                if ($(document).scrollTop() >= boxTop) {

                    $(".back").fadeIn();

                } else {

                    $(".back").fadeOut();

                }

            });

            // 返回顶部

            $(".back").click(function() {

                // $(document).scrollTop(0);

                $("body, html").stop().animate({

                    scrollTop: 0

                });

                // $(document).stop().animate({

                //     scrollTop: 0

                // }); 不能是文档而是 html和body元素做动画

            })

        })

</script>

**1.4.4. 案例： 品优购电梯导航（上）**

1.当我们滚动到 今日推荐 模块，就让电梯导航显示出来

2.点击电梯导航页面可以滚动到相应内容区域

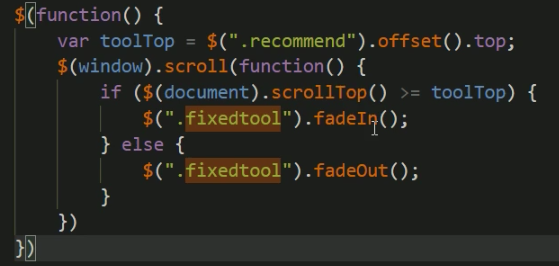
3.核心算法：因为电梯导航模块和内容区模块一一对应的

4.当我们点击电梯导航某个小模块，就可以拿到当前小模块的索引号

5.就可以把animate要移动的距离求出来：当前索引号内容区模块它的offset().top

6.然后执行动画即可

显示：



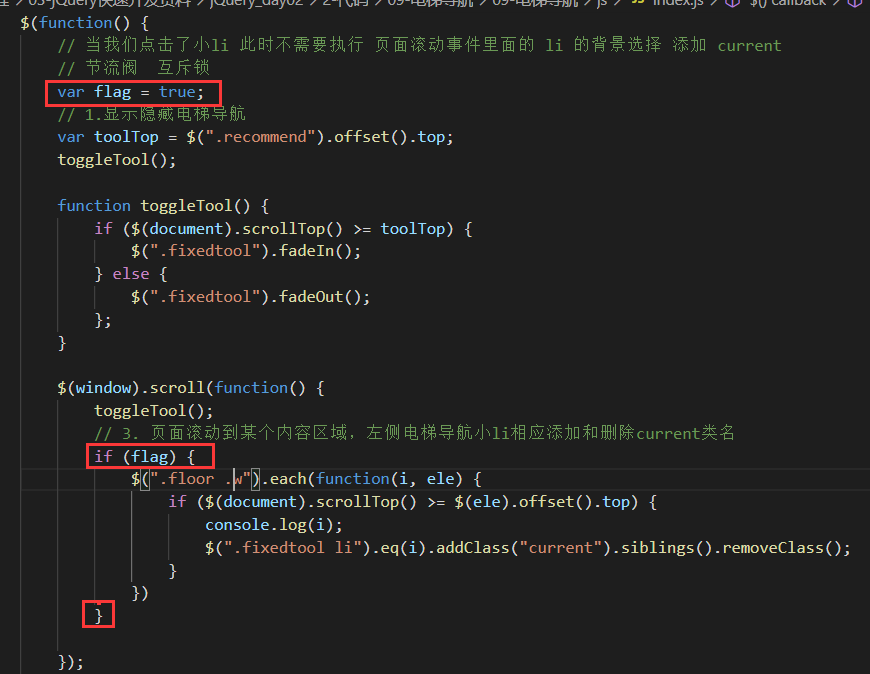
点击：

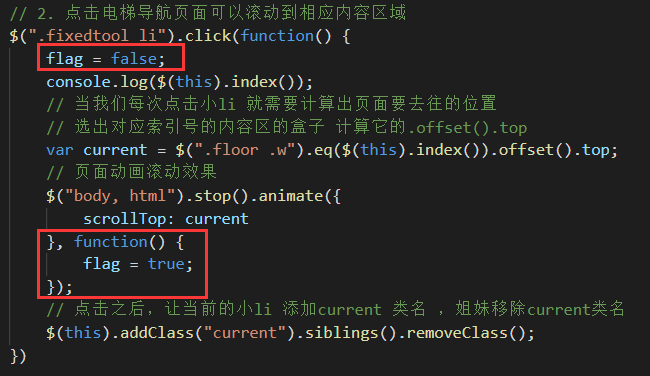


封装显示隐藏，滚动调用：



改bug：





**1.4.5. 案例：品优购电梯导航（下）**

1.当我们点击电梯导航某个小li， 当前小li 添加current类，兄弟移除类名

2.当我们页面滚动到内容区域某个模块， 左侧电梯导航，相对应的小li模块，也会添加current类， 兄弟移除current类。

3.触发的事件是页面滚动，因此这个功能要写到页面滚动事件里面。

4.需要用到each，遍历内容区域大模块。 each里面能拿到内容区域每一个模块元素和索引号

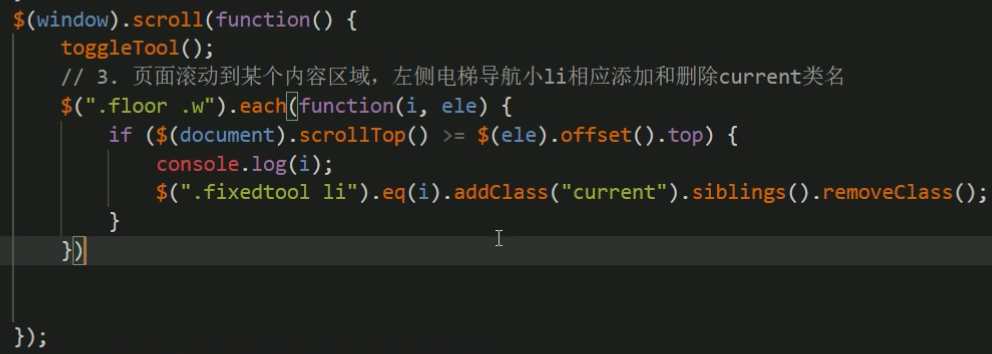
5.判断的条件： 被卷去的头部 大于等于 内容区域里面每个模块的offset().top

6.就利用这个索引号找到相应的电梯导航小li添加类。

点击li类名改变：



滚动类名改变：



**作业：**

$(function() {

            //需求：页面滚动，如果被卷去的头部超过第一个盒子的高，给第二个盒子加固定定位，第三个盒子加上padding;

            $(window).scroll(function() {

                //判断：页面被卷去的顶部的高;

                if ($(window).scrollTop() > $("#top").height()) {

                    //第二个盒子加定位，第三个盒子加padding;

                    var json = {

                        position: 'fixed',

                        top: 0,

                        left: "50%",

                        "margin-left": -$("#Q-nav").width() / 2

                    };

                    $("#Q-nav").css(json);

                    $("#main").css("padding-top", $("#Q-nav").height());

                } else {

                    //第二个盒子去除定位，第三个盒子去除padding;

                    var json = {

                        position: 'static',

                        left: 0,

                        "margin-left": 0,

                        "margin": "0 auto"

                    };

                    $("#Q-nav").css(json);

                    $("#main").css("padding-top", 0);

                }

            });

        });

**1.5. 今日总结**

