Java第二阶段\_day13\_Jquery基础案例

1. 原生js案例。

<!DOCTYPE html>

<html>

<head lang="en">

<meta charset="UTF-8">

<title>标题</title>

<style>

div {

height: 100px;

margin-top: 10px;

}

</style>

</head>

<body>

<input type="button" value="设置边框" id="btn1"/>

<input type="button" value="设置文本" id="btn2"/>

<div></div>

<div></div>

<div></div>

<script>

//需求:点击按钮,给div设置边框和文本

//以前的做法

//入口函数

window.onload = function () {

//获取元素

var btn1 = document.getElementById("btn1");

var btn2 = document.getElementById("btn2");

var divs = document.getElementsByTagName("div");

//设置边框

btn1.onclick = function () {

for(var i = 0 ; i < divs.length; i++){

divs[i].style.border = "1px solid red";

}

}

//设置文本

btn2.onclick = function () {

for(var i = 0 ; i < divs.length; i++){

divs[i].textContent = "设置的文本";

}

}

}

// 再写一个入口函数window.onload

// window.onload = function () {

// console.log("我又是一个入口函数");

// }

//原生js有哪些缺点:

//1.不能写多个入口函数,如果写了多个,后面的会覆盖前面的.

//2.原生js的api名字都太长,太难记

//3.原生js有代码冗余.

//4.原生js中有些方法或者属性,有些浏览器不支持.

//5.容错率较低,前面的代码出现了问题,后面的代码也不会执行.

</script>

</body>

</html>

1. jquery实现

<!DOCTYPE html>

<html>

<head lang="en">

<meta charset="UTF-8">

<title>标题</title>

<style>

div {

height: 100px;

margin-top: 10px;

}

</style>

</head>

<body>

<input type="button" value="设置边框" id="btn1"/>

<input type="button" value="设置文本" id="btn2"/>

<div></div>

<div></div>

<div></div>

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

<script>

//需求:点击按钮,给div设置边框和文本

//jQuery的做法

//入口函数.

$(document).ready(function () {

//设置边框

$('#btn1').click(function () {

$('div').css('border','1px solid red');

});

//设置文本

$('#btn2').click(function () {

$('div').text('设置的文本');

});

});

//再写一个入口函数

// $(document).ready(function () {

// console.log("我又是一个入口函数");

// });

//jQuery的优势

//1.可以写多个入口函数.

//2.jQuery的api都比较好记.

//3.代码简洁. 隐式迭代

//4.jQuery已经帮你处理好了浏览器兼容.

//5.jQuery容错率较高.

</script>

</body>

</html>

1. 如何使用jQuery

<!DOCTYPE html>

<html>

<head lang="en">

<meta charset="UTF-8">

<title>标题</title>

</head>

<body>

<script>

//如何使用jQuery.

//1.导入jQuery这个库.

//2.写一个入口函数.

//3.获取你想要操作的元素,从而操作他(设置边框等样式,设置文本,设置内容....).

//1.可以从官网上下载jQuery文件.

//jQuery有版本

//1.x 不更新 支持老浏览器的.

//2.x 不更新 不支持老浏览器.

//3.x 更新中 不支持老浏览器.

//每一个版本又有压缩版和未压缩版.

//压缩过的 生产环境,实际开发产品

//未压缩的 开发测试.

</script>

</body>

</html>

1. jQuery的入口函数

<!DOCTYPE html>

<html>

<head lang="en">

<meta charset="UTF-8">

<title>标题</title>

</head>

<body>

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

<script>

//1.jQuery的入口函数.

//写法1:

// $(document).ready(function () {

// console.log("这是一个jQuery入口函数");

// });

//写法2:

// $(function () {

// console.log("这又是一个jQuery入口函数的写法");

// });

//2.jQuery的入口函数和原生js的入口函数(window.onload)的区别.

//a.原生js的入口函数不能写多个; jQuery入口函数可以写多个.

//b.执行时机不同:

// 原生js入口函数,要等待页面上所有的资源(外部css/js文件,dom树加载,图片...)都加载完毕后才执行.

// jQuery入口函数只需要等待页面dom树加载完毕后执行.

$(function () {

alert('jQuery入口函数2');

});

$(function () {

alert('jQuery入口函数1');

});

// window.onload = function () {

// alert('原生js入口函数');

// }

//原生js入口函数.

//只要是给window添加load事件,那就是原生js入口函数.

// $(window).load(function () {

// alert("入口函数")

// });

</script>

</body>

</html>

1. 设置、获取文本内容text()

<!DOCTYPE html>

<html>

<head lang="en">

<meta charset="UTF-8">

<title>标题</title>

</head>

<body>

<input type="button" value="获取" id="getBtn"/>

<input type="button" value="设置" id="setBtn"/>

<div id="div1">我是一个div标签

<p>我是一个p标签 <span>span1</span></p>

</div>

<div>我是一个div2标签

<p>我是一个p2标签 <span>span2</span></p>

</div>

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

<script>

$(function () {

//text()方法是用来设置/获取文本的.

//1.获取文本.

//text(); 不给参数

$('#getBtn').click(function () {

//1.1 获取id为div1的文本

//所有的文本都被获取到了.

// console.log($('#div1').text());

//1.2 获取标签为div的文本.

//获取到每一个div的文本.

console.log($('div').text()); //隐式迭代,偷偷的遍历.

});

//2.设置文本.

//text(需要设置的文本); 给参数

//会覆盖元素的内容.

//如果设置的文本中包含了标签,是不会把标签给解析出来的.

$('#setBtn').click(function () {

//2.1 给id为div的标签设置文本

$('#div1').text('我是设置的文本<a href="https://www.baidu.com">我是超链接</a>')

});

});

</script>

</body>

</html>

1. 设置、获取样式css()

<!DOCTYPE html>

<html>

<head lang="en">

<meta charset="UTF-8">

<title>标题</title>

<style>

#div1 {

height: 200px;

background-color: pink;

border: 2px solid red;

}

</style>

</head>

<body>

<input type="button" value="获取" id="getBtn"/>

<input type="button" value="设置" id="setBtn"/>

<div id="div1" style="width:200px"></div>

<div id="div2" style="width:300px"></div>

<div id="div3" style="width:400px"></div>

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

<script>

$(function () {

//css(); 获取/设置样式的.

//1.获取样式

//css(样式名);

$('#getBtn').click(function () {

//1.1 获取id为div1的标签的样式.

// console.log($('#div1').css('width')); //'200px'

// console.log($('#div1').css('height'));//'200px'

// console.log($('#div1').css('background-color'));//'rgb(255, 192, 203)'

// console.log($('#div1').css('backgroundColor'));//'rgb(255, 192, 203)'

// console.log($('#div1').css('border'));//'2px solid rgb(255, 0, 0)'

//在ie浏览器中,获取样式,要写准确的样式,比如边框要写是那个边框那个样式.

//console.log($('#div1').css('border-top-width'));//'2px'

//1.2 获取标签为div的元素的样式.

//如果获取样式的jQuery对象里面包含多个dom对象,那只会返回第一个dom对象的样式.

console.log($('div').css('width'));//'200px'

});

//2.设置样式.

//css()方法设置的样式在元素的行内.

//css(样式名,样式值);

//css({ 这个对象就可以传多个样式名样式值 });

$('#setBtn').click(function () {

//2.1 设置id为div1的样式.

//设置单样式

$('#div1').css('width',300);

$('#div1').css('height','300px');

$('#div1').css('backgroundColor','green');

$('#div1').css('border-top-width',10);

//2.2 设置id为div1的样式.

//设置多样式

// $('#div1').css({

// width:300,

// 'height':'300px',

// backgroundColor:'green',

// 'border-top-width':10

// });

//2.3 设置标签为div的元素

$('div').css({

width:300,

'height':'300px',

backgroundColor:'green',

margin:10

});

});

});

</script>

</body>

</html>

1. 基本选择器

<!DOCTYPE html>

<html>

<head lang="en">

<meta charset="UTF-8">

<title>基本选择器</title>

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

<script>

$(function () {

// 基本选择器 和css里面的选择器是一样的

// id选择器 $('#id')

// 类选择器 $(".类名")

// 标签选择器 $("标签名")

// 并集选择器 $('.box,.box2')

// 交集选择器 $('li.current')

//抗日开始，action

//首先，大当家的立威 把自己变大

// id选择器

// 格式：$("#id")

// 作用：通过id来获取页面中的元素

$('#one').css('fontSize','40px');

// 自己立了为变大了 也不能寒酸了二当家的 所以把二当家的也变大一点

// 类选择器

// 格式：$(".class");

// 作用：获取页面中一类元素，通过class属性来获取

// jQuery中支持隐式迭代，迭代就是遍历的意思

$('.two').css('fontSize','30px');

// 有一天官兵来剿匪 到了紧急时刻 不算是伙夫还是哨兵 通通上战场

//要把所有的并都派上去,上战场，前线的兵背景色都是绿色。

// 标签选择器

// 格式；$("标签名");

// 通过标签来获取对应元素

$('li').css('backgroundColor','green');

//打不赢就要跑嘛， 所以装备要带足 吃饭的家伙事不能少, 伙夫背着粮食 锅碗瓢盆, 所以伙夫和卫生员是没有战斗力的，应该撤下来。

//需要把伙夫和医生的背景色改成白色。

// 伙夫和卫生员也想着要帮忙为抗日做贡献。

// 并集选择器

// 格式：$("selector1,selector2,...");

// 功能：获取多个选择器的并集

$('.hf,.doctor').css('backgroundColor','white');

//现在，突然发现官兵很快攻进了老巢。为什么呢？现在在土匪里面出现内奸。

//现在把土匪里面的内奸找出来，不能干扰到老百姓 而且老百姓里面也有间隙 但是为了不打扰百姓 咱只招土匪兵里面的间隙

// 交集选择器 li.current

// 格式： $("selector1selector2");注意，中间没有任何的空格

// 获取两个选择器之间的交集。

$('li.nj').css('backgroundColor','green');

})

</script>

</head>

<body>

<div>

<ul>

<li id="one">我是大当家的</li>

</ul>

<!-- 东山头 -->

<ul id="east">

<li class="two">我是二把手</li>

<li>哨兵</li>

<li>土匪蛋子</li>

<li>土匪蛋子</li>

<li>土匪蛋子</li>

<li class="hf">伙夫</li>

<li class="doctor">医生</li>

</ul>

<!-- 西山头 -->

<ul id="west">

<li class="two">我是二把手</li>

<li>哨兵</li>

<li class="nj">土匪蛋子</li>

<li>土匪蛋子</li>

<li>土匪蛋子</li>

<li class="hf">伙夫</li>

<li class="doctor">医生</li>

</ul>

</div>

<div>

<p class="nj">老百姓</p>

<p class="nj">老百姓</p>

<p>老百姓</p>

<p>老百姓</p>

<p>老百姓</p>

<p>老百姓</p>

<p>老百姓</p>

</div>

</body>

</html>

1. 层次选择器

<!DOCTYPE html>

<html>

<head lang="en">

<meta charset="UTF-8">

<title>标题</title>

</head>

<body>

<p></p>

<div id="father">

<div>

<span>span1</span>

<span>span2</span>

<span>span3</span>

</div>

<div>

<b>1</b>

<div>div1</div>

<div>div2</div>

<div>div3</div>

<p></p>

</div>

<div></div>

<p></p>

<p></p>

<p></p>

<span>sss1</span>

</div>

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

<script>

$(function () {

//需求1:查找到id为father的这个div的所有子代div.

console.log($('#father>div'));

//需求2:查找到id为father的这个div的所有子代div,以及所有子代p.

//console.log($('#father>div,p'));//这样不能,这样获取到的是id为father的div的子div以及页面上所有的p.

console.log($('#father>div , #father>p'));

//需求3:查找id为father这个div的所有后代div.

console.log($('#father div'));

});

</script>

</body>

</html>

1. 隔行变色

<!DOCTYPE html>

<html>

<head lang="en">

<meta charset="UTF-8">

<title>标题</title>

</head>

<body>

<ul>

<li>我是第1个li标签</li>

<li>我是第2个li标签</li>

<li>我是第3个li标签</li>

<li>我是第4个li标签</li>

<li>我是第5个li标签</li>

<li>我是第6个li标签</li>

<li>我是第7个li标签</li>

<li>我是第8个li标签</li>

<li>我是第9个li标签</li>

<li>我是第10个li标签</li>

</ul>

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

<script>

$(function () {

//隔行变色

$('li:odd').css('backgroundColor','pink');

$('li:even').css('backgroundColor','skyblue');

//把首尾两行背景色变成绿色.

$('li:eq(0)').css('backgroundColor','green');

$('li:eq(9)').css('backgroundColor','green');

//补充.

var bgColor = null;//声明一个变量,用来保存li标签鼠标没有移入之前的颜色.

//鼠标移入事件.

$('li').mouseover(function () {

//当前鼠标移入的li标签背景色变红.

//console.log(this);//此时this是一个dom对象.

//鼠标移入但是没有修改颜色之前,把原来的颜色保存起来.

bgColor = $(this).css('backgroundColor');//用css()方法获取背景色.

$(this).css('backgroundColor','red');

});

//鼠标移出事件.

$('li').mouseout(function () {

$(this).css('backgroundColor',bgColor);

});

});

</script>

</body>

</html>

1. 操作class

<!DOCTYPE html>

<html>

<head lang="en">

<meta charset="UTF-8">

<title>标题</title>

<style>

div {

width: 100px;

height: 100px;

margin-top:10px;

}

.bgc {

background-color: green;

}

.fontSize30 {

font-size: 30px;

}

.width200 {

width: 200px;

}

.height200 {

height: 200px;

}

</style>

</head>

<body>

<input type="button" value="添加类" id="addClass"/>

<input type="button" value="移除类" id="removeClass"/>

<input type="button" value="判断类" id="hasClass"/>

<input type="button" value="切换类" id="toggleClass"/>

<div id="div1" class="bgc ">div1</div>

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

<script>

$(function () {

//css()设置样式,设置在的行内

//1.添加类

$('#addClass').click(function () {

//给id为div1的元素添加类.

//设置单个类

//$('#div1').addClass('fontSize30');

//设置多个类

$('#div1').addClass('fontSize30 width200');

});

//2.移除类

$('#removeClass').click(function () {

//给id为div1的元素移除类

$('#div1').removeClass('fontSize30 width200');

//$('#div1').removeClass();//全部类都移除了

});

//3.判断类

//判断某个元素是否拥有某个类,如果有返回ture,如果没有返回false.

$('#hasClass').click(function () {

console.log($('#div1').hasClass('fontSize30'));

});

//4.切换类

//如果操作的元素拥有这个类,就移除这个类; 如果操作的元素没有这个类,就添加这个类.

$('#toggleClass').click(function () {

$('#div1').toggleClass('fontSize30');

});

});

</script>

</body>

</html>

1. 案例-tab栏切换

<!DOCTYPE html>

<html>

<head lang="en">

<meta charset="UTF-8">

<title></title>

<style type="text/css">

\* {

margin: 0;

padding: 0;

}

ul {

list-style: none;

}

.wrapper {

width: 1000px;

height: 475px;

margin: 0 auto;

margin-top: 100px;

}

.tab {

border: 1px solid #ddd;

border-bottom: 0;

height: 36px;

width: 320px;

}

.tab li {

position: relative;

float: left;

width: 80px;

height: 34px;

line-height: 34px;

text-align: center;

cursor: pointer;

border-top: 4px solid #fff;

}

.tab span {

position: absolute;

right: 0;

top: 10px;

background: #ddd;

width: 1px;

height: 14px;

overflow: hidden;

}

.products {

width: 1002px;

border: 1px solid #ddd;

height: 476px;

}

.products .main {

float: left;

display: none;

}

.products .main.selected {

display: block;

}

.tab li.active {

border-color: red;

border-bottom: 0;

}

</style>

</head>

<body>

<div class="wrapper">

<ul class="tab">

<li class="tab-item active">国际大牌<span>◆</span></li>

<li class="tab-item">国妆名牌<span>◆</span></li>

<li class="tab-item">清洁用品<span>◆</span></li>

<li class="tab-item">男士精品</li>

</ul>

<div class="products">

<div class="main selected">

<a href="###"><img src="images/guojidapai.jpg" alt=""/></a>

</div>

<div class="main">

<a href="###"><img src="images/guozhuangmingpin.jpg" alt=""/></a>

</div>

<div class="main">

<a href="###"><img src="images/qingjieyongpin.jpg" alt=""/></a>

</div>

<div class="main">

<a href="###"><img src="images/nanshijingpin.jpg" alt=""/></a>

</div>

</div>

</div>

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

<script>

$(function () {

//需求:给tab栏的每一个li标签设置鼠标移入事件: 当前li添加active类,其他的兄弟li移除active类.

// 找到当前tab栏索引一致的div,让他添加 selected类,其他的兄弟div移除selected类.

$('.wrapper .tab-item').mouseenter(function () {

//当前鼠标移入的li添加active类,其他的兄弟li移除active类.

$(this).addClass('active').siblings('li').removeClass('active');

//获取当前鼠标移入的li标签的索引

var idx = $(this).index();

//找到当前tab栏索引一致的div,让他添加 selected类,其他的兄弟div移除selected类.

$('.products .main').eq(idx).addClass('selected').siblings('div').removeClass('selected');

});

});

</script>

</body>

</html>

1. dom操作find

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

<script src="https://cdn.bootcss.com/jquery/1.11.3/jquery.js"></script>

</head>

<body>

<div class="wrapper">

<ul>

<li>1</li>

<li>2</li>

<li>3</li>

<li>4</li>

<li>5</li>

</ul>

</div>

<ul>

<li>1</li>

<li>2</li>

<li>3</li>

<li>4</li>

<li>5</li>

</ul>

<script>

// 场景：给wrapper一个相对定位，给ul一个绝对定位，给li一个颜色

// $(".wrapper").css({position : "relative"});

// $(".wrapper ul").css({position : "absolute"});

// $(".wrapper ul li").css({color : "red"});

// 写起来比较麻烦，使用find来在指定条件下进行选择

// find()

$(".wrapper")

.css({position : "relative"})

.find("ul")

.css({position : "absolute"})

.find("li")

.css({color : "red"})

//代码其实没有减少多少，但是效率提升，不用每次从头进行查找

//prevObject 中有一个属性，表示的是上一个对象

console.log( $(".wrapper") )

console.log( $(".wrapper").find("ul") )

console.log( $(".wrapper").find("ul").find("li") )

// 问题 true？false

console.log( $(".wrapper").find("ul").find("li").prevObject == $(".wrapper").find("ul") )

</script>

</body>

</html>

1. dom操作filter，not.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

<script src="https://cdn.bootcss.com/jquery/1.11.3/jquery.js"></script>

</head>

<body>

<div class="wrapper">

<ul>

<li>1</li>

<li class="demo">2</li>

<li>3</li>

<li>4</li>

<li class="demo">5</li>

</ul>

</div>

<ul>

<li>1</li>

<li class="demo">2</li>

<li>3</li>

<li>4</li>

<li class="demo">5</li>

</ul>

<script>

// 把demo获取到进行操作

// console.log( $(".wrapper ul").find(".demo") )

// filter() 过滤

// 两种参数：选择器，function

//第一种为选择器

// $(".wrapper")

// .find("ul")

// .find("li")

// .filter(".demo")

// .css({color : "red"})

//传入jquery选择器筛选

// $(".wrapper ul")

// .filter("li:odd") //注意：只能在选择好的基础之上进行筛选

// .css({color : "red"})

// $(".wrapper ul")

// .find("li")

// .filter(":odd")

// .css({color : "red"})

// 第二种为函数

$(".wrapper ul li").filter(function(index,ele){

console.log(ele)

console.log(index)

// return false //返回boolean表示是否选中

return index % 2 == 0;

}).css({color : "red"})

// not()

// 跟filter完全相反

</script>

</body>

</html>

1. dom操作has，is

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

<script src="https://cdn.bootcss.com/jquery/1.11.3/jquery.js"></script>

</head>

<body>

<ul>

<li>

<ul>

<li>1</li>

<li>2</li>

<li>3</li>

<li>4</li>

<li>5</li>

</ul>

</li>

<li></li>

<li></li>

<li></li>

<li></li>

</ul>

<div class="wrapper">

<span class="demo">span-1</span>

<span class="demo">span-2</span>

<p class="demo">p-1</p>

<span class="demo">span-3</span>

</div>

<ul>

<li>1</li>

<li>2</li>

<li>3</li>

<li>4</li>

<li>5</li>

</ul>

<script>

//给后代中有ul的li加样式

// has()

$("li").has("ul").css({background : "red"});

// is() 返回一个boolean值

var flag = $(".wrapper .demo").is("span"); //是否含有span

console.log(flag);

// is使用场景：

// 当点击li的时候输出li的内容，当点击ul的时候，输出整体的内容

$("ul").click(function(e){

if( $(e.target).is("li") ){

console.log( $(e.target).text() )

}else{

console.log( $(this).text() )

}

})

</script>

</body>

</html>

1. dom操作add、end

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

<!-- <script src="https://cdn.bootcss.com/jquery/1.11.3/jquery.js"></script> -->

<script src="08-myjQuery.js"></script>

</head>

<body>

<div class="wrapper">

<ul class="demo">

<li></li>

<li></li>

<li></li>

<li></li>

<li></li>

</ul>

</div>

<ul>

<li></li>

<li></li>

<li></li>

<li></li>

<li></li>

</ul>

<script>

// 想要给.wrapper和ul都有样式

// $(".wrapper")

// .css({position : "absolute"})

// .find("ul")

// .css({position : "absolute"})

//add() 集合操作

// $(".wrapper")

// .add("ul")

// .css({position : "absolute"})

// end() 回退操作 类似于里面的prevObject属性

// $(".wrapper")

// .css({position : "absolute"})

// .find("ul")

// .css({position : "absolute"})

// .end()

// .css({position : "relative"})

// 源码中测试add中也是含有end方法的

// $(".wrapper")

// .add(".ul")

// .css({position : "relative"})

// .end()

// .css({background : "red"})

// 测试myjQuery

console.log($(".wrapper").add(".demo").end().eq(0).css({"color" : "red"}))

</script>

</body>

</html>

1. css()

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

<script src="https://cdn.bootcss.com/jquery/1.11.3/jquery.js"></script>

</head>

<body>

<div class="demo"></div>

<script>

// css() -- 赋值

// 1.修改一个属性，传递两个参数

$(".demo").css("width",100) //可以不用带单位

// 2.修改多个属性，传递一个对象

$(".demo")

.css({"height" : "100px","background" : "red"})

.css({width : "+=100px"}) //可以做加法运算

// css() -- 取值

console.log( $(".demo").css("width") )

console.log( $(".demo").css("background") )//出来的是计算属性

</script>

</body>

</html>

1. attr,prop

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

<script src="https://cdn.bootcss.com/jquery/1.11.3/jquery.js"></script>

</head>

<body>

<div class="demo" abc="bcd"></div>

<input type="text" abc="efg">

<input class="demo2" type="checkbox" checked="false" abc="bcd">

<script>

// attr() 基于setAttribute getAttribute

// 取值

console.log( $(".demo").attr("class") )

console.log( $(".demo").attr("abc") )

console.log( $("input").attr("abc") )

// 赋值 一个属性

$(".demo").attr("id","test1")

// 赋值 多个属性

$(".demo").attr({"id":"test1","aslkdfj":"asuidyhfiuyh"})

// 针对于selected checked disabled

// attr()

console.log( $(".demo2").attr("checked") ) //无论添加什么值都是checked

// prop 是特性， attr是属性，属性包含特性，prop关注的是状态，checked是否存在

// 取值

console.log( $(".demo2").prop("class") ) //demo2

console.log( $(".demo2").prop("checked") ) //true

console.log( $(".demo2").prop("abc") ) //undefined

// 赋值

$(".demo2").prop("blb","log");

//属性通过prop赋值，在标签中看不到，但是可以通过查看

console.log($(".demo2").prop("blb"))

//特性可以直接查看

//设置一个

$(".demo2").prop("id","test")

//设置多个

$(".demo2").prop({"id":"test1","title":"asdfsdaf"})

</script>

</body>

</html>

1. 插入方法

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

<script src="https://cdn.bootcss.com/jquery/1.11.3/jquery.js"></script>

</head>

<body>

<ul id="city">

<li id="bj">北京</li>

<li id="tj">天津</li>

<li id="nj">南京</li>

</ul>

<ul id="game">

<li id="cs">反恐精英</li>

<li id="xj">星际争霸</li>

<li id="ms">魔兽世界</li>

</ul>

<script>

// 这一系列的方法就是为了完成链式调用的方便

// 所有的都是剪贴的操作

// .insertAfter() 将前面的插入到后面成下个兄弟元素

// $("#tj").insertAfter($("#xj"));

// .after() 将后面的插入到前面下一个兄弟元素

// $("#tj").after($("#xj"));

// .insertBefore() 将前面的插入到后面成上一个兄弟元素

// $("#tj").insertBefore($("#xj"));

// .before() 将后面的插入到前面上一个兄弟元素

// $("#tj").before($("#xj"));

// .appendTo() 将前面的插入到后面的里面成为最后一个子节点

// $("#tj").appendTo($("#xj"));

// .append() 将后面的插入到前面的里面成为最后一个子节点

// $("#tj").append($("#xj"));

// .prependTo() 将前面的插入到后面的里面成为第一个子节点

// $("#tj").prependTo($("#xj"));

// .prepend() 将后面的插入到前面的里面成为第一个子节点

// $("#tj").prepend($("#xj"));

</script>

</body>

</html>