***jQuery学习内容***

***知识点：***第一种写法:页面中所有的内容全部加载后才触发该事件(标签,图片,外部的js文件)

$(window).load(function () {

console.log('11');

})

jQuery中页面的加载事件可以出现多个,DOM中只能出现一个,出现多个也只是执行最后一个

第二种写法:页面中基本的DOM元素加载后就触发

$(document).ready(function () {

console.log('1');

});

第三种写法:和第二种写法的效果是相同的

jQuery(function () {

console.log('1');

});

第四种写法:和第三种是一样的

$(function () {

console.log('1');

});

DOM中的顶级对象是:document

BOM中的顶级对象是:window

jQuery中顶级对象是:jQuery---但是:$更简单 ,jQuery对象和DOM对象不是同一个对象(但是在页面中确实是同一个标签)

DOM的写法

window.onload = function () {

console.log('助教好帅哦');

};

***知识点：***DOM对象转jQuery对象:例子: $(DOM对象)

jQuery对象转DOM对象:例子: $('#btn')[0]或者: $('#btn').get(0);

层次选择器

//$('#dv p').css('backgroundColor','red');

//div的直接的子元素

//$('#dv>p').css('backgroundColor','red');

//div的后面的兄弟元素

//$('#dv~p').css('backgroundColor','red');

//div后面紧跟的那个p

//$('#dv+p').css('backgroundColor','red');

**案例：隔行变色 even和odd**

$(function () {

//获取按钮,添加点击事件

//隔行变色:奇红偶黄

//odd---奇数

//even--偶数

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

//奇数行的背景为黄色---但是用户看到的这个奇数行恰巧是偶数行

$('#uu>li:odd').css('backgroundColor','yellow');

//偶数行的背景为红色---用户看到的是奇数行

$('#uu>li:even').css('backgroundColor','red');

});

});

**案例：文本高亮显示**

//页面加载

$(function () {

//获取所有的li标签,注册鼠标进入和离开事件

//鼠标进入事件

$('#uu>li').mouseenter(function () {

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

});

//鼠标离开事件

$('#uu>li').mouseleave(function () {

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

});

});

//页面加载

$(function () {

//找class为wrap中直接子元素ul中直接子元素li,并添加鼠标进入事件

$('.wrap>ul>li').mouseenter(function () {

//设置li的子元素ul显示出来

$(this).children('ul').stop().show(500);//500毫秒

});

//鼠标离开事件

$('.wrap>ul>li').mouseleave(function () {

$(this).children('ul').stop().hide(500);//500毫秒

});

});

**案例：下拉框效果实现**

<body>

<div class="wrap">

<ul>

<li>

<a href="javascript:void(0);">帅哥系列</a>

<ul>

<li><a href="javascript:void(0);">乔峰</a></li>

<li><a href="javascript:void(0);">张无忌</a></li>

<li><a href="javascript:void(0);">郭靖</a></li>

</ul>

</li>

<li>

<a href="javascript:void(0);">美女系列</a>

<ul>

<li><a href="javascript:void(0);">柳 岩</a></li>

<li><a href="javascript:void(0);">杨 幂</a></li>

<li><a href="javascript:void(0);">萝 拉</a></li>

</ul>

</li>

<li>

<a href="javascript:void(0);">渣男妇腐女</a>

<ul>

<li><a href="javascript:void(0);">凤姐</a></li>

<li><a href="javascript:void(0);">助教</a></li>

<li><a href="javascript:void(0);">导师</a></li>

</ul>

</li>

</ul>

</div>

</body>

***知识点：***J**avaScript函数库:就是一个js文件,里面封装了很多的函数：**

\* JavaScript函数库,简称:js库

\* jQuery就是js库中的一种

\* 优点:开源,免费,链式编程,隐式迭代,写的少做的多

\* jQuery中的顶级对象:jQuery或者是$

页面加载的不同的方式(jQuery中的写法)

\* 1.$(window).load(function(){});页面中所有的内容全部加载后才触发该事件

\* 2.$(document).ready(function(){});页面中的基本的DOM元素加载后才触发该事件

\* 3.jQuery(function(){});

\* 4.$(function(){});

为什么要学习jQuery,jQuery的本质也是js

\* jQuery的js文件中封装了很多的方法,也可以解决很多相关浏览器的兼容问题

\* DOM的作用:操作页面中的元素(操作页面中的标签)

\* 学习jQuery其实还是在操作DOM元素

jQuery中获取元素是通过选择器的方式来获取元素

\* 1.id选择器 $('#id属性的值')

\* 2.标签选择器 $('标签名字')

\* 3.类选择器 $('.类样式的名字')

\* 4.层次选择器

\* $('div p') 获取的是div中所有的p标签

\* $('div>p') 获取的是div中直接的子元素p标签

\* $('div+p') 获取的是div后面紧跟着的兄弟元素p标签

\* $('div~p') 获取的是div后面所有的兄弟元素p标签

\* $('li:odd') 获取的是奇数的li标签,但是在页面中是偶数的(用户看到的是偶数行的)

\* $('li:even') 获取的是偶数的li标签,但是在页面中是奇数的(用户看到的是奇数行的)

\*

\* .text();设置或者是获取成对的标签中的文本内容的

\* .html();设置或者是获取成对的标签中的html内容的

\* .val();设置或者是获取表单标签的value属性值的

\* .css();设置元素的样式的

\*

\* DOM对象转jQuery对象,为了方便操作

\* $(DOM对象)----此时就是一个jQuery对象

\*

\*

jQuery对象转DOM对象,有可能jQuery中没有封装相关的兼容代码,此时必须要使用原生的js来实现该功能,那么可能会把jQuery对象转DOM对象,有的时候在元素的相关事件中要使用该对象,此时的this是属于DOM对象,需要转jQuery对象(DOM转jQuery的)

\* 1. $('#btn')[0]

\* 2. $('#btn').get(0)

\* $(this)---把thisDOM对象转jQuery对象

**通过索引找到元素eq lt gt**

$(function () {

//点击按钮显示效果

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

//设置索引为5的li改变背景颜色

//$('ul>li:eq(5)').css('backgroundColor','red');

//索引小于5的所有的li改变背景颜色

//$('ul>li:lt(5)').css('backgroundColor','red');

//索引大于5的所有的li改变背景颜色

//$('ul>li:gt(5)').css('backgroundColor','red');

});

});

***知识点：***函数调用两种方式：

1>直接调用 ::

function f1() {

// console.log('1');

// }

// f1();

//

// var num=10;

2>函数的自调用----一次性的：

// (function () {

// console.log(1);

// })();//调用

沙箱:模拟真实的环境做实验,效果和真实的效果一样,但是不会影响外面的真实世界

// var num=1000;

// (function () {

// var num=10;

// console.log(num);

// })();

//

// console.log(num);

案例：好友面板切换效果实现

<script>

//页面加载

$(function () {

//设置id为u1的ul中所有的li中的每个ul隐藏

$('#u1>li>ul').hide();

//获取id为u1的ul中的每个li添加点击事件

$('#u1>li').click(function () {

//当前的li中的ul要显示出来

$(this).find('ul').show(500);

//当前的li的其他的兄弟元素li中的ul要隐藏

$(this).siblings('li').find('ul').hide(500);

});

});

</script>

</head>

<body>

<div style=" width:200px; height:500px; border:1px solid red;">

<ul id="u1" style=" list-style-type:none; margin:0; padding:0; text-align:center;">

<li>

幼儿园同学

<ul>

<li>鼻涕虫</li>

<li>爱哭鬼</li>

<li>张大胆</li>

<li>班主任</li>

</ul>

</li>

<li>小学同学

<ul>

<li>张三丰</li>

<li>张无忌</li>

<li>乔布斯</li>

<li>助教</li>

</ul>

</li>

<li>

初中同学

<ul>

<li>盖茨</li>

<li>川普</li>

<li>奥巴马</li>

<li>凤姐</li>

</ul>

</li>

</ul>

</div>

</body>

**元素的样式操作推荐写法**

var json={'width':'300px','height':'200px','opacity':0.5,'backgroundColor':'green'};

$('div).css(json);

**按钮显示和隐藏，先进行判断**

$('#btn3').click(function () {

//判断div标签是否应用了cls类样式

if($('#dv').hasClass('cls')){

//干掉这个类样式

$('#dv').removeClass('cls');

}else{

//没有应用这个类样式,那么就添加类样式

$('#dv').addClass('cls');

}

});

});

**类样式切换 toggleClass**

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

//切换类样式的方法

$('body').toggleClass('cls');

});

});

**设置相关兄弟元素样式**

$('ul>li').mouseenter(function () {

//当前的为红色

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

}).click(function () {

//断链

//.end()---恢复断链(恢复到断链之前的效果)

$(this).prevAll().css('backgroundColor','yellow').end().nextAll().css('backgroundColor','blue');

}).mouseleave(function () {

//干掉当前的背景颜色

//$(this).css('backgroundColor','').siblings().css('backgroundColor','');

$('ul>li').css('backgroundColor','');

});

});

**动画方法：**

//.hide和.show中有两个参数:

//参数1:时间---毫秒

//参数2:回调函数----函数作为参数使用

//在动画结束后才执行回调函数中的代码

//页面加载

$(function () {

//隐藏

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

$('#dv').hide(500,function () {

alert('结束了');

});

});

//显示

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

$('#dv').show(500,function () {

alert('嗯,结束了');

});

});

});

.slideUp和.slideDown的方法的参数一样

//参数1:时间---毫秒

//参数2:回调函数

//页面加载

$(function () {

//隐藏

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

//滑入

$('#dv').slideUp(1000,function () {

alert('哦,你干啥捏');

});

});

//显示

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

//滑出

$('#dv').slideDown(1000,function () {

alert('出来了');

});

});

$('#btn3').click(function () {

//滑出

$('#dv').slideToggle();

});

});

.fadeOut和.fadeIn参数一样

//参数1:时间---毫秒

//参数2:回调函数

//页面加载

$(function () {

//隐藏

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

//淡入

$('#dv').fadeOut(1000,function () {

alert('哭了哇,助教迪斯噶');

});

});

//显示

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

//淡出

$('#dv').fadeIn(1000);

});

$('#btn3').click(function () {

//滑出

$('#dv').fadeToggle();

});

});

//页面加载

$(function () {

//点击按钮实现动画

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

$('#im').animate({'width':'300px','height':'300px','left':'50px','top':'500px'},1000,function () {

$('#im').animate({'width':'30px','height':'30px','left':'500px','top':'50px','opacity':0.2},3000);

});

});

});

***知识点：***DOM中创建元素：

document.getElementById('btn').onclick=function () {

document.write('<p>这是p</p>');

};

***知识点：***jquery中动态创建元素：

$(function () {

//点击按钮创建一个p标签添加到div中

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

//主动的

//子级元素.appendTo(父级元素);

$('<p>这是一个p</p>').appendTo($('#dv'));

});

});

jQuery**遍历方法**.each()

//参数1:索引

//参数2:元素

***知识点：***.each()方法是jQuery中封装的一个方法,本身jQuery中有隐式迭代(自动的帮我们遍历元素)程序员想要自己操作每个元素标签,此时可以调用each方法,并且传入一个函数,在函数中传入两个参数,

希望操作每个单独的元素,此时就可以使用each方法

$('li').each(function (index, ele) {

//ele----就是每个li标签

//为每个li标签添加一个自定义属性

$(ele).attr('score', (index + 1) \* 10);

});

//点击每个li显示自己的自定义属性的值

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

console.log($(this).attr('score'));

});

**自定属性**

//点击每个li都可以显示自己的标签中自定义属性的值

//$('li').click(function () {

//DOM写法

//alert(this.getAttribute('score'));

//jQuery的写法

//alert($(this).attr('score'));

//li标签有很多,每个li标签都有自己的自定义属性和值

//}); if ($('#r2').attr('checked') == 'undefined') {

// $('#r2').attr('checked', true);

// }else{

// $('#r2').attr('checked', false);

// }

**.prop方法**

.attr()方法主要操作元素的自定义属性和基本的属性,如果想要设置和获取单选框是否选中,此时推荐用.prop()方法

$(function () {

//点击按钮

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

//先判断.是否选中

if($('#ck').prop('checked')){

//如果为true,说明已经选中,那就应该不选中

$('#ck').prop('checked',false);

}else{

//没有选中,那就应该选中

$('#ck').prop('checked',true);

}

});

});

//如果操作的是可以选中的标签,建议使用prop()方法来操作

//.prop('属性的名字','值');---设置该属性值

//.prop('属性名字');---获取该属性的值

**案例：全选反选jquery实现**

$(function () {

//最上面的复选框的点击事件

$('#j\_cbAll').click(function () {

//设置tbody中所有复选框的选中状态和上面全选的复选框的选中状态一致

$('#j\_tb input').prop('checked', $('#j\_cbAll').prop('checked'));=>全选

});

//下面所有的复选框添加点击事件

$('#j\_tb input').click(function () { =>反选

//先获取所有的复选框的个数

var length1 = $('#j\_tb input').length;

//获取所有选中的复选框的个数

var length2 = $('#j\_tb input:checked').length;

//对比,如果相同则就全选,否则就不选中(全选的复选框)

$('#j\_cbAll').prop('checked', length1 == length2);

// if(length1==length2){

// $('#j\_cbAll').prop('checked',true);

// }else{

// $('#j\_cbAll').prop('checked',false);

// }

});

});

***知识点：***DOM中的三大系列:

offset系列:

\* offsetLeft:元素距离左边位置的距离值

\* offsetTop:元素距离上面位置的距离值

\* offsetWidth:元素的宽

\* offsetHeight:元素的高

\*

scroll系列:

\* scrollLeft:向左卷曲出去的距离值,向左滚出去的距离

\* scrollTop:向上卷曲出去的距离值

\* scrollWidth:里面内容的实际的宽度,如果内容少就是元素的宽

\* scrollHeight:里面内容的实际的高度,如果内容少就是元素的高

\*

client系列:

\* clientX:可视区域的横坐标

\* clientY:可视区域的纵坐标

\* clientWidth: 可视区域的宽

\* clientHeight: 可视区域的高

\* clientLeft: 边框的宽

\* clientTop: 边框的宽

、

//页面加载事件

$(function () {

//点击按钮获取div的宽和高,设置div的宽和高为原来的2倍

//.css('width')----字符串类型的数据,还需要转换

***知识点：***.width()可以设置元素的宽,也可以获取元素的宽=>都是数字类型

.height()可以设置元素的高,也可以获取元素的高=>都是数字类型

总结:如果只是单纯的设置和获取宽和高,推荐使用上面的两个方法

//点击事件

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

//获取宽和高

// var width = $('#dv').css('width');

// width=parseInt(width);

// $('#dv').css('width',width\*2);

//获取元素的宽----数字类型

//console.log($('#dv').width());

//$('#dv').width(500);

// $('#dv').width($('#dv').width()\*2);

// $('#dv').height($('#dv').height()\*2);

console.log($('#dv').width());

});

});

**设置元素的left和top值**

***知识点：①***.offset()得到的是一个对象,该对象中有left和top属性

***②.***offset({'left':'值','top':'值'})-----设置left和top属性值

//页面加载

$(function () {

//点击按钮,设置或者获取元素的left和top值

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

//获取div的left值

//console.log($('#dv').css('left'));

// console.log($('#dv').offset().left);

//console.log($('#dv').offset().top);

//$('#dv').offset({'left':200,'top':30});

});

});

**jQuery中获取scroll系列的属性：**

$(function () {

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

//获取scroll系列的属性

console.log($('#dv').scrollTop());

console.log($('#dv').scrollLeft());

//元素.scrollTop()方法

//元素.scrollLeft()方法

});

});

Jquery事件绑定：

bind的方式绑定事件：

//bind用法一: bind('没有on的事件名字',事件处理函数);

//bind用法二:bind({'事件名字':事件处理函数,'事件名字':事件处理函数...})

//如果bind使用的是键值对的方式为元素绑定事件,多个相同的事件只会执行最后一个,前面的会被覆盖掉

绑定事件有4种方式，

DOM中为按钮绑定多个点击事件，后者会覆盖前者

// document.getElementById('btn').onclick=function () {

// console.log('助教好帅');

// };

总结:如果是先为已经存在的元素绑定事件,后创建的元素,是没有这个事件的,使用的是事件委托方式添加事件

***知识点：***如果想要解绑多个事件,请把所有的事件名字中间用空格隔开

$('#btn1').unbind('mouseenter mouseleave');

delegate的方式绑定事件：

//页面加载

$(function () {

//点击第一个按钮为div中的p绑定点击事件

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

//通过父级元素代理子级元素绑定事件

$('#dv').delegate('p', 'click', function () {

console.log('哎呀,我又被点了');

});

});

//第二个按钮,点击的时候向div中添加一个p标签

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

$('<p>来玩呀~</p>').appendTo($('#dv'));

});

});

delegate的方式绑定事件：

delegate方法的内部是通过on的方式来实现绑定事件的,所以,可以直接使用on的方式来绑定事件

delegate用法:

delegate('选择器','事件名字',事件处理函数);

on用法之一:

on('事件名字','选择器',事件处理函数);

on用法之二:

on('事件名字',事件处理函数)

delegate的方式解绑事件：

//为div添加了点击事件

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

console.log('div被点了');

});

});

//第二个按钮解绑div中的p的事件

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

//undelegate的方式解绑事件

div中的p的点击事件解绑

$('#dv').undelegate('p','click');

div中的p的多个事件解绑

$('#dv').undelegate('p','mouseenter mouseleave');

div中p所有的事件全部被解绑

$('#dv').undelegate('p','');

});

onmouseover事件会产生冒泡

**绑定事件推荐使用on的方式**

***知识点：*以后在jQuery中为元素绑定事件推荐使用on的方式**

//为一个按钮绑定点击事件

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

console.log('被点了');

});

解绑事件on---off()方法解绑

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

// //把div和里面的p的所有的事件全部解绑

// //$('#dv').off();

// //div中的p的点击事件被解绑了,div的事件还是存在的

// //$('#dv').off('click','p');

// //div中的p的鼠标进入和鼠标离开事件都会被解绑

// //$('#dv').off('mouseenter mouseleave','p');

// //是干掉div中所有子元素的点击事件

// $('#dv').off('click','\*\*');

// });

**案例：Jquery中事件触发**

//为文本框添加获取焦点的事件

$('#txt').focus(function () {

$(this).next().text('此时文本框获取焦点事件触发了');

});

//点击按钮的时候要触发文本框获取焦点的事件

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

//方式1:

//$('#txt').focus();

//方式2:

//$('#txt').trigger('focus');

//方式3:

$('#txt').triggerHandler('focus');

//第一种方式和trigger的方式,会触发浏览器的默认行为,但是,triggerHandle这种方式不会触发浏览器的默认行为,是看不到光标闪动的效果

});

<input type="button" value="触发文本框获取焦点的事件" id="btn"/>

<input type="text" value="" id="txt" /><span id="sp"></span>

**函数有返回值 :**

***知识点：***//函数没有返回值:

//函数中如果没有return,则函数没有返回值,如果函数中有return,则该函数中有返回值

//如果函数中有return,但是return后面没有任何内容,则该函数没有返回值(此时函数就是没有明确返回值,则该函数调用后,如果

//对象中有\_\_proto\_\_,函数中有prototype

// //对象能够调用自己的方法

// //对象中的\_\_proto\_\_指向的是所在的构造函数的prototype的cunstructor创建出来的接收该该函数,此时,的结果就是undefined)

**案例：五星好评**

$('.comment>li').mouseenter(function () {

//当前的li和前面所有的li是实心的五角星,后面的所有的li是空心的五角星

$(this).text('★').prevAll('li').text('★').end().nextAll('li').text('☆');

}).click(function () {

//点击的时候为点击的li添加一个标识---记号

$(this).attr('nihao','haha').siblings('li').removeAttr('index');

}).mouseleave(function () {

//设置所有的li都是空心的五角星

$('.comment>li').text('☆');

//根据标识找到对应的li,设置这个li及前面所有的li为实心的五角星

$('.comment>li[nihao=haha]').text('★').prevAll('li').text('★');

});

属性选择器：

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

**①**获取有name属性的input标签

//$('input[name]').css('backgroundColor','red');

//$('input[id]').css('backgroundColor','red');

**②**获取name值是userName的input标签

//$('input[name=userName]').css('backgroundColor','red');

//$('input[name!=userName]').css('backgroundColor','red');

**③**获取的是name属性值是以p开头的input

//$('input[name^=p]').css('backgroundColor','red');

//$('input[name$=e]').css('backgroundColor','red');

**④**name属性的值中有u的input

//$('input[name\*=u]').css('backgroundColor','red');

});

把$的权限给了xy这个变量了---此时当前的这个页面中xy才是jQuery中的顶级对象

var zj = $.noConflict();

//多库共存

var $ = 10;

//$就是一个普通的变量

zj(function () {

//点击事件

zj('#btn').click(function () {

console.log('小杨好帅哦');

});

});

案例：页面加载后,每个li的透明度都是不一样的

$(function () {

each()遍历jQuery中的对象来操作元素的相关样式

每个li的透明度,都要单独的设置

遍历每个li标签,分别设置透明度的值

$('ul>li').each(function (index, ele) {

//该函数会被执行10次,因为有10个li

//此时ele是属于DOM对象

$(ele).css('opacity', (index + 1) / 10);

});

});

***知识点：***jQuery获取的对象是jQuery对象

//DOM的方式获取的对象是DOM对象

//jQuery的本质就是js

//所以,jQuery的方式获取的标签对象,内部实际上就是DOM对象

**案例：点击按钮在div中创建一个p（id的唯一性）**

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

if($('#p1').length==0){

var pObj=$('<p>这是一个p</p>');

pObj.attr('id','p1');

$('#dv').append(pObj);

}

});