目录

[jQuery简介 2](#_Toc533347328)

[jQuery设计思想之写法 2](#_Toc533347329)

[jQuery设计思想之选择元素 3](#_Toc533347330)

[$()下的常用方法一 3](#_Toc533347331)

[$()下的常用方法二 5](#_Toc533347332)

[$()下的常用方法三 8](#_Toc533347333)

[$()下的常用方法四 10](#_Toc533347334)

[$()下的常用方法五 11](#_Toc533347335)

[$()下的常用方法六 11](#_Toc533347336)

[$下的常用方法 13](#_Toc533347337)

[jQuery下使用ajax 14](#_Toc533347338)

[$.get(url [, data ] [, success(data, textStatus, jqXHR) ] [, dataType ]) 14](#_Toc533347339)

[$.getJSON(url [, data ] [, success(data, textStatus, jqXHR) ]) 15](#_Toc533347340)

[$.getScript(url [, data ] [, success(data, textStatus, jqXHR) ]) 15](#_Toc533347341)

[$.post(url [, data ] [, success(data, textStatus, jqXHR) ] [, dataType ]) 16](#_Toc533347342)

[$().load()加载页面和页面片段 16](#_Toc533347343)

[$.ajax(url [, settings ]) 或 $.ajax([settings ]) 17](#_Toc533347344)

[$.ajax() 实现 jsonp 跨域请求 17](#_Toc533347345)

[$.ajaxSetup()预设ajax 18](#_Toc533347346)

[全局 Ajax 事件处理器 19](#_Toc533347347)

[创建jQuery插件 20](#_Toc533347348)

[插件的类型 20](#_Toc533347349)

[开发插件的格式 20](#_Toc533347350)

[$.extend()的另一个作用 20](#_Toc533347351)

[$.插件()—全局(类)方法 插件开发 20](#_Toc533347352)

[$().插件()—对象(实例)方法 插件开发 21](#_Toc533347353)

[$().插件()—对象(实例)方法 插件开发 模板 22](#_Toc533347354)

# jQuery简介

1、什么是JQ？

javascript query(查询) 一个优秀的JS库，大型开发必备

2、JQ的好处？

简化JS的复杂操作

不再需要关心兼容性

提供大量实用方法

3、如何学习JQ？

www.jquery.com

JQ只是辅助工具，要正确面对

需要分阶段学习

# jQuery设计思想之写法

方法函数化

window.onload = function(){

    var oDiv = document.getElementById('div1');

    oDiv.onclick = function(){

        alert( this.innerHTML );

    };

};

$(function(){

    $('#div1').click(function(){

        alert( $(this).html() );

    });

});

/\*

window.onload = function(){};

$(function(){});

function $(){}

innerHTML = 123;

html(123)

function html(){}

onclick = function(){};

click(function(){})

function click(){}

\*/

# jQuery设计思想之选择元素

# $()下的常用方法一

css() html() get()和length属性 click()

has() not() filter()

next() prev() find() eq() index() attr()

addClass()removeClass()toggleClass()add() slice() size() each()

模拟CSS选择元素

$('#div1').css('background','red');

$('div').css('background','red');

$('.box').css('background','red');

独有表达式选择

$('li:eq(2)').css('background','yellow');

$('li:odd').css('background','red');

多种筛选方法

$('li').filter('.box').css('background','red');

$('li').filter('[title=hello]').css('background','red');

<script>

$(function(){

    var elem = $('div');

    var elem2 = elem.add('span');

    elem.css('color','red');

    elem2.css('background','yellow');

    $('li').slice(1,4).css('background','red');

});

$(function(){

    alert( $('li').size() ); //4 像length

    $('li').each(function(i,elem){ //一参：下标 二参 : 每个元素

        $(elem).html(i);

    });

});

</script>

链式操作

$(function(){

    $('#div1').html('hello').css('background','red').click(function(){

        alert(123);

    });

});

取值赋值合体

$(function(){

    oDiv.innerHTML = 'hello'; //赋值

    alert( oDiv.innerHTML ); //取值

    $('#div1').html('hello'); //赋值

    alert( $('#div1').html() ); //取值

    alert( $('li').html() ); //当一组元素的时候，取值是一组中的第一个

    $('li').html('hello'); //当一组元素的时候,赋值是一组中的所有元素

    css('width','200px');

    css('width');

});

JQ与JS关系 可以共存，不能混用

$(function(){

    $('#div1').click(function(){

        //alert( $(this).html() ); //jq的写法

        //alert( this.innerHTML ); //js的写法

        alert( $(this).innerHTML ); //错误的

        alert( this.html() ); //错误的

    });

});

**案例-编写选项卡**

$(function(){

    $('#div1').find('input').click(function(){

        $('#div1').find('input').attr('class','');

        $('#div1').find('div').css('display','none');

        $(this).attr('class','active');

        $('#div1').find('div').eq( $(this).index() ).css('display','block');

    });

});

<div id="div1">

    <input class="active" type="button" value="1" />

<input type="button" value="2" />

<input type="button" value="3" />

<div style="display:block">111111</div>

<div>222222</div>

<div>333333</div>

</div>

# $()下的常用方法二

width() innerWidth() outerWidth()

insertBefore() before() insertAfter() after()

appendTo() append() prependTo() prepend()

wrap() wrapInner() wrapAll() unwrap()

remove() empty() detach()

clone()浅拷贝 clone(true)深拷贝

scrollTop()

**宽高属性的获取**

$().width() width

$().innerWidth() width+padding

$().outerWidth() width+padding+border

**节点的操作**

$(A).insertBefore( $(B) ) 把A元素插入到B元素之前

$(A).before( $(B) ) 在A元素之前插入B元素

$(A).insertAfter( $(B) ) 把A元素插入到B元素之后

$(A).after( $(B) ) 在A元素之后插入B元素

\*\***A、B元素都有可能是已有的，也有可能是创建的新元素**\*\*

\*\***后续的链式操作都是对A元素进行控制**\*\*

$(A).prependTo( $(B) ) 把A元素添加到B元素内容之前

$(A).prepend( $(B) ) 在A元素内容之前添加B元素

$(A).appendTo( $(B) ) 把A元素追加到B元素内容之后

$(A).append( $(B) ) 在A元素内容之后追加B元素

\*\***A、B元素都有可能是已有的，也有可能是创建的新元素**\*\*

\*\***后续的链式操作都是对A元素进行控制**\*\*

$(A).remove() 从文档中移出A 元素

$('div').empty();  //清空内容  遗留节点

$('div').detach() //跟remove方法一样，只不过会保留删除这个元素的操作行为

$(function(){

    $('div').click(function(){

        alert(123);

    });

    var oDiv = $('div').detach();

    $('body').append( oDiv );

});

clone()浅拷贝 clone(true)深拷贝

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

        alert(123);

    });

$('div').clone(true).appendTo( $('span') );

$('span'). wrap('<div></div>') 使用div标签包裹span标签，每一个span标签都用一个div标签包裹

$('span'). wrapAll('<div></div>') 使用div标签包裹span标签，所有span标签都用一个div标签包裹

$('div'). wrapInner('<span></span>'); 使用span标签包裹所有div里面的内容，每一个div的内容都用一个span标签包裹

$('span').unwrap() //删除父级标签

**滚动的距离**

<script>

$(function(){

    $(document).click(function(){

        alert( $(window).scrollTop() ); //滚动距离

    });

});

</script>

**案例-编写弹窗**

<script>

$(function(){

    $('#input1').click(function(){

        var oLogin = $('<div id="login"><p>用户名:<input type="text" /></p><p>密码:<input type="text" /></p><div id="close">X</div></div>');

        $('body').append( oLogin );

oLogin.css('left', ($(window).width()-oLogin.outerWidth())/2);

oLogin.css('top', ($(window).height()-oLogin.outerHeight())/2);

        $('#close').click(function(){

            oLogin.remove();

        });

        $(window).on('resize scroll',function(){

oLogin.css('left' , ($(window).width() - oLogin.outerWidth())/2 );

oLogin.css('top' , ($(window).height() - oLogin.outerHeight())/2 + $(window).scrollTop() );

        });

    });

});

</script>

<body style="height:2000px;">

<input type="button" value="点击" id="input1" />

# $()下的常用方法三

on() off() bind() unbind() one() trigger()

ev ev.pageX ev.pageY ev.which

preventDefault() stopPropagation()

**ev.data ev.target ev.type**

delegate() undelegate()

offset() position()

**JQ中事件的写法**

$(function(){

    /\*$('div').click(function(){

        alert(123);

    });\*/

    /\*$('div').on('click',function(){

        alert(123);

    });\*/

    /\*$('div').on('click mouseover',function(){

        alert(123);

    });\*/

    /\*$('div').on({

        'click' : function(){

            alert(123);

        },

        'mouseover' : function(){

            alert(456);

        }

    });\*/

    $('div').on('click mouseover',function(){

        alert(123);

        $('div').off('mouseover');

});

$('div').one('click',function(){ //只执行事件一次

        alert(123);

});

$('#div1').on('show',function(){

        alert(123);

    });

    $('#div1').on('show',function(){

        alert(456);

    });

    $('#div1').trigger('show');

});

$(function(){

    $('#div1').on('click',{name:'hello'},function(ev){

        console.log(ev.data.name);

        console.log( ev.target );

        console.log( ev.type );

    });

});

</script>

**临时事件对象**

<script>

$(function(){

    /\*$('div').click(function(ev){

        //ev : event对象

        //ev.pageX(相对于文档的) : clientX(相对于可视区)

        //ev.which : keyCode

        ev.preventDefault(); //阻止默认事件

        ev.stopPropagation(); //阻止冒泡的操作

        return false; //阻止默认事件 + 阻止冒泡的操作

    });\*/

    $('div').one('click',function(){ //只执行事件一次

        alert(123);

    });

});

</script>

**事件委托**

$('ul').delegate('li','click',function(){

        this.style.background = 'red';

        $('ul').undelegate();

});

**偏移距离**

<script>

$(function(){

    //div2.offsetLeft

    //alert( $('#div2').offset().left ); //获取到屏幕的左距离

    alert( $('#div2').position().left ); //到有定位的父级的left值,把当前元素转化成类似定位的形式

});

</script>

**案例-编写拖拽**

$(function(){

    var disX = 0;

    var disY = 0;

    $('div').mousedown(function(ev){

        disX = ev.pageX - $(this).offset().left;

        disY = ev.pageY - $(this).offset().top;

        $(document).mousemove(function(ev){

            $('div').css('left',ev.pageX - disX);

            $('div').css('top',ev.pageY - disY);

        });

        $(document).mouseup(function(){

            $(document).off();

        });

        return false;

    });

});

# $()下的常用方法四

parent() offsetParent() parents() closets()

siblings() nextAll() prevA

parentsUntil() nextUntil() prevUntil()

**元素的特殊找法-通过自己找父级**

<script>

$(function(){

    //parent() : 获取父级

    //offsetParent() : 获取有定位的父级

    //$('#div2').parent().css('background','blue');

    $('#div2').offsetParent().css('background','blue');

});

</script>

<script>

//parents() : 获取当前元素的所有祖先节点，参数就是筛选功能

//closest() : 获取最近的指定的祖先节点(包括当前元素自身),必须要写筛选的参数,只能找到一个元素

$(function(){

    //$('#div2').parents('.box').css('background','red');

    $('#div2').closest('.box').css('background','red');

});

</script>

**元素的特殊找法-通过自己找兄弟们和until**

<script>

//siblings() : 找所有的兄弟节点，参数也是筛选功能

//nextAll() : 下面所有的兄弟节点，参数也是筛选功能

//prevAll() : 上面所有的兄弟节点

//\*Until() : 截止

$(function(){

    $('span').nextUntil('h2').css('background','red');

});

</script>

# $()下的常用方法五

val() serialize() serializeArray()

$(function(){

    alert( $('input').val() );

    $('input').val(456);

});

<script>

$(function(){

    console.log($('form').serialize()); //string : a=1&b=2&c=3

    console.log( $('form').serializeArray() );

    /\* [

        { name : 'a' , value : '1' },

        { name : 'b' , value : '2' },

        { name : 'c' , value : '3' }

    ] \*/

});

</script>

# $()下的常用方法六

hover()

show()hide() fadeIn()fadeOut()fadeTo()

slideDown() slideUp() slideToggle()

<script>

$(function(){

    $('#div1').hover(function(){

        //$(this).css('background','blue');

        //$('#div2').hide(3000);

        //$('#div2').fadeOut(1000); //默认400

        //$('#div2').slideUp();

        $('#div2').fadeTo(1000,0.5);

    },function(){

        //$(this).css('background','red');

        //$('#div2').show(3000);

        //$('#div2').fadeIn(1000);

        //$('#div2').slideDown();

        $('#div2').fadeTo(1000,1);

    });

});

</script>

animate() stop() delay()

<script>

//animate() :

//第一个参数 : {} 运动的值和属性

//第二个参数 : 时间(运动快慢的) 默认 : 400

//第三个参数 : 运动形式 只有两种运动形式 ( 默认 : swing(慢快慢) linear(匀速) )

//第四个参数 : 回调函数

$(function(){

    $('#div1').click(function(){

$(this).animate({width : 300 , height : 300} , 4000 , 'linear',function(){alert(123);});

$('#div2').animate({width : 300 , height : 300} , 4000 , 'swing');

    });

});

</script>

$(function(){

    $('#div1').click(function(){

        /\*$(this).animate({width : 300} , 2000 , 'linear',function(){

            $(this).animate({height : 300});

        });\*/

        $(this).animate({width : 300} , 2000).delay(1000).animate({height : 300} , 2000);

    });

    $('#div2').click(function(){

        $('#div1').stop(); //默认 : 只会阻止当前运动

        //$('#div1').stop(true); //阻止后续的运动

        //$('#div1').stop(true,true); //可以立即停止到指定的目标点

        //$('#div1').finish(); //立即停止到所有指定的目标点

    });

});

# $下的常用方法

$().css() $().html() $().val(): 只能给JQ对象用

$.xxx() $.yyy() $.zzz():不仅可以给JQ用，也可以给原生JS用:叫做工具方法

type()

noConflict()

trim() 去除字符串两边空格

inArray(元素,数组)相当于数组的indexOf() makeArray()将类数组转换为数组

parseJSON() 和原生的JSON.parse()一样

proxy()

$.type() : 也是判断类型

    var a = null;

    alert( typeof a );

    alert( $.type(a) );

$.noConflict() : 防止$冲突的 释放$的使用权

var web = $.noConflict();

var $ = 10;

var jQuery=20;

web(function(){

    web('body').css('background','red');

});

$.proxy(): 改变this指向的

    function show(n1,n2){

        alert(n1);

        alert(n2);

        alert(this);

    }

    //show();

    //$.proxy(show , document,3)(4);

    $(document).click( $.proxy(show,window,3,4) );

# jQuery下使用ajax

## $.get(url [, data ] [, success(data, textStatus, jqXHR) ] [, dataType ])

第一个参数URL表示请求的的地址；

第二个参数data表示一个对象结构的键:值对列表；

第三个参数表示成功后的回调函数，改函数的参数表示服务器端响应的信息

第四个参数表示服务器端响应信息返回的内容格式，如XML、HTML、Script、JSON、Text,或者\_default。

<script>

//1页面打开就发送get请求

        $.get('1.txt',function(data){

            alert(data);

        })

//2点击按钮发送请求

        $(function(){

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

                    $.get('2.txt',function(data){

                    alert(data);

                })

            })

        })

//3 get带数据请求

        $(function(){

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

                    $.get('3.php',{name: 'get'},function(data){

                    alert(data);

                })

            })

        })

</script>

**3.php页面代码**

<?php

$name=$\_GET['name'];

if($name=="get")

{

echo "get带数据过来了";

}

**\*\*\*get带数据也可以放在URL后面**

$.get('3.php?name=get',function(data){alert(data);})

## $.getJSON(url [, data ] [, success(data, textStatus, jqXHR) ])

第一个参数URL表示请求的的地址；

第二个参数data表示一个对象结构的键:值对列表；

第三个参数表示成功后的回调函数，改函数的参数表示服务器端响应的信息

<script>

        $(function(){

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

                /\* $.get('4.php',function(data){

                        var data=$.parseJSON(data);

                    alert(data[0].name+data[0].pass+data[0].age);

                }) \*/

                    $.getJSON('4.php',function(data){

                    alert(data[0].name+data[0].pass+data[0].age);

                })

            })

        })

</script>

4.php页面代码

<?php

$arr=array(

    array('name'=>"zhu",'pass'=>"123456",'age'=>"1"),

    array('name'=>"zhang",'pass'=>"abcde",'age'=>"2")

);

echo json\_encode($arr);

## $.getScript(url [, data ] [, success(data, textStatus, jqXHR) ])

第一个参数URL表示请求的的地址；

第二个参数data表示一个对象结构的键:值对列表；

第三个参数表示成功后的回调函数，改函数的参数表示服务器端响应的信息

<script>

$(function(){

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

/\* $.get('5.js',function(data){

data;

}) \*/

$.getScript('5.js',function(data){

data;

})

})

})

</script>

5.js文件中的代码

alert("5.js文件");

## $.post(url [, data ] [, success(data, textStatus, jqXHR) ] [, dataType ])

$.get()与$.post()用法上完全相同

**区别：**1、发送数据的方式不一样

2、数据量不一样

3、get方式有缓存post方式没缓存

## $().load()加载页面和页面片段

加载整个页面

<script>

$(function(){

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

$("#load").load("6-1.html");

});

});

</script>

加载页面片段

<script>

$(function(){

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

//$("#load").load("6-2.html #container");

$("#load").load("6-2.html #container,#box");

//url+空格+选择器（最好是id选择器，火狐对类选择器不支持）

});

});

</script>

**用 $.ajax() 方法 取代 $().load()**

<script>

$(function(){

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

var str=($.ajax({

url:"6-1.html",

async:false//这里必须设为同步加载

})).responseText;

$("#load").html(str);

});

});

</script>

## $.ajax(url [, settings ]) 或 $.ajax([settings ])

<script>

$(function(){

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

$.ajax({

url:"7.txt",

type:'get',

dataType:"text",

async:true,

success:function(data){

alert(data);

},

error:function(xhr,textStatus,errorThrown) {

alert(errorThrown);

}

})

});

})

</script>

## $.ajax() 实现 jsonp 跨域请求

<script>

$(function(){

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

/\* $.get("http://127.0.0.1/5JQ-AJAX/7.txt",function(data){

alert(data);

}) \*/

$.ajax({

url:"http://127.0.0.1/5JQ-AJAX/8.php",

type:"get",

dataType:"jsonp",

success:function(data){

console.log(data);

}

})

});

})

</script>

案例

https://blog.csdn.net/csdn\_yudong/article/details/52683018

## $.ajaxSetup()预设ajax

<script>

$(function(){

$.ajaxSetup({

url:"9.php",

type:"post",

dataType:"text",

global:false,//禁止触发全局ajax事件

success:function(data){

alert(data);

}

});

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

$.ajax({

data:"name=lc"

})

});

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

$.ajax({

data:"name=web"

})

});

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

$.ajax({

data:"name=jquery"

})

});

});

</script>

<?php

header('content-type:text/html;charset="utf-8"');

$name=$\_POST['name'];

if($name)

{

echo $name;

}else{

echo "没有收到信息";

}

## 全局 Ajax 事件处理器

$(document).ajaxComplete(callback)

$(document).ajaxError(callback)

$(document).ajaxSend(callback)

$(document).ajaxStart(callback)

$(document).ajaxStop(callback)

$(document).ajaxSuccess(callback)

function callback(event, xhr, settings) {

if ( settings.url === "test.html" ) {

$( ".log" ).text( "触发完成、错误、发送、开始、停止、成功的处理结果 " +

xhr.response\* );

}

}

# 创建jQuery插件

## 插件的类型

**①全局（类）方法 【开发频率低】**  **②对象（实例）方法【开发频率高】** ***~~选择器~~***

## 开发插件的格式

$.插件=fun $.名字空间={插件:fun,插件:fun} $.extend({})

$.fn.插件=fun $.fn.名字空间={插件:fun,插件:fun} $.fn.extend({})

{

方法名:function(){},

方法名:function(){},

……

}

## $.extend()的另一个作用

var newObject=$.extend({},{},{}…) 合并多个对象

<script>

function fn(arg){

var newObject=$.extend({

name1:"value1",

name2:"value2",

name3:"value3"

},arg);

return newObject;

}

console.log(fn({name4:"value4",name5:"value5"}));//添加

console.log(fn({name1:"value4",name2:"value5"}));//覆盖

console.log(fn());//默认

</script>

## $.插件()—全局(类)方法 插件开发

//1.直接给jquery添加全局函数

jQuery.myAlert1=function (str) {

alert(str);

};

//2.使用命名空间(如果不使用命名空间容易和其他引入的JS库里面的同名方法冲突)

jQuery.webdev={

myAlert2:function (str) {

alert(str);

},

centerWindow:function (obj) {

obj.css({

'top':($(window).height()-obj.height())/2,

'left':($(window).width()-obj.width())/2

});

//必须进行返回对象的操作，否则就不能继续往下进行链式操作了。。

return obj;

}

};

//3.用extend()方法。

jQuery.extend({

myAlert3:function (str1) {

alert(str1);

},

myAlert4:function () {

alert(11111);

}

});

## $().插件()—对象(实例)方法 插件开发

<script>

jQuery.fn.myAlert1=function(msg)

{

alert(msg)

}

jQuery.fn.webdev={

myAlert2:function(msg)

{

alert(msg)

},

myAlert3:function(msg)

{

alert(msg)

}

}

jQuery.fn.extend({

centerWindow:function()

{

$(this).css("left",($(window).innerWidth()-$(this).width())/2);

$(this).css("top",($(window).innerHeight()-$(this).height())/2);

},

myAlert:function(msg)

{

alert(msg)

}

})

$(function(){

$("#box").css({width:200,height:300,backgroundColor:"red",position:"absolute"});

$("#box").click(function(){

$(this).centerWindow();

$(this).myAlert("abc");

});

})

</script>

## $().插件()—对象(实例)方法 插件开发 模板

;(function ($) {

$.fn.plugin=function (options) {

var defaults={

//各种参数、各种属性

};

//options合并到defaults上,defaults继承了options上的各种属性和方法,将所有的赋值给endOptions

var endOptions=$.extend(defaults,options);

this.each(function () {

//实现功能的代码

});

};

})(jQuery);

案例1

需求：开发一个插件，要求奇数行颜色是yellow，偶数行颜色是green，鼠标移到的行变为blue，移除变为原来的颜色；

案例2

实现选项卡功能

https://www.cnblogs.com/yuqingfamily/p/5813270.html