JQ笔记

[JQ笔记 1](#_Toc4838238)

[什么是JQuery? 3](#_Toc4838239)

[为什么要用jQuery？(非设计思想上的好处) 3](#_Toc4838240)

[jQuery学习注意点 3](#_Toc4838241)

[jQuery使用-引入 3](#_Toc4838242)

[jQuery使用-精髓 4](#_Toc4838243)

[jQuery使用-起步 4](#_Toc4838244)

[核心全局函数 4](#_Toc4838245)

[选择元素 4](#_Toc4838246)

[参数规则 4](#_Toc4838247)

[模写$()和.css方法 4](#_Toc4838248)

[jQuery实例方法-DOM操作 5](#_Toc4838249)

[进一步选择元素相关方法 5](#_Toc4838250)

[.get() 5](#_Toc4838251)

[.eq() 5](#_Toc4838252)

[.find() 6](#_Toc4838253)

[.filter() 6](#_Toc4838254)

[.not() 6](#_Toc4838255)

[.is() 7](#_Toc4838256)

[.has() 7](#_Toc4838257)

[取赋值相关方法 7](#_Toc4838258)

[.html() 7](#_Toc4838259)

[.val() 7](#_Toc4838260)

[.addClass 、.removeClass、.hasClass 7](#_Toc4838261)

[.attr() 7](#_Toc4838262)

[基于jQuery对象查改删增相关方法 7](#_Toc4838263)

[.next() 7](#_Toc4838264)

[.prev() 7](#_Toc4838265)

[.prevAll() 8](#_Toc4838266)

[.nextAll() 8](#_Toc4838267)

[.prevUntil() 8](#_Toc4838268)

[.nextUntil() 8](#_Toc4838269)

[.siblings() 8](#_Toc4838270)

[.parent() 9](#_Toc4838271)

[.parents() 9](#_Toc4838272)

[.offsetParent() 9](#_Toc4838273)

[.closest() 9](#_Toc4838274)

[.slice() 9](#_Toc4838275)

[.insertAfter()、.After() 9](#_Toc4838276)

[.insertBefore() 9](#_Toc4838277)

[.before() 9](#_Toc4838278)

[.appendTo()、.append() 9](#_Toc4838279)

[.prependTo()、.prepend() 9](#_Toc4838280)

[.remove()、.detach() 9](#_Toc4838281)

[.empty() 9](#_Toc4838282)

[.wrap() 9](#_Toc4838283)

[.wrapInner() 9](#_Toc4838284)

[.wrapAll() 9](#_Toc4838285)

[.unwrap() 9](#_Toc4838286)

[.clone() 9](#_Toc4838287)

[.data() 9](#_Toc4838288)

[jQuery对象事件绑定相关方法 11](#_Toc4838289)

[.on() 11](#_Toc4838290)

[.one() 11](#_Toc4838291)

[.off() 11](#_Toc4838292)

[.trigger() 11](#_Toc4838293)

[.hover() 11](#_Toc4838294)

[动画相关方法 12](#_Toc4838295)

[.hide() 12](#_Toc4838296)

[.show() 12](#_Toc4838297)

[.toggle() 12](#_Toc4838298)

[.fadeIn(淡入)、.fadeout(淡出) 、.fadeToggle、.fadeTo(淡入到) 12](#_Toc4838299)

[.slideDown(卷入)、.slideUp(卷出)、.slideToggle() 12](#_Toc4838300)

[.animate() 12](#_Toc4838301)

[.stop() 12](#_Toc4838302)

[.finish() 12](#_Toc4838303)

[.queue() 12](#_Toc4838304)

[.dequeue() 12](#_Toc4838305)

[.clearQueue() 13](#_Toc4838306)

[动画插件 13](#_Toc4838307)

[位置坐标图形大小相关方法 13](#_Toc4838308)

[.offset() 13](#_Toc4838309)

[.position() 13](#_Toc4838310)

[.scrollTop() 、.scrollLeft() 13](#_Toc4838311)

[.width()、.height() 13](#_Toc4838312)

[.innerWidth() 13](#_Toc4838313)

[.outerWidth() 13](#_Toc4838314)

[.innerHeight()、.outerHeight() 13](#_Toc4838315)

[遍历索引相关方法 13](#_Toc4838316)

[.each() 13](#_Toc4838317)

[.children() 13](#_Toc4838318)

[.index() 14](#_Toc4838319)

[jQuery工具方法: 14](#_Toc4838320)

[.type() 14](#_Toc4838321)

[.trim() 14](#_Toc4838322)

[.proxy() 14](#_Toc4838323)

[.noConflict() 14](#_Toc4838324)

[.each()、.map() 14](#_Toc4838325)

[.parseJSON() 14](#_Toc4838326)

[.makeArray() 14](#_Toc4838327)

[. $.extend() 插件扩展（工具方法） 15](#_Toc4838328)

[. $.fn.extend() 插件扩展（实例方法） 15](#_Toc4838329)

[$.ajax() 16](#_Toc4838330)

[$.Callbacks() 16](#_Toc4838331)

[$.Deferred() 17](#_Toc4838332)

[$.when() 19](#_Toc4838333)

## 什么是JQuery?

jQuery其实就是一堆的js函数(js库)，也是普通的js而已.

## 为什么要用jQuery？(非设计思想上的好处)

jQuery面向用户良好的设计使得在使用过程中彻底解放了你记忆原生操作DOM的接口。

jQuery中包含多个可重用的函数，用来辅助我们简化javascript开发。

jQuery在半数以上并没有复杂交互的网站中得以大面积使用，因为它们需要的仅仅是一些兼容低级浏览器而又呈现酷炫效果动画的页面。（jQuery出到3，但大公司pc端依然用1.x版本、移动端2.x版本）

jQuery改变了数百万人编写JavaScript的方式，当然部分人已经觉得时过境迁，组件化，工程化，大行其道，但请不要忘记他的前端开发者的启蒙意义！且很多公司很多项目依然需要他，所以笔面试必会！

## jQuery学习注意点

jQuery只是辅助工具，不能完全替代js，二者并存当方式出现在项目中

jQuery很庞杂，先学使用再学思想

jQuery方法很多，按需学习，把常用的有价值的学会

jQuery api 可以现查现用

## jQuery使用-引入

引入jQuery工具库

cdn：[http://www.jq22.com/cdn/#a2](http://www.jq22.com/cdn/)

下载地址：<http://www.jq22.com/jquery-info122>

官方地址（下载 api查询等）:

中文：

<https://www.jquery123.com/>

英文原版：

<https://jquery.com/>

## jQuery使用-精髓

选择元素

循环操作

链式调用

## jQuery使用-起步

核心全局函数：

$ (jQuery)

一顿操作猛如虎，全从$开始撸

撸代码从选择开始:

选择元素：

$(); 此函数可以传递多种参数，返回值是对象（jq对象）

参数规则：

css selector（css样式选择器）、 jquery unique selector（JQ特有选择器）

null、undefined（容错机制不报错）、dom（传入原生dom会包装成jquery对象）

$(function(){})= $(document).ready() 在DOMContentLoaded触发之后执行，在window.onload之前执行

css selector和context

<http://www.w3school.com.cn/jquery/jquery_ref_selectors.asp>

模写$()和.css方法：

(**function** () {  
 **function** jQuery(selector) {  
 **return new** jQuery.prototype.init(selector);  
 }  
 jQuery.prototype.init = **function** (selector) {  
 //选出dom对象并包装成jQuery对象  
 **let** dom = **null**;  
 **this**.length = 0;  
 **if** (selector.indexOf('.') !== -1) {  
 dom = document.getElementsByClassName(selector.slice(1));  
 } **else if** (selector.indexOf('#') !== -1) {  
 dom = document.getElementById(selector.slice(1));  
 }  
 **if** (dom.length === **undefined**) {  
 **this**[0] = dom;  
 **this**.length++;  
 } **else** {  
 //基础铺垫  
 **for** (**let** i = 0; i < dom.length; i++) {  
 **this**[i] = dom[i];  
 **this**.length++;  
 }  
 }  
 };  
 jQuery.prototype.css = **function** (config) {  
 //循环操作每一个dom  
 **for** (**let** i = 0; i < **this**.length; i++) {  
 **for** (**let** attr **in** config){  
 **this**[i].style[attr]=config[attr];  
 }  
 }  
 //jQuery 链式操作精髓  
 **return this**;  
 };  
 jQuery.prototype.init.prototype= jQuery.prototype;  
 window.$ = window.jQuery = jQuery;  
}());

**JQuert方法分为实例方法和工具方法。**

## jQuery实例方法-DOM操作

进一步选择元素相关方法：

.get():括号里可传索引，也可以传负数。得到一个原生的dom对象;不传参数或者传null、undefined返回原生dom数组。(将jQuery对象转换为原生对象)

jQuery.prototype.get = **function** (num) {  
 **return** num != **null** ? (num >= 0 ? **this**[num] : **this**[num + **this**.length]) : [].slice.call(**this**, 0);  
};

.eq():括号里可传索引，也可以传负数。得到一个jQuery对象

jQuery.prototype.init = **function** (selector) {  
 //选出dom对象并包装成jQuery对象  
 **let** dom = **null**;  
 **this**.length = 0;  
 **if** (selector === **null** || selector === **undefined**) {  
 **return this**;  
 }  
 **if** (**typeof** selector === 'string' && selector.indexOf('.') !== -1) {  
 dom = document.getElementsByClassName(selector.slice(1));  
 } **else if** (**typeof** selector === 'string' && selector.indexOf('#') !== -1) {  
 dom = document.getElementById(selector.slice(1));  
 }  
 **if** (selector **instanceof** Element || dom.length === **undefined**) {  
 **this**[0] = dom || selector;  
 **this**.length++;  
 } **else** {  
 //基础铺垫  
 **for** (**let** i = 0; i < dom.length; i++) {  
 **this**[i] = dom[i];  
 **this**.length++;  
 }  
 }  
};

jQuery.prototype.eq = **function** (num) {  
 **let** dom = num != **null** ? (num >= 0 ? **this**[num] : **this**[num + **this**.length]) :**null**;  
 **return** jQuery(dom)  
};

.find():查找的意思;在原有的基础上去往下查找。可传的参数和$()一样，非常常用。

.filter():过滤符合条件的元素，基于之前的进行限制,可传的参数和$()一样。传入方法第一个参数索引，第二个元素。

$('.demo').filter(**function** (index, ele) {  
 console.log(index, ele);  
 **return** index % 2 === 0;  
}).css({color:'#f40'});

.not():找到相反条件的元素

$('.demo').not(**function** (index, ele) {  
 console.log(index, ele);  
 **return** index % 2 === 0;  
}).css({color:'#f40'});

.is():is的参数是否和前面选出的元素是否有交集。

.has():也是筛选,后代元素必须有has参数的元素。

.add() 集中操作 .end() 回退操作

取赋值相关方法：

.html():jQuery是通过一个方法选出一堆元素进行操作,js选出一个组需要单独把组里的东西拿出来操作。取值就取第一个，赋值所有。赋值时参数可以传方法。赋值参数中的返回值决定了赋值的内容。

$('.warp>.demo').html(**function** (index, ele) {  
 **return** '<p style="color: #ff4400">' + (index+2) + '</p>';  
});

.text()、.size()

.val():操作的都是表单中元素的值，取值和.html相似

$('input').val(**function** (index, oldValue) {  
 **return** oldValue+index;  
});

.addClass 、.removeClass、.hasClass特别简单，使用频率极其高，也特别重要。

.css

.attr()=setAttribute setAttribute;.prop()基于原生jsDom对象的操作 ,特性才能映射。

给dom元素设置一些非特性的属性用attr，设置特性属性用prop。

注意：

1. 尽量避免直接添加行间样式，因为其权重过高 ，这样不利于响应式设计，破坏了c3+h5优雅的解决方案

2.attr和prop的区别：

jQuery认为：attribute的checked、selected、disabled就是表示该属性初始状态的值，property的checked、selected、disabled才表示该属性实时状态的值(值为true或false)

基于jQuery对象查改删增相关方法：

.next():获取选中元素的下一个兄弟元素节点,参数可以限制一下选中的兄弟节点.

.prev():获取选中元素的前一个兄弟元素节点.

.prevAll():获取选中元素前面所有兄弟元素节点.

.nextAll():获取选中元素后面所有兄弟元素节点.



.prevUntil():获取选中元素之前到参数为止的元素

.nextUntil():获取选中元素之后到参数为止的元素，例: nextUntil(‘h1’)选中元素到h1元素为止。有两个参数第一个到哪个元素为止，第二个更准确限制一下哪个元素为止。例如:netUntil(‘h1’,’input[type=”checkbox”]’);



.siblings():获取选中元素的所有兄弟节点，参数可以限制选中什么兄弟节点;做反选时经常用到siblings

.parent():返回选中元素上一级的父级

.parents():返回选中元素所有父级

.offsetParent():获取离选中元素最近有定位的父级。

.closest():获取离选中元素最近满足条件的父级。从自己开始找，如果自己符合条件也会被选中。

.slice():截取选中元素，前包后不包。

.insertAfter()、.After():将某元素插入选中元素之后

.insertBefore():将某元素插入选中元素之前,例如:$(‘content’). insertBefore(‘box’);都是剪切操作

.before():例: $(‘box’). before($(‘content’));将content插入box之前,before参数需要一个jQuery对象。

.appendTo()、.append():将元素添加到某元素里面最后，例: $(‘content’). appendTo (‘warp’);

.prependTo()、.prepend():将元素添加到某元素里面最前

.remove()、.detach():剪切元素,remove会清空方法和属性，detach不会清空

$() 参数：标签字符串 创建jQuery对象

.empty():清空所有子元素。

.wrap():给选中元素添加父级,参数可以是字符串、jQuery对象、function，当参数是已存在标签进行复制操作。

.wrapInner():给选中元内部添加父级。参数是方法时只有一个参数index

.wrapAll():给所有选中的元素，统一添加父级。封装插件时会用到。

.unwrap():去掉选中元素的直接父级。

.clone():克隆选中元素,参数传true可以克隆方法，用户操作某个dom时，需要复制另一份;例如:拖拽排序

.data():往jQuery对象里存信息、数据、状态;jQuery是映射操作比把属性放在行间或者dom身上,后期取值的时候比两种方法都简单;效率更高。

jQuery不操作dom,形成映射池，效率更高。用vue时,出现数据和结构相对应时，写到行间是更符合vue语法。

<div class='wrapper'>  
 <div class='tpl'>  
 <p></p>  
 <span></span>  
 <button>add</button>  
 </div>  
  
 <p class="show">  
 <span>sum</span>  
 <span class='sum'>0</span>  
 </p>  
</div>  
  
  
<script type="text/javascript">  
 **let** shopArr = [  
 {  
 name: 'james solider',  
 shopName: 'nike',  
 price: 110,  
 id: '1001'  
 }, {  
 name: 'Rose crazyLight',  
 shopName: 'adidas',  
 price: 90,  
 id: '2002'  
 }, {  
 name: 'curry one',  
 shopName: 'Under Armour',  
 price: 120,  
 id: '3003'  
 }  
 ];  
 shopArr.forEach(**function** (ele, index) {  
 **let** oCloneDom = $('.tpl').clone().removeClass('tpl');  
 oCloneDom.data({  
 shopName: ele.shopName,  
 id: ele.id,  
 price: ele.price,  
 })  
 .find('p')  
 .text(ele.name)  
 .next().text(ele.price);  
 oCloneDom.insertBefore($('.show'));  
 });  
 $('.wrapper button').click(**function** () {  
 $('.sum').text(+$('.sum').text()+$(**this**).parent().data('price'));  
 })  
</script>

jQuery对象事件绑定相关方法：

.on():实现任何事件绑定,有四个参数1.type(事件类型)2.selector()3.data()4handle(方法);如果没填第二个值,第三个值填了字符串会认为是第二个值，填其他类型不影响。可以在方法形参e.data上查看。第二个参数作用1.精确筛选元素，2事件委托

On事件绑定多个事件:

$('div').on({  
 click: **function** () {  
 console.log('click');  
 },  
 mouseenter: **function** () {  
 console.log('mouseenter');  
 },  
 mouseleave: **function** () {  
 console.log('mouseleave');  
 }  
});

.one():绑定事件只触发一次.

$('a').one('click',**function** () {  
 window.open('http://taobao.com');  
 **return false**;  
})

.off():移除对象所有事件,可传事件类型,移除指定事件。

.trigger():主动触发一系列事件（包括系统事件和自定义事件）,第一个参数事件类型,第二个参数实参数组。

.hover():移入移出事件集合,有两个参数，第一个参数代表移入处理函数，第二个代表移出处理函数。

.click/keydown/mouseenter…

兼容的事件对象：

e.pageX、e.clienX、e.which、e.button

e.preventDefault()阻止默认事件

e.stopPropagation()阻止冒泡

return false; 兼容阻止事件冒泡和默认事件

动画相关方法：

.hide():隐藏,将display变为none;

.show():显示,将display:none调用show时还原状态，有两种情况;

1.手动设置display:none状态，调用show会还原成默认状态;

2.当设置display:inline-block或者其它状态，调用hide变为none时,再调用show时不会还原成默认的,会还原成设置的状态。

.toggle():集合方法，内部有奇偶判断

参数：null 或 （duration(过度时间), easing(过度方式), callblack）

.fadeIn(淡入)、.fadeout(淡出) 、.fadeToggle、.fadeTo(淡入到):

参数：null或 （duration, [opacity], easing, callblack）

.slideDown(卷入)、.slideUp(卷出)、.slideToggle():

参数：null或 （duration, easing, callblack）

.animate():参数：(target(目标点) duration(过渡时间) easing(运动方式) callback);

.stop():和animate配合使用; 不传参数会阻止当前运动,开启下一个运动;传一个true在当前运动停止，后面一切运动都停止;传两个true,停止当前运动并且移动到当前运动目标点

.finish():和animate配合使用,直接完成所有运动，到达目标点。

.delay():延迟多少毫秒执行下一个动画.(加在两个动画之间)

jQuery.fx.off = true 运动的开关 可以关闭过滤动画。

.queue():创建一个队列，并起一个名字，有两个参数type(名字),content(内容);内部是一个数组

.dequeue():出队列,参数队列名;全部出列的写法

$('.demo').queue('chain',**function** (next) {  
 console.log('1');  
 next();  
}).queue('chain',**function** (next) {  
 console.log('2');  
 next();  
}).queue('chain',**function** (next) {  
 console.log('3');  
});

.clearQueue():清空队列。

动画插件:

插件名称：

jQuery Easing Plugin:

目的:

用于扩展jQuery动画过渡效果

链接地址：

<https://cdnjs.cloudflare.com/ajax/libs/jquery-easing/1.4.1/jquery.easing.min.js>

位置坐标图形大小相关方法：

.offset():获取对象相对于文档位置，不考虑父级定位，返回一个对象有两个属性left和top。

.position():获取相对于父级定位的位置，没有有定位的父级就相对于文档进行定位。

.scrollTop() 、.scrollLeft():获取滚动条的距离，也可以赋值，参数只能是数字。

.width()、.height():获取内容区宽和高;

.innerWidth():content区+padding

.outerWidth():content区+padding+border,

如果参数传true,获取content区+padding+border+margin

.innerHeight()、.outerHeight():同innerWidth和outerWidth同理。

遍历索引相关方法：

.each():类似数组的foreach方法。

$('li').each(**function** (index,ele) {  
 $(ele).text(index).css('color','#f40');  
})

.children():获取所有子元素。

$('div').children().each(**function** (index, ele) {  
 ele.nodeName === 'I' ? $(ele).text('i-' + index) : $(ele).text('span-' + index);  
})

.index():获取在兄弟中的索引,传参数可以看在某一类兄弟元素中的位置

$('ul').on('click','li',**function** (e) {  
 console.log($(e.target).index())  
})

.index带参数的例子:

$('div').children().each(**function** (index, ele) {  
 ele.nodeName === 'I' ? $(ele).text('i-' + index) : $(ele).text('span-' + index);  
 $(ele).on('click',**function** (e) {  
 console.log($('div span').index($(e.target)));  
 });  
});

## jQuery工具方法

.type():判断数据类型 $.isArray() $.isFunction() $.isWindow…

typeof()与tyoe的区别：typeof()里面的数据是什么类型，type里面的数据到底是什么

.trim():消除空格

.proxy():改变this指向,返回一个对象;两个参数:第一个要改变的对象,第二个this指向的对象。应用比较多

.noConflict():防止冲突,变量命名冲突$

.each()、.map():循环处理数组、类数组、对象

.parseJSON():严格json字符串转换成对象 – 原生JSON.parse();

.makeArray():类数组转换成数组,可以填两个参数第一个任意类型，第二个类数组，会把第一个值添加到类数组中,底层是一个拼接,将第一个参数拼接到第二个参数里去，返回类型根据第二个参数类型决定。

将对象变为数组的方法:1.[].slice.call(ogj,0);

2.在对象上定义slice:Arrary.prototype.slice属性，然后对象调用slice属性转换为数组。

. $.extend() 插件扩展（工具方法）:可以完成几个功能:

1. 扩展方法\*\*\*\*api没有

把自己定义的方法加到jQuery工具方法里面

$.extend({  
 //自定义生成随机数 start开始位置，final截止位置  
 defineMandom: **function** (start, final) {  
 // [0,1)\*len+start  
 **let** len=final-start;  
 **return** Math.random()\*len+start;  
 }  
})

1. 浅层克隆

$.extend===$.fn.extend,将第二参数的属性浅层克隆到第一个参数里;第一个参数里没有的,第二个参数拿自身的补上;第一个有的属性,第二个也有的属性会覆盖第一个属性的值。参数可以无限个从参数二开始都是把参数里的属性放到参数一里去,有就覆盖，没有就添加。

$.extend(obj,obj2);

1. 深层克隆

第一个参数变为true，剩下的规则和浅层克隆一样。和浅层克隆的区别是引用值是否在一个索引上。

$.extend(**true**,obj,obj2);

. $.fn.extend() 插件扩展（实例方法）:可以完成几个功能:

1. 扩展方法\*\*\*\*api没有

把自己定义的方法加到jQuery实例方法里面

$.fn.extend({  
 drag: **function** () {  
 **let** disX, disY, self = **this**;  
 $(**this**).on('mousedown', **function** (e) {  
 disX = e.pageX - $(**this**).offset().left;  
 disY = e.pageY - $(**this**).offset().top;  
 $(document).on('mousemove', **function** (e) {  
 $(self).css({left: e.pageX - disX, top: e.pageY - disY});  
 }).on('mouseup', **function** (e) {  
 $(**this**).off('mousemove').off('mouseup');  
 });  
 });  
 **return this**;  
 }  
});

1. 浅层克隆:同$.extend()一样.
2. 深层克隆:同$.extend()一样.

高级方法

$.ajax():,前端向后端发送请求，获得后端传过来的数据应用的工具。

参数只有一个对象，对象上有一些必须的属性:

1. url:地址 必填
2. type:请求方式 必填
3. data:参数 值可以是字符串或者对象
4. success:请求成功后处理函数,函数有一个形参res
5. error:请求失败处理函数,返回一个错误对象e，status、statusText两个属性比较有用。
6. complete:请求完成的处理函数,最后直接
7. context:改变函数上下文;例:context:$(‘.demo’)
8. timeout:请求时间。
9. asyns:true或false 是否异步
10. dataType:’jsonp’-----跨域类型;平时不用填,期待服务器返回数据类型

$.ajax({  
 url: 'https://easy-mock.com/mock/5c0b4a876162b83fe0a50cb9/person',  
 type: 'get',  
 success: **function** (res) {  
 // console.log(res);  
 $.each(res.data, **function** (index, ele) {  
 console.log(ele.name);  
 })  
 }  
})

$.ajax的返回值是deferred（延迟对象）。

异步编程离不开回调机制。

Js单线程→异步编程优化体验 防止阻塞页面→回调函数

$.Callbacks():异步回调;前提：看一下，《你不知道的js》课程中的UI多线程-深入剖析js执行机制. 返回一个回调对象(cb=$Callbacks)与cb.add()和cb.fire()配合使用。

参数可以填once(只执行一次) 最常用

memor(具有记忆功能)只要执行过fire后面再加add也会执行之前的fire. 最常用

unique(将回调方法去重)

stopOnFallse碰到回调函数返回false时,无论后面有没有回调函数都不执行了。

回调地狱:可用性、可阅读性、可维护性、可迭代性、拓展性差

设计模式六大原则:单一职责原则(一个函数只做一件事)

开闭原则（对外拓展是开放的，在外部留出一个方法，调用该方法可以加一些功能）

例一:单纯添加回调函数,然后执行。调用几次fire就执行几次回调函数。

**function** a (ms) {  
 console.log('a', ms);  
}  
**function** b (ms) {  
 console.log('b', ms);  
}  
**let** cb = $.myCallBacks();  
cb.add(a);  
cb.add(b);  
cb.fire('null');  
cb.fire('null');

例二:传入参数once,无论后面调用多少次fire,回调函数只执行一次。

**let** cbOnce = $.myCallBacks('once');  
cbOnce.add(a);  
cbOnce.add(b);  
cbOnce.fire('once');  
cbOnce.fire('once');

例三:传入参数memory,具有记忆功能,调用一次fire,执行一次之前添加的所有回调函数。

**let** cbMemory = $.myCallBacks('memory');  
cbMemory.add(a);  
cbMemory.add(b);  
cbMemory.fire('memory');  
cbMemory.add(c);  
cbMemory.fire('memory');

例四:传入参数once memory,具有记忆功能,调用fire,只执行一次之前添加的所有回调函数。

**let** cbOnceMemory = $.myCallBacks('once memory');  
cbOnceMemory.add(a);  
cbOnceMemory.add(b);  
cbOnceMemory.fire('OnceMemory');  
cbOnceMemory.add(c);  
cbOnceMemory.fire('OnceMemory');

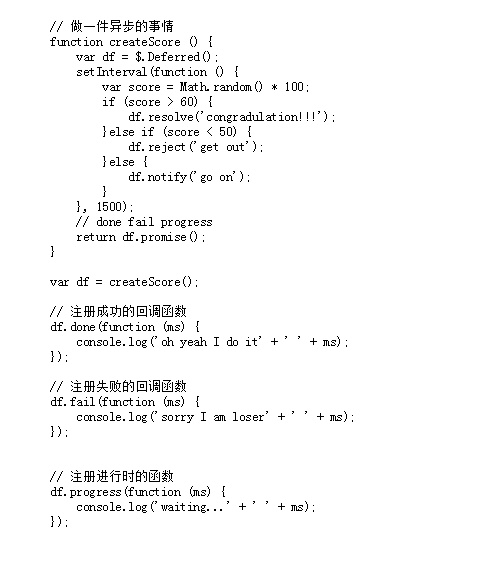
$.Deferred():异步也是完成回调操作;根据不同状态，都有对应的方式触发回调。

内部划分三种状态: resolve触发done（成功）回调;

reject 触发fail(失败) 回调

notify 触发progress (正在进行) 回调

df.promise():返回一个阉割版的deferred对象。



df.then是核心方法，返回的是阉割版的deferred对象。

作用:1.方法简化注册

2.可以接力式的去处理(根据上一个then返回的值,下一个then可以进行操作)

**function** createScore () {  
 **let** df = $.Deferred();  
 setInterval(**function** () {  
 **let** score = Math.random() \* 100;  
 **if** (score > 60) {  
 df.resolve('congradulation!!!');  
 }**else if** (score < 50) {  
 df.reject('get out');  
 }**else** {  
 df.notify('go on');  
 }  
 }, 1500);  
 **return** df.promise();  
}  
**let** df = createScore();  
df.then(**function** (ms) {  
 console.log('oh Yeah!' + ' ' + ms);  
 **return** 'ok';  
}, **function** (ms) {  
 console.log('oh No!' + ' ' + ms);  
 **return** 'no';  
}, **function** () {  
 console.log('what?' + ' ' + ms);  
 **return** 'go on';  
}).then(**function** (param) {  
 console.log('泡妞' + param);  
}, **function** (param) {  
 console.log(param + '脸');  
}, **function** (param) {  
 console.log(param + '加油');  
});

$.when():可以传一到多个延迟deferred对象，返回一个promise对象，然后调用.then方法所有参数都触发成功才会触发成功回调函数，有一个失败调用失败调用函数。