此MD/PDF/HTML文档由 @竹梦者也 编辑制作，不可贩卖✕。仅供学习交流使用，下载后切勿随意传播。转载麻烦加上出处~

欢迎在留言区提供修改建议~

本文内容和代码来自于Pink老师的JQ视频教程

2021.6.22　@竹梦者也(https://www.zjgsuzjx.top)

一、jQuery入门

一、jQuery概述

1. JavaScript库

JavaScript库：即library，是一个封装好的特定的集合（方法和函数）。从封装一大堆函数的角度理解库，就是在这个库中，封装了很多预先定义好的函数在里面，比如动画animate、hide、show，比如获取元素等。

简单理解：就是一个JS文件，里面对我们原生JS代码进行了封装，存放到里面。这样我们可以快速高效的使用这些封装好的功能了。

> 比如jQuery，就是为了快速方便的操作DOM，里面基本都是函数（方法）。

常见的JavaScript库：

- jQuery
- Prototype
- YUI
- Dojo
- ExtJS
- 移动端的zepto

> 这些库都是对原生JavaScript的封装，内部都是用JavaScript实现的，我们主要学习的是jQuery。

2. jQuery的概念

jQuery是一个快速、简洁的JavaScript库，其设计的宗旨是“write Less，Do More”，即倡导写更少的代码，做更多的事情。

j就是JavaScript；Query查询；意思就是查询js，把js中的DOM操作做了封装，我们可以快速的查询使用里面的功能。

jQuery封装了JavaScript常用的功能代码，优化了DOM操作、事件处理、动画设计和Ajax交互。

> 学习jQuery本质：就是学习调用这些函数（方法）。

jQuery出现的目的是加快前端人员的开发速度，我们可以非常方便的调用和使用它，从而提高开发效率。

3. jQuery的优点

- 轻量级。核心文件才几十kb，不会影响页面加载速度
- 跨浏览器兼容。基本兼容了现在主流的浏览器
- 链式编程、隐式迭代
- 对事件、样式、动画支持，大大简化了DOM操作
- 支持插件扩展开发。有着丰富的第三方的插件，例口：树形菜单、日期控件、轮播图等
- 免费、开源

二、jQuery的基本使用

1. jQuery的下载

https://jquery.com

2. jQuery的使用步骤

引入jQuery文件、使用即可。

3. jQuery的入口函数

```
// 第一种
$(document).ready(function () {
    $('div').hide();
})
// 第二种
$(function () {
    $('div').hide();
})
```

4. jQuery的顶级对象$

$ 是jQuery的别称，在代码中可以使用jQuery代替$，但一般为了方便，通常都直接使用$。

$ 是jQuery的顶级对象，相当于原生JavaScript中的window。把元素利用 $ 包装成Query对象，就可以调用jQuery的方法。

5. jQuery 对象和DOM对象

用原生JS获取来的对象就是DOM对象

jQuery方法获取的元素就是jQuery对象。

jQuery对象本质是：利用对DOM对象包装后产生的对象（伪数组形式存储）。

> jQuery对象只能使用jQuery方法，DOM对象则使用原生的Javascirpt属性和方法

DOM对象与jQuery对象之间是可以相互转换的。

因为原生js比jQuery更大，原生的一些属性和方法jQuery没有给我们封装，要想使用这些属性和方法需要把jQuery对象转换为DOM对象才能使用。

DOM对象转换为jQuery对象：

```
var myvideo=document.querySelector('video');
$('myvideo');
```

jQuery对象转换为DOM对象：

```
// 两种方式
$('video')[0].play();
$('video').get(0).play();
```

二、jQuery 常用API

一、jQuery 选择器

1. jQuery 基础选择器

原生JS获取元素方式很多，很杂，而且兼容性情况不一致，因此jQuery给我们做了封装，使获取元素统一标准。

名称	用法	描述
ID选择器	$("#id")	获取指定ID的元素
全选选择器	$('*')	匹配所有元素
类选择器	$(".class")	获取同一类class的元素
标签选择器	$("div")	获取同一类标签的所有元素
并集选择器	$("div,p,li")	选取多个元素
交集选择器	$("li.current")	交集元素

```
$(function () {
    $(".nav"); // 单引号和双引号都可以
})
```

2. jQuery 层级选择器

名称	用法	描述
子代选择器	`$("ul>li");`	使用>号，获取亲儿子层级的元素；注意，并不会获取孙子层级的元素
后代选择器	`$("ul li");`	使用空格，代表后代选择器，获取ul下的所有li元素，包括孙子等

```
$(function () {
    $("ul li"); // 和CSS一样
})
```

3. 隐式迭代

遍历内部DOM元素（伪数组形式存储）的过程就叫做隐式迭代。

> 简单理解：给匹配到的所有元素进行循环遍历，执行相应的方法，而不用我们再进行循环，简化我们的操作，方便我们调用。

```
$("div").css("background", "pink");
$("ul li").css("color", "red");
```

4. jQuery 筛选选择器

语法	用法	描述
:first	$('li:first')	获取第一个li元素
:last	$('li:last')	获取最后一个li元素
:eq(index)	$("li:eq(2)")	获取到的li元素中，选择索引号为2的元素，索引号index从0开始。
:odd	$("li:odd")	获取到的li元素中，选择索引号为奇数的元素
:even	$("li:even")	获取到的li元素中，选择索引号为偶数的元素

```
$(function () {
    $("ul li:first").css("color", "red");
    $("ul li:eq(2)").css("color", "red");
    $("ul li:odd").css("color", "red");
    $("ul li:even").css("color", "red");
})
```

5. jQuery 筛选方法

语法	用法	说明
`parent()`	`$("li").parent();`	查找父级
`children(selector)`	`$("ul").children("li")`	相当于`$("ul>li")`，最近一级（亲儿子）
`find(selector)`	`$("ul").find("li");`	相当于$("ul li"),后代选择器
`siblings(selector)`	`$(".first").siblings("li");`	查找兄弟节点，不包括自己本身
`nextAll([expr])`	`$(".first").nextAll()`	查找当前元素之后所有的同辈元素
`prevtAll([expr])`	`$(".last").prevAll()`	查找当前元素之前所有的同辈元素
`hasClass(class)`	`$('div').hasClass("protected")`	检查当前的元素是否含有某个特定的类，如果有，则返回true
`eq(index)`	`$("li").eq(2);`	相当于`$("li:eq(2)")`,`index`从0开始

```
// 父亲 返回最近一级的父亲元素
$(".son").parent();
// 只选亲儿子
$(".nav").children("p").css("color", "red");
// 选所有的后代 类似于后代选择器
$(".nav").find("p").css("color", "red");
```

仿微博下拉菜单：

```
$(function () {
    // 鼠标经过
    $(".nav>li").onmouseover(function () {
        // this表示当前元素
        $(this).children("ul").show();
    });
    // 鼠标离开
    $(".nav>li").onmouseout(function () {
        $(this).children("ul").hide();
    })
})
```

其它方法：

```
// 除了自身元素之外的所有兄弟元素
$("ol .item").siblings("li").css("color", "red");
// 选择第几个子元素 由两种方法 推荐第二种
$("ul li:eq(2)").css("color", "red");
$("ul li").eq(2).css("color", "red");
// 判断是否由某个类名 有就返回true
$("div:first").hasClass("current");
```

> 重点记住：parent children find siblings eq

6. jQuery 里面的排他思想

想要多选一的效果，排他思想：当前元素设置样式，其余的兄弟元素清除样式。

```
//隐式迭代
$("button").click(function () {
    // 当前元素变化颜色
    $(this).css("background", "pink");
    // 其它兄弟去掉背景颜色
    $(this).siblings("button").css("background", "");
})
```

淘宝服饰精品案例分析：

```
// 鼠标经过左侧的小li
$("#left li").onmouseover(function () {
    // 得到当前li的索引号
    var index=$(this).index();
    // 对应索引号显示
    $("#content div").eq(index).show();
    // 让其他图片隐藏
    $("#content div").eq(index).siblings().hide();
})
```

> 也是用排他思想

7. 链式编程

上面案例的代码其实可以简化：

```
$(function () {
    // 链式编程
    $("button").click(function () {
        $(this).css("color", "red").siblings().css("color", "")
    })
})
```

> 使用链式编程一定注意是哪个对象执行样式

二、jQuery 样式操作

1. 操作css方法

jQuery可以使用css方法来修改简单元素样式；也可以操作类，修改多个样式。

- 参数只写属性名，则是返回属性值：

```
console.log($("div").css("width"))
```

- 参数是属性名，属性值，逗号分隔，是设置一组样式，属性必须加引号，值如果是数字可以不用跟单位和引号：

```
$("div").css("width", 300);
```

- 参数可以是对象形式，方便设置多组样式。属性名和属性值用冒号隔开，属性可以不用加引号：

```
$("div").css({
    width:400,
    height:300,
    backgroundColor:"red"
})
```

> 如果是复合属性则必须采取驼峰命名法，如果值不是数字，则需要加引号

2. 设置类样式方法

作用等同于以前的classList，可以操作类样式，注意操作类里面的参数不要加点。

```
$("div").click(function () {
    // 添加类
    $(this).addClass("current");
})
$("div").click(function () {
    // 删除类
    $(this).removeClass("current");
})
$("div").click(function () {
    // 切换类 有类就移除，没类就加类
    $(this).toggleClass("current");
})
```

Tab切换栏案例：

```
$(function () {
    // 点击上部的li 当前li添加类，其余移除类
    $(".tab_list li").click(function () {
        // 链式编程操作
        $(this).addClass("current").siblings().removeClass("current");
        // 点击的同时获得索引号
        var index=$("this").index();
        // 相应索引号的item显示 其它隐藏
        $(".tab_list item").eq(index).show().siblings().hide();
    })
})
```

3. 类操作与className区别

原生JS中className会覆盖元素原先里面的类名。

jQuery里面类操作只是对指定类进行操作，不影响原先的类名。

三、jQuery效果

jQuery给我们封装了很多动画效果，最为常见的如下：

1. 显示隐藏效果

语法：

```
show([speed,[easing],[fn]])
```

显示参数：

（1）参数都可以省略，无动画直接显示。
（2）speed：三种预定速度之一的字符串（“slow”，“normal"，or“fast'）或表示动画时长的毫秒数值（如：1000）。
（3）easing：（Optional）用来指定切换效果，默认是“swing"，可用参数“linear"。

（4）fn：回调函数，在动画完成时执行的函数，每个元素执行一次。

手册：jQuery API 中文文档

例子：

```
$("button").eq(0).click(function () {
    // 显示
    $("div").show(1000, function () {
        alert(1);
    });
})
$("button").eq(1).click(function () {
    // 隐藏
    $("div").hide(1000, function () {
        alert(2);
    });
})
$("button").eq(2).click(function () {
```

```
    // 切换
    $("div").toggle(1000);
})
```

2. 滑动效果

```
$("button").eq(0).click(function () {
    // 下滑动
    $("div").slideDown();
})
$("button").eq(1).click(function () {
    // 上滑动
    $("div").slideUp();
})
$("button").eq(2).click(function () {
    // 滑动切换
    $("div").slideToggle();
})
```

3. 事件切换

```
hover([over,]out);
```

（1）over：鼠标移到元素上要触发的函数（相当于mouseenter）
（2）out：鼠标移出元素要触发的函数（相当于mouseleave）

导航栏下拉菜单优化：

```
// 事件切换效果
$(".nav>li").hover(function () {
    $(this).children("ul").slideDown(200);
},function () {
    $(this).children("ul").slideUp(200);
})
// 可以再优化
$(".nav>li").hover(function () {
    $(this).children("ul").slideToggle();
})
```

4. 动画队列及其停止排队方法

动画或效果队列

动画或者效果一旦触发就会执行，如果多次触发，就造成多个动画或者效果排队执行。

停止排队

（1）stop()方法用于停止动画或效果。

> （2）注意：stop()写到动画或者效果的前面，相当于停止结束上一次的动画。

```
$(".nav>li").hover(function () {
    // stop一定要写到动画前面
    $(this).children("ul").stop().slideToggle();
})
```

5. 淡入淡出效果

淡入效果语法规范

```
fadeIn([speed],[easing],[fn])
```

淡入淡出切换效果参数

（1）参数都可以省略。

（2）speed：三种预定速度之一的符串（“slow"，“normal"，or“fast”）或表示动画时长的毫秒数值（如：1000）。

（3）easing：（Optional）用来指定切换效果，默认是“swing"，可用参数“linear"。

（4）fn：回调函数，在动画完成时执行的函数，每个元素执行一次。

```
$("button").eq(1).click(function () {
    // 淡入
    $("div").fadeIn(100);
})
$("button").eq(2).click(function () {
    // 淡出
    $("div").fadeOut(100);
})
$("button").eq(3).click(function () {
    // 淡入淡出切换
    $("div").fadeToggle(100);
})
```

渐进方式调整到指定的不透明度

```
$("button").eq(2).click(function () {
    // 修改透明度 第一个是速度，第二个是透明度
    $("div").fadeTo(1000, 0.5);
})
```

切换透明度案例：

```
$(".wrap li").hover(function () {
    // 鼠标经过
    $(this).siblings().stop().fadeTo(400, 0.5);
}, function () {
    // 鼠标离开
    $(this).siblings().stop().fadeTo(400, 1);
})
```

6. 自定义动画 animate

```
animate(params,[speed],[easing],[fn])
```

params:一组包含作为动画属性和终值的样式属性和及其值的集合

speed:三种预定速度之一的字符串("slow","normal", or "fast")或表示动画时长的毫秒数值(如：1000)

easing:要使用的擦除效果的名称(需要插件支持).默认jQuery提供"linear" 和 "swing".

fn:在动画完成时执行的函数，每个元素执行一次。

```
$("button").click(function () {
    $("div").animate({
        left: 200,
        top: 150,
        opacity: .4,
        width: 500
    }, 500)
})
```

> 注意div要加定位

王者荣耀手风琴效果案例：

```
// ①鼠标经过某个i有两步操作：
// ②当前小i宽度变为224px，同时里面的小图片淡出，大图片炎入
// ③其余兄弟小i宽度变为69px，小图片淡入，大图片淡出
$(".king li").mouseenter(function () {
    $(this).stop().animate({
        width: 224
    } ,200).find(".small").stop().fadeOut().siblings("big").stop().fadeIn();
    $(this).siblings("li").stop().animate({
        width:69
    }).find(".samll").stop().fadeIn().siblings(".big").stop().fadeOut();
})
```

四、jQuery 属性操作

1. 设置或获取元素固有属性值 prop

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

```
console.log($("a").prop("href"));
$("a").prop("title", "xxx");
```

2. 设置或获取元素自定义属性值 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"。

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

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。

```
// 数据缓存 存放到内存里
$("span").data("uname", "zjgsuzjx");
// 获取内存中的值
console.log($("span").data("uname"));
// 获取h5的自定义属性 不需要加data-前缀 返回数字型
console.log($("div").data("index"));
```

购物车复选框案例：

```
$(".checkall").change(function () {
    // 获取全选的状态 来给其它框对应赋值 另外一个全选按钮也要变化
    $(".j-checkbox, .checkall").prop("checked", $(this).prop("checked"));
})
$(".j-checkbox").change(function () {
    // 被选中的复选框个数===总共的复选框的个数
    if($(".j-checkbox:checked").length === $(".j-checkbox").length){
        $(".checkall").prop("checked", true);
    }else{
        $(".checkall").prop("checked", false);
    }
})
```

五、jQuery内容文本值

主要针对元素的内容还有表单的值操作。

普通元素内容html（相当于原生innerHTML）：

```
// 获取设置元素内容 html()
console.log($("div").html());
$("div").html("123");
```

普通元素文本内容text（相当与原生innerText）：

```
// 获取设置元素文本内容 text()
console.log($("div").text());
$("div").text("123");
```

表单的值val（相当于原生value）：

```
// 获取设置表单值
console.log($("input").val());
$("input").val("123");
```

购物车增减商品数量案例：

```
// 修改数量加减
$(".increment").click(function () {
    // 得到当前兄弟文本框的值
    var n = $(this).siblings(".itxt").val();
    n++;
    $(this).siblings(".itxt").val();
    // 当前商品的价格 要修改当前商品的额度
    var p = $("this").parent().parent().siblings(".p-price").html();
    p = p.substr(1);
    // 保留两位小数 toFixed()
    var price = (p * n).toFixed(2);
    $("this").parent().parent().siblings(".p-sum").html("￥" + price);
})
$(".decrement").click(function () {
    var n = $(this).siblings(".itxt").val();
    if (n === 1) {
        return false;
    }
    n--;
    $(this).siblings(".itxt").val();
    // 当前商品的价格 要修改当前商品的额度
    // var p = $("this").parent().parent().siblings(".p-price").html();
    // 更好的写法 parents() 返回指定的祖先
```

```
    var p = $("this").parents(".p-sum").siblings(".p-price").html();
    p = p.substr(1);
    $("this").parent().parent().siblings(".p-sum").html("￥" + p * n);
})
// 用户修改文本框的值 值也要变化
$(".itxt").change(function () {
    var n = $(this).val();
    var p = $("this").parents(".p-sum").siblings(".p-price").html();
    p = p.substr(1);
    $("this").parent().parent().siblings(".p-sum").html("￥" + (p *
n).toFixed(2));
})
```

六、jQuery 元素操作

主要是遍历、创建、添加、删除元素操作。

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 。

第一种 方法：

```
var arr=["red", "blue", "yellow"]
$("div").each(function (index, domEle) {
    // 这是索引，相当于for循环的i
    console.log(index);
    // 这是DOM对象
    console.log(domEle);
    // 转换为jq对象
    $(domEle).css("color" ,arr[index]);
})
```

each方法遍历匹配的每一个元素。主要用DOM处理。

里面的回调函数有2个参数：index是每个元素的索引号；demEle是每个DOM元素对象，不是jquery对象

> 所以要想使用jquery方法，需要给这个dom元素转换为jquery对象$（domEle）

还有第二种方法遍历：

```
// 可以用来处理数据 可以遍历对象等
$.each(arr, function (i, ele) {
    console.log(i);
    console.log(ele);
})
```

> 第一种方法适合用户遍历DOM，第二种方法适合用于遍历数据。

购物车总金额计算案例：

```
$(function () {
    function getSum() {
        var count = 0;
        var money = 0;
        $(".itxt").each(function (i, ele) {
            count += parseInt($(ele).val);
        })
        $(".amout em").text(count);
        $(".p-sum").each(function (i, ele) {
            money += parseFloat($(ele).text().substr(1));
        })
        $(".price-sum em").text("￥" + money.toFixed(2));
    }
})
```

2. 创建元素

```
// 创建元素
var li = $("<li>我是后来的</li>>");
```

3. 添加元素

```
// 添加元素 内部添加并且放到最后面
$("ul").append(li);
// 添加元素 内部添加并且放到最前面
$("ul").prepend(li);
var div=$("<div>hello</div>");
// 外部添加
$(".text").after(div);
$(".text").before(div);
```

> 内部添加元素，生成之后，它们是父子关系。外部添加元素，生成之后，他们是兄弟关系。

4. 删除元素

```
// 删除元素 删除匹配的元素本身
$("ul").remove();
// 删除匹配元素的子节点
$("ul").empty();
// 清空匹配元素里的内容
$("ul").html("");
```

购物车删除商品模块案例：

```
$(".p-action a").click(function () {
    // 删除的是当前商品
    $(this).parents(".cart-item").remove();
    // 删除之后要重新计算一次数量和总金额
    getSum();
})
$(".remove-batch").click(function () {
    // 删除的是复选框被选择的商品
    $(".j-checkbox:checked").parents(".cart-item").remove();
})
$(".clear-all").click(function () {
    // 删除所有商品
    $(".cart-item").remove();
})
```

购物车选中商品添加背景案例：

```
// 点全选按钮的情况
if($(this).prop("checked")){
    $(".cart-item").addClass("check-cart-item");
}else{
    $(".cart-item").removeClass("check-cart-item");
}
```

七、jQuery尺寸、位置操作

1. jQuery尺寸

语法	用法
width() / height()	取得匹配元素宽度和高度值 只算 width / height
innerWidth() / innerHieght()	取得匹配元素宽度和高度值 包含 padding
outerWidth() / outerHeight()	取得匹配元素宽度和高度值 包含 padding 、border
outerWidth(true) / outerHeight(true)	取得匹配元素宽度和高度值 包含 padding 、borde、margin

```
// width和height获取元素的对应的值
console.log($("div").width());
// innerWidth/innerHeight 获取元素含padding的大小
console.log($("div").innerWidth());
// outerWidth/outerHeight 获取元素+padding+border的大小
console.log($("div").outerWidth());
// outerWidth(true)/outerHeight(true)获取元素+padding+border+margin的大小
console.log($("div").outerWidth(true));
```

- 以上参数为空，则是获取相应值，返回的是数字型
- 如果参数为数字，则是修改相应值。
- 参数可以不必写单位。

2. jQuery位置

位置主要有三个：offset、position、scrollTop/scrolleft

offset设置或获取元素偏移

① offset0方法设置或返回被选元素相对于文档的偏移坐标，跟父级没有关系。

② 该方法有2个属性left、top。offset0.top用于获取距离文档顶部的距离，offset0.left用于获取距离文档左侧的距离。

③ 可以设置元素的偏移：offset({top: 10,left: 30})；

```
// 获取元素距离文档的位置
console.log($(".son").offset().left);
// 修改值
$(".son").offset({
    top: 200,
    left: 200
})
```

position获取元素偏移

position方法用于返回被选元素相对于带有定位的父级偏移坐标，如果父级都没有定位，则以文档为准。

```
// 获取距离带有定位父级位置 没有则为文档为准
// 只能获取，不能设置偏移
console.log($(".son").position());
```

scrollTop/scrollLeft设置或获取元素被卷去的头部和左侧

scrollTop方法设置或返回被选元素被卷去的头部。

```
// 打印被卷去的值
console.log($(document).scrollTop());
// 也可以设置一打开就滚动到指定位置
$(document).scrollTop(100);
```

带有动画的返回顶部：

```
var boxtop = $(".container").offset().top;
$(window).scroll(function () {
    if ($(document).scrollTop >= boxtop) {
        $(".back").fadeIn();
    } else {
        $(".back").fadeOut();
    }
})
// 带有动画的返回顶部
$(".back").click(function () {
    $("body,html").stop().animate({
        scrollTop:0
```

```
    });
})
```

电梯导航案例：

```
var toolTop = $("recommend").offset().top;
// 使用互斥锁/节流阀来解决小bug
var flag = true;
toggleTool();
// 封装函数
function toggleTool() {
    if ($(document).scrollTop >= toolTop) {
        $(".fixedtool").fadeIn();
    } else {
        $(".fixedtool").fadeOut();
    }
}
$(window).scroll(function () {
    toggleTool();
    // 页面滚动的某个内容区域，li也会发生相应的变化
    $(".floor .w").each(function (i, ele) {
        if ($(document).scrollTop() >= $(ele).offset().top) {
            $(".fixedtool li").eq(i).addClass("current").siblings().removeClass();
        }
    })
})
// 点击电梯导航页面可以滚动到相应的页面
$(".fixedtool li").click(function () {
    // 关闭阀
    flag = false;
    // 选出对应索引号的内容区的盒子
    var current = $(".floor .w").eq($(this).index()).offset().top;
    // 页面动画滚动效果
    $("body,html").stop().animate({
        scrollTop: current;
    }, function () {
        // 动画执行完之后再打开阀
        flag = true;
    });
    // 点击之后，让当前小li 添加current类名
    $(this).addClass("current").siblings().removeClass();
```

```
})
```

三、jQuery 事件

一、jQuery事件注册

单个事件注册

语法：

```
$("div").click(function () {
    // 事件处理程序
})
```

其他事件和原生基本一致。

比如mouseover、mouseout、blur、focus、change、keydown、keyup、resize、scroll等

二、jQuery 事件处理

1. 事件处理on绑定事件

> on方法在匹配元素上绑定一个或多个事件的事件处理函数。

```
$("div").on({
    mouseenter: function () {
        $(this).css("background","skyblue");
    },
    click:function () {
        $(this).css("background","red")
    },
    mouseleave: function () {
        $(this).css("background","yellow")
    }
})
```

```
on(events,[selector],[data],fn)
```

events：一个或多个用空格分隔的事件类型，如“click"或“keydown"。

selector：元素的子元素选择器。

fn：回调函数即绑定在元素身上的侦听函数。

on方法优势1：

可以绑定多个事件，多个处理事件处理程序。

```
// 有多个事件有同样的功能时
$("div").on("mouseenter mouseleave", function () {
    $(this).toggleClass("current");
})
```

on方法优势2：

可以事件委派操作。事件委派的定义就是，把原来加给子元素身上的事件绑定在父元素身上，就是把事件委派给父元素。

```
// click时绑定在ul身上的，但是除法对象时ul里面的小li
$("ul").on("click", "li", function () {
    alert("zjgsuzjx");
})
```

on方法优势3：

动态创建的元素，click0没有办法绑定事件，on可以给动态生成的元素绑定事件

```
// 一般动态创建的元素不能绑定事件，用on可以
$("ol").on("click", "li", function () {
    alert("zjgsuzjx");
})
var li =$("<li>zjgsuzjx</li>");
$("ol").append(li);
```

微博发布案例：

```
$(".btn").on("click",function () {
    var li=$("<li></li>");
    li.html($(".txt").val()+"<a href='javascript:;'> 删除</a>");
    $("ul").prepend(li);
    li.slideDown();
    $(".txt").val("");

    $("ul").on("click" ,"a", function () {
        $(this).parent().slideUp(function () {
            $(this).remove();
        })
    })
})
```

2. 事件处理off

解绑事件off方法可以移除通过on方法添加的事件处理程序。

```
// 解除div身上所有的事件
$("div").off();
// 解除div身上的点击事件
$("div").off("click");
// 解除事件委托
$("ul").off("click", "li");
```

> 如果有的事件只想触发一次，可以使用one来绑定事件。

```
// 只触发一次
$("p").one("click", function () {
    alert("zjgsuzjx");
})
```

3. 自动触发事件trigger

有些事件希望自动触发，比如轮播图自动播放功能跟点击右侧按钮一致。可以利用定时器自动触发右侧按钮点击事件，不必鼠标点击触发。

```
// 自动触发事件
$("div").click();
$("div").trigger("click");
// 不会触发元素的默认行为
$("div").triggerHandler("click");
```

> 第三种和前两种的区别在于，比如input获得焦点触发功能，但是第三种不会获得焦点，第二种获得了焦点。

三、jQuery 事件对象

事件被触发，就会有事件对象的产生。

```
$(document).on("click", function () {
    console.log("dou");
})
$("div").on("click",function (event) {
    console.log("div");
    // 阻止事件冒泡
    event.stopPropagation();
})
```

阻止默认行为：event.preventDefault 或者 return false

阻止冒泡：event.stopPropagation

四、jQuery 其它方法

一、jQuery 拷贝对象

如果想要把某个对象拷贝（合并）给另外一个对象使用，此时可以使用$.extend()方法

```
jQuery.extend([deep], target, object1, [objectN])
```

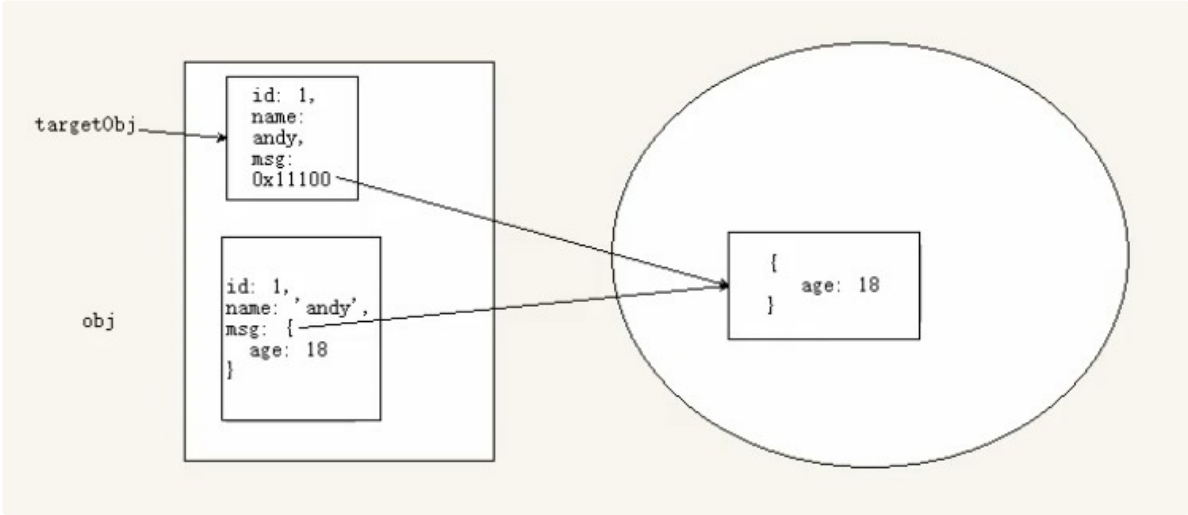
deep：如果设为true为深拷贝，默认为false浅拷贝

target：要拷贝的目标对象

object1：待拷贝到第一个对象的对象。

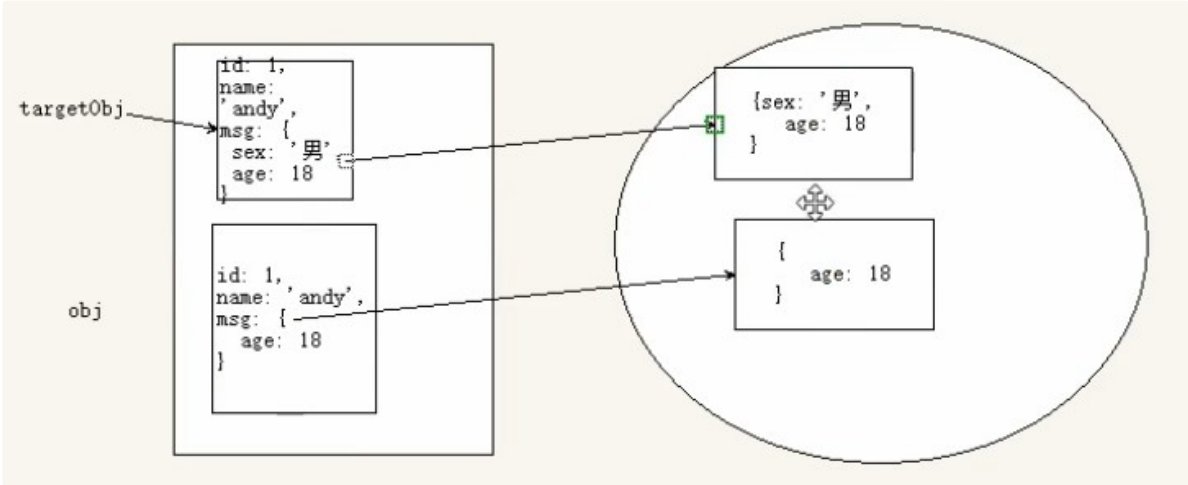
objectN：待拷贝到第N个对象的对象。

> 浅拷贝是把被拷贝的对象复杂数据类型中的地址拷贝给目标对象，修改目标对象会影响被拷贝对象。




```
var targetObj = {};
var obj = {
    id: 1,
    name: "zjgsuzjx"
}
// 把obj拷贝到targetObj
// 如果原先有数据，会被覆盖掉
$.extend(targetObj, obj);
```

> 深拷贝，前面加true，完全克隆（拷贝的对象，而不是地址），修改目标对象不会影响被拷贝对象。



二、多库共存

问题概述：

jQuery使用作为标示符，随jQuery的流行，其他js库也会用这S作为标识符，这样一起使用会引起冲突。

客观需求：

需要一个解决方案，让jQuery和其他的JS库不存在冲突，可以同时存在，这就叫做多库共存。

jQuery解决方案：

把里面的$符号统一改为jQuery。比如jQuery（"div'）

jQuery变量规定新的名称：

```
var zjgsuzjx = jQuery.noConflict();
console.log(zjgsuzjx("span"));
```

三、jQuery 插件

jQuery功能比较有限，想要更复杂的特效效果，可以借助于jQuery插件完成。

> 注意：这些插件也是依赖于jQuery来完成的，所以必须要先引入jQuery文件，因此也称为jQuery插件。

jQuery插件常用的网站

jQuery插件库 http://www.jq22.com/

jQuery之家 http://www.htmleaf.com/

jQuery插件使用步骤

1. 引入相关文件。（jquery文件和插件文件）
2. 复制相关html、css、js（调用插件）。

jQuery 插件演示

1. 瀑布流
2. 图片懒加载（图片使用延迟加载在可提高网页下载速度。它也能帮助减轻服务器负载）当我们页面滑动到可视区域，再显示图片。

> 我们使用jquery插件库EasyLazyload。注意，此时的js引入文件和js调用必须写到DOM元素（图片）最后面

3. 全屏滚动（fullpage.js）

   gitHub：https://github.com/alvarotrigo/fullPage.js

   中文翻译网站：http://www.dowebok.com/demo/2014/77/

4. bootstrap JS插件：

   bootstrap框架也是依赖于jQuery开发的，因此里面的js插件使用，也必须引入jQuery文件。

toDoList案例：

> 重点是将数据储存到浏览器缓存当中。

建议自己做一下

思路：

① 文本框里面输入内容，按下回车，就可以生成待力事项。

② 点击待办事项复选框，就可以把当前数据添加到已完成事项里面。

③ 点击已完成事项复选框，就可以把当前数据添加到待办事里面。

④ 但是本页面内容刷新页面不会丢失。