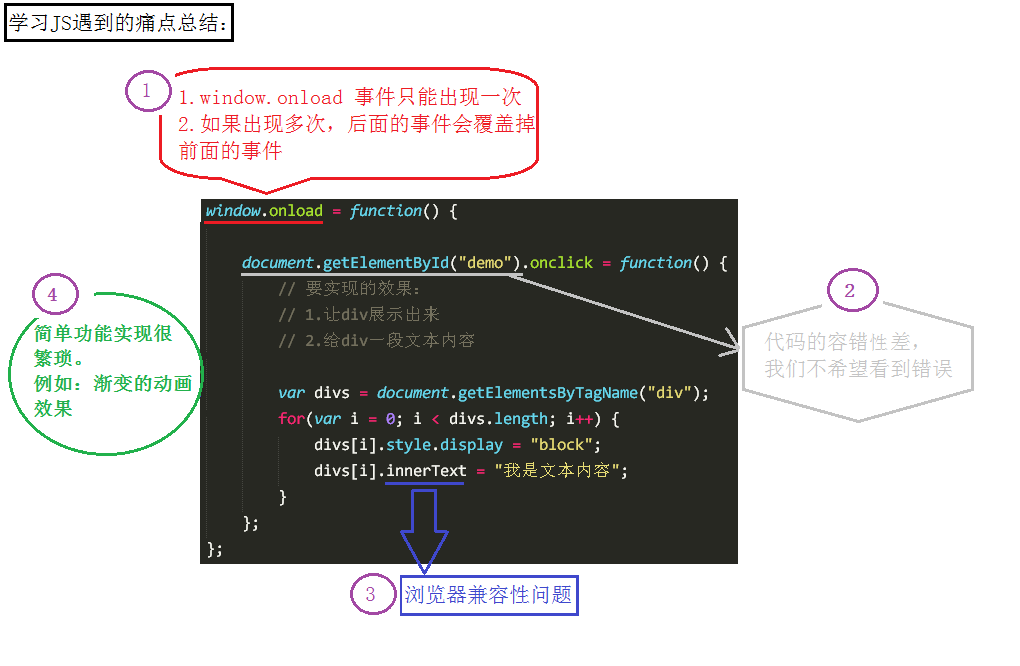
# 一 jQuery简介

## 1 jQuery的使用

版本一：1.x版本

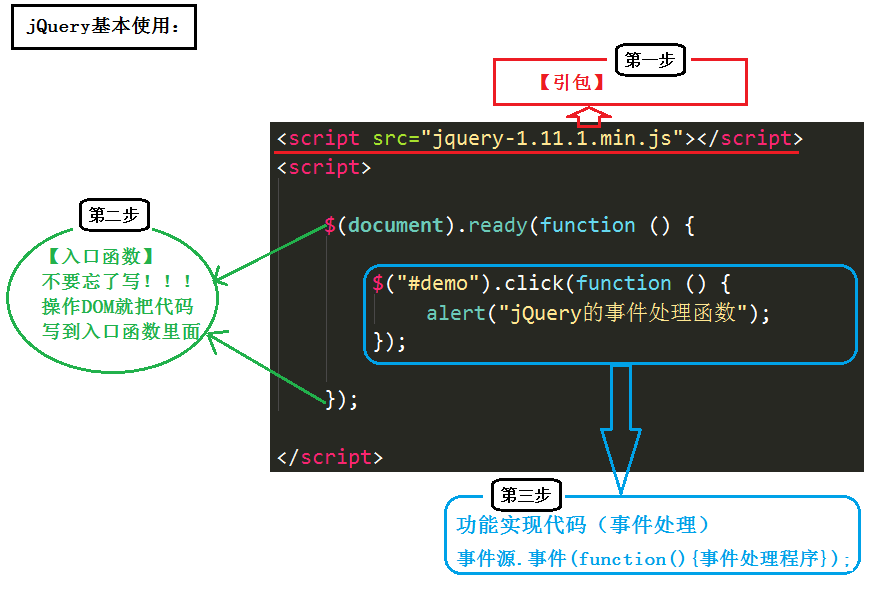
版本二：2.x版本

两个版本的区别：2.x版本，不再支持IE6、7、8



jQuery是js的一个库，封装了我们开发过程中常用的一些功能，方便我们来调用，提高了我们的开发效率。在使用jQuery的API时，都是方法调用，也就是说要加小括号()，小括号里面是相应的参数，参数不同，功能不同。

使用步骤：



<script src=”jQuery.js”></script>

//1 入口函数方式一

$(document).ready(function(){

});

//2 入口函数方式二

$(function(){

})

## 2 jQuery入口函数与js入口函数

js入口函数指的是：window.onload = function() {};

区别一：书写个数不同

Js入口函数只能出现一次，出现多次会存在事件覆盖的问题。

jQuery的入口函数，可以出现任意多次，并不会存在事件覆盖问题。

区别二：执行时机不同

Js入口函数是在所有的文件资源加载完成后，才执行。这些文件资源包括：页面文档、外部的js文件、外部的css文件、图片等。

jQuery的入口函数，是在文档加载完成后，就执行。文档加载完成指的是：DOM树加载完成后，就可以操作DOM了，不用等到所有的外部资源都加载完成。

文档加载的顺序：从上往下，边解析边执行。

## 3 jQuery的$符号

jQuery使用$符号原因：书写简洁、相对于其他字符与众不同、容易被记住。

怎么理解jQuery里面的$符号：

$实际上表示的是一个函数。

$(document）.ready(function(){}); // 调用入口函数

$(function(){}); // 调用入口函数

$(“#btnShow”) // 获取id属性为btnShow的元素

$(“div”) // 获取所有的div元素

jQuery里面的$函数，根据传入参数的不同，进行不同的调用，实现不同的功能。返回的是jQuery对象

jQuery这个js库，除了$之外，还提供了另外一个函数：jQuery

jQuery函数跟$函数的关系：jQuery ===$;

### 3.1 jQuery的事件源 事件处理函数

1. 事件源

Js方式：document.getElementById(“id”)

jQuery方式：$(“#id”)

1. 事件

Js方式 :document.getElementById(“id”).onclick

jQuery方式: $(“#id”).click

1. 事件处理程序

Js 书写方式:

document.getElementById(“id”).onclick = function(){

// 语句

}

jQuery 书写方式:

$(“#id”).click(function(){

// 语句

});

区别：jQuery的事件不带on

### 3.2 jQuery对象和DOM对象转换

DOM对象此处指的是：使用js操作DOM返回的结果。

var btn = document.getElementById(“btnShow”);

// btn就是一个DOM对象

jQuery对象此处指的是：使用jQuery提供的操作DOM的方法返回的结果。

jQuery拿到DOM对象后又对其做了封装，让其具有了jQuery方法的jQuery对象，说白了，就是把DOM对象重新包装了一下。

var $btn = $(“#btnShow”); // $btn就是一个jQuery对象

DOM对象转换成jQuery对象：

var $btn1 = $( document.getElementById(“btn”));

// 此时就把DOM对象btn转换成了jQuery对象$btn1

// $(document）.ready(function(){});

// 这个入口函数其实是将document这个js的DOM对象，转换成了jQuery对象，然后才能调用jQuery提供的方法：ready

jQuery对象转换成DOM对象：jQuery对象是一个数组。

// 第一种方式

var btn1 = $btn[0]; // 推荐使用此方式

// 第二种方式

var btn2 = $btn.get(0);

比如一个ul下所有的li获取：

$(‘li’).length length是jQuery属性

$(‘li’).get().length length是原生JS属性

$(‘li’)[1] 也可以直接获取第二个元素li

# 二 Jquery的常见使用

## 1 与原生JS混用以及jquery取值

jQuery与原生JS可以同时使用，但是最好不要混用：

<script src="jquery-3.3.1.min.js"></script>  
<script>  
 $(function () {  
 $('#div').click(function () {  
 alert($(this).html());  
 });  
 });  
</script>

这里使用$(this)来避免直接使用this引起的原生JS与jquery的混用。

其中html()方法没有参数，则是获取内容，有参数则是传递内容。

## 2 jQuery基础方法

$(function () {  
 $('div').filter('#div1').css('background','red');  
 $('div').has('span').css('background','green');  
});

filter: 过滤  
 not： filter的反义词  
 has： 是否包含,has查看的是当前元素是否包含，filter过滤的是所有同级元素

prev next : 选择上一个/下一个兄弟节点

find： 查找方法

eq(1): 结果集中的第1个，类似数组中使用下标

index(): 当前元素在所有兄弟节点中的位置

attr():

## 3 实战-原生JS选项卡

window.onload = function(){

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

    var aInput = oDiv.getElementsByTagName('input');

    var aCon = oDiv.getElementsByTagName('div');

    for(var i=0;i<aInput.length;i++){

        aInput[i].index = i;

        aInput[i].onclick = function(){

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

                aInput[i].className = '';

                aCon[i].style.display = 'none';

            }

            this.className = 'active';

            aCon[this.index].style.display = 'block';

        };

    }

};

## 4 实战-jQuery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');

    });

});

## 5 jQuery常见方法

添加/移除CSS类：

addClass() removeClass()

//获取宽度

width() 获取width

innerWidth() 获取width+左右padding

outerWidth() 获取width+左右padding+左右边框宽度

outerWidth(true)获取width+左右padding+左右边框宽度+左右margin

//获取到的元素剪切到某个位置（原生JS这里是选中后复制到某个位置）

insertBefore() insertAfter() 原生JS没有insertAfter()

//获取到的元素插入到某个位置

appendTo() 类似原生的appendChild()

prependTo() 插入到最开始位置，原生没有该方法

//元素必须在参数元素之前、之后等等

before() after() append() prepend()

区别：后续操作变了，如下所示第一行是span变红，第二行是div变红

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

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

//删除节点

remove()

//事件的新写法

on() off()

新写法案例：

    $('div').click(function(){

        alert(123);

    });

    $('div').on('click',function(){

        alert(123);

    });

新写法的强大之处：

    $('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');

    });

//滚动距离

scrollTop()

//获取父级

parent() : 获取父级

offsetParent() : 获取有定位的父级

val() //获取元素的value，带参数就是设置value值

size() //获取一组元素的长度，比如ul下的li的数量

hover()

元素.hover(function(){},function(){}); //第一个函数鼠标移入执行，第二个函数鼠标移除执行

//元素隐藏展示

show() hide()

//淡入淡出

fadeIn() fadeOut() fadeTo()

//向上卷曲 乡下展开

slideDown() slideUp()

## 6 事件对象

jQuery的事件对象ev已经是兼容的，不需要像原生一样再写兼容格式。

ev.pageX----相对于文档

ev.clientX----相对于可视区

以上同理还有Y轴对应方法。

ev.which:keycode

ev.preventDefault(); //阻止默认事件

 ev.stopPropagation(); //阻止冒泡的操作

如果：return false 则直接阻止全部！

    $('div').click(function(ev){

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

    });

//只执行事件一次

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

alert(123);

});

## 7 距离offset

    //原生

    div2.offsetLeft

    //获取到屏幕的左距离

    $('#div2').offset().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

    $('#div2').position().left;

## 8 循环each

可以看做for循环的加强版：

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

        $(elem).html(i);

    });

## 9 实战-jQuery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;

    });

});

## 10 text()

    $('div').html();

    $('div').text(); //会获取所有的内容(特例)

    $('div').text('<h3>h3</h3>');//全部以文本形式插入

# 三 jQuery选择器

jQuery选择器是jQuery强大的体现，jQuery实现了从CSS1到CSS3所有的选择器以及其他常用的选择器。

## 1 基本选择器

|  |  |  |
| --- | --- | --- |
| **符号(名称)** | **说明** | **用法** |
| **#** | Id选择器 | $(“#btnShow”).css(“color”, “red”);  选择id为btnShow的一个元素 |
| **.** | 类选择器 | $(“.liItem”).css(“color”, “red”);  选择含有类liItem的所有元素 |
| **element** | 标签选择器 | $(“li”).css(“color”, “red”);  选择标签名为li的所有元素 |

规律：$(selector).css(“background”,”red”);

## 2 层级选择器、基本过滤选择器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **符号(名称)** | | **说明** | **用法** | |
| **层级选择器** | | | | |
| **空格** | | 后代选择器  所有后代 | $(“#wrap li”).css(“color”, “red”);  选择id为wrap的元素的所有后代元素li | |
| **>** | | 子代选择器  直系后代 | $(“#wrap > ul > li”).css(“color”, “red”);  选择id为wrap的元素的所有子元素ul的所有子li | |
| **基本过滤选择器** | | | | |
| **:eq(index)** | 选择匹配元素中索引号index的一个元素，index从0开始 | | | $(“li:eq(2)”).css(“color”, ”red”);  li元素中索引号为2的一个元素 |
| **:odd** | 选择匹配元素中索引号为奇数的所有元素，index从0开始 | | | $(“li:odd”).css(“color”, “red”);  li元素中索引号为奇数的所有元素 |
| **:even** | 选择匹配元素中索引号为偶数的所有元素，index从0开始 | | | $(“li:odd”).css(“color”, “red”);  li元素中索引号为偶数的所有元素 |

层级选择器选择了选择符 后面那个元素，比如，div > p，是选择>后面的p元素。

## 3 筛选选择器

|  |  |  |
| --- | --- | --- |
| **符号(名称)** | **说明** | **用法** |
| **find(selector)** | 查找所有后代元素 | $(“#j\_wrap”).find(“li”).css(“color”, “red”);  选择id为j\_wrap的所有后代元素li |
| **children()** | 查找直接子元素 | $(“#j\_wrap”).children(“ul”).css(“color”, “red”);选择id为j\_wrap的所有子代元素ul |
| **siblings()** | 查找所有兄弟元素（不包括自己） | $(“#j\_liItem”).siblings().css(“color”, “red”);  选择id为j\_liItem的所有兄弟元素 |
| **parent()** | 查找父元素 | $(“#j\_liItem”).parent(“ul”).css(“color”, “red”);选择id为j\_liItem的父元素 |
| **eq(index)** | 查找指定元素的第index个元素 | $(“li”).eq(2).css(“color”, “red”);  选择所有li元素中的第二个 |

# 四 jQuery与DOM

## 1 jQuery样式操作

.css() 作用：设置或获取元素的样式属性值

$(selector).css(“color”, “red”); //设置单个样式

$(selector).css({“color”: “red”, “font-size”: “30px”}); //设置多个样式

$(selector).css(“font-size”); //获取样式

添加类样式：**(addClass)**

$(selector).addClass(“liItem”);

移除类样式：**removeClass(className)**

$(selector).removeClass(“liItem”);

$(selector).removeClass(); 不指定参数，表示移除被选中元素的所有类名

判断有没有类样式：hasClass(calssName) 返回true或false

$(selector).hasClass(“liItem”);

切换类样式：toggleClass(className) 该元素有类则移除，没有指定类则添加。

$(selector).toggleClass(“liItem”);

注意点：

操作类样式的时候，所有的类名，都不带点（.）

## 2 jQuery节点操作

### 2.1 动态创建元素

// $()函数的另外一个作用：动态创建元素

var $spanNode = $(“<span>我是一个span元素</span>”);

var node = $(“#box”).html（“<li>我是li</li>”）；

### 2.2 **append()**添加元素

**append()：**在元素的最后一个子元素后面追加元素（存在或者创建出来的元素都可以），

常用参数：htmlString 或者 jQuery对象

// 在$(selector)中追加$node

$(selector).append($node);

// 在$(selector)中追加div元素，参数为htmlString

$(selector).append('<div></div>');

其他操作：

1 appendTo() 把$(selector)追加到node中去

2 prepend() 在元素的第一个子元素前面追加内容或节点

3 after() 在被选元素之后，作为兄弟元素插入内容或节点

4 before() 在被选元素之前，作为兄弟元素插入内容或节点

### 2.3 html()创建元素

作用：设置或返回所选元素的html内容（包括 HTML 标记）

设置内容的时候，如果是html标记，会动态创建元素，此时作用跟js里面的 innerHTML属性相同

// 动态创建元素

$(selector).html(‘<span>传智播客</span>’);

// 获取html内容

$(selector).html();

推荐使用html(“<span></span>”)方法来创建元素

### 2.4 empty()清空元素

// 清空指定元素的所有子元素（光杆司令）

$(selector).empty();

$(selector).html(“”);

// “自杀” 把自己（包括所有内部元素）从文档中删除掉

$(selector).remove();

推荐使用html(“<span></span>”)方法来创建元素或者html(“”)清空元素

### 2.5 clone()复制元素

作用：复制匹配的元素，返回值为复制的新元素，参数的作用是复制操作行为

$(selector).clone();

$(selector).clone(true); //复制操作行为

### 2.6 删除元素

元素.remove();

返回值是当前被删除的元素，在其他地方仍然可以使用该返回值，但是事件没有了

元素.detach();

与remove使用一致，但是会保留元素的操作行为，即事件。

### 2.7 其他节点操作

parents()

获取所有祖先节点生成集合，参数具备筛选功能，只要集合有对应的参数都会被选中；

closesll()

获取最近的指定的祖先节点（包括元素自身），必须有参数，所以只能找到一个元素；

siblings()

查找所有的兄弟节点，参数具备筛选功能；

nextAll() prevAll()

nextAll查找下一个所有兄弟节点；preAll查找上一个所有的兄弟节点

nextUntil() prevUntil()

分别对应nextAll，preAll作用一致，但是可以写入参数，查找到指定位置；

parentsUntil()

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

包装方法：$(selector).wrap(‘<div></div>’); 给每个单独的选中标签外部包装一个div

wrapAll()：给选中的所有的标签整体包装一个标签；

wrapInner():将选中的每个标签的value值包装一个标签；

unwrap():删除父级，但是不包括body

add() slice()

add():组合操作，返回一个元素集；$(‘div’).add(‘span’);

slice(start,end):返回选择元素集中从第start-end位置的元素

## 5 jQuery操作form表单

### 5.1 属性操作

$(selector).attr(“title”, “jQeury简介”); //设置属性

$(selector).attr(“title”); //获取属性

$(selector).removeAttr(“title”); //移除属性

注意：checked、selected、disabled要使用.prop()方法。

prop方法通常用来影响DOM元素的动态状态，而不是改变的HTML属性。例如：input和button的disabled特性，以及checkbox的checked特性。

细节参考：<http://api.jquery.com/prop/>

### 5.2 值和内容

val()方法：设置或返回表单元素的值，例如：input,select,textarea的值

// 获取匹配元素的值，只匹配第一个元素

$(selector).val();

// 设置所有匹配到的元素的值

$(selector).val(“具体值”);

text() 方法:设置或获取匹配元素的文本内容

//获取操作不带参数（注意：这会把所有匹配到的元素拼接为一个字符串！）

$(selector).text();

//设置操作带参数，参数表示要设置的文本内容

$(selector).text(“我是内容”);

### 5.3 数据串连 serialize() serializeArray()

<form>

    <input type="text" name="a" value="1">

<input type="text" name="b" value="2">

<input type="text" name="c" value="3">

</form>

<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' }

        ]

    });

## 6 jQuery尺寸位置操作

### 6.1 高度和宽度操作

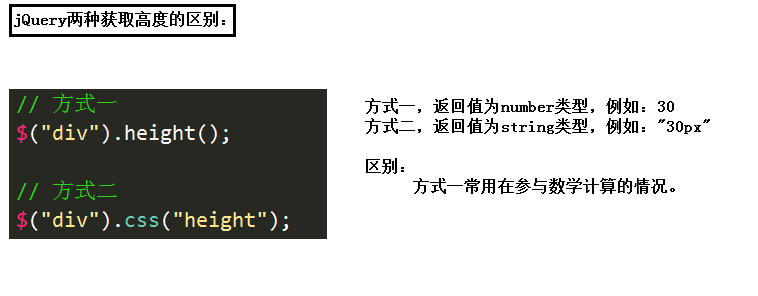
height() ： 设置或获取匹配元素的高度值

$(selector).height(200); //带参数表示设置高度

$(selector).height(); //不带参数获取高度

width() ： 设置或获取匹配元素的宽度值，用法同height()

css()获取高度和height获取高度的区别？



补充：

innerHeight：内边距+内容的高度 只能获取

innerWidth：内边距+内容的宽度 只能获取

outerHeight：左右内边距+内容+左右边框

outerWidth：同上

原生的outerWidth无法获取隐藏元素的值，而jQquery可以。所以jQuery中可以获取到隐藏元素的一些属性值。

### 6.2 坐标值操作

offset() : 获取或设置元素相对于文档的位置

// 无参数表示获取，返回值为：{left:num, top:num}，值是相对于document的位置

$(selector).offset();

// 有参数表示设置，参数推荐使用数值类型

$(selector).offset({left:100, top: 150});

注意点：设置offset后，如果元素没有定位(默认值：static)，则被修改为relative

position(): 获取相对于其最近的具有定位的父元素的位置。

// 获取，返回值为对象：{left:num, top:num}

$(selector).position();

注意：只能获取，不能设置。

scrollTop() : 获取或者设置元素垂直方向滚动的位置

// 无参数表示获取偏移,有参数表示设置偏移，参数为数值类型

$(selector).scrollTop(100);

scrollLeft() : 获取或者设置元素水平方向滚动的位置

$(selector).scrollLeft(100);

总结：

$(“div”).offset(); // 获取或设置坐标值 设置值后变成相对定位

$(“div”).position(); // 获取坐标值 子绝父相 只能读取不能设置

$(“div”).scrollTop(); // 相对于滚动条顶部的偏移

$(“div”).scrolllLeft(); // 相对于滚动条左部的偏移

**垂直滚动条位置 是可滚动区域 在 可视区域上方的 被隐藏区域的高度。**

**如果滚动条在最上方没有滚动 或者 当前元素没有出现滚动条，那么这个距离为0**

* scroll(hander) 绑定滚动事件
* scroll() 触发滚动事件 $(selector).scroll(function(){ //当选择的元素发生滚动的时候触发 });

## 7 jQuery动画

### 7.1 显示/隐藏show/hide

show方法:让匹配的元素展示出来

//参数为数值类型，表示：执行动画时长，单位为：毫秒（ms）

$(selector).show(2000); //参数2000即2秒钟显示完毕

//参数为字符串类型(jQuery预设的值:slow（600ms）、normal（400ms）、fast（200ms）

$(selector).show(“slow”);

//参数一可以是数值类型或者字符串类型，动画执行完后立即执行的回调函数

$(selector).show(2000, function() {});

//不带参数，作用等同于 css(“display”, ”block”)，此时没有动画效果

$(selector).show();

hide方法 让匹配元素隐藏掉，用法参考show方法

### 7.2 滑入/滑出slideDown/slideUp

滑入动画效果：让元素以下拉动画效果展示出来

$(selector).slideDown(speed,callback);

滑出动画效果：让元素以上拉动画效果隐藏起来

$(selector).slideUp(speed,callback);

滑入滑出切换动画效果

$(selector).slideToggle(speed,callback);

注意：以上方法省略参数或者传入不合法的字符串，那么则使用默认值：400毫秒

### 7.3 淡入/淡出fadeIn/fadeOut

淡入动画效果：让元素以淡淡的进入视线的方式展示出来

$(selector).fadeIn(speed, callback);

淡出动画效果：让元素以渐渐消失的方式隐藏起来

$(selector).fadeOut(1000);

淡入淡出切换动画效果：通过改变透明度，切换匹配元素的显示或隐藏

$(selector).fadeToggle('fast',function(){});

### 7.4 改变透明度fadeTo

与淡入淡出的区别：淡入淡出只能控制元素的不透明度从 完全不透明 到完全透明；而fadeTo可以指定元素不透明度的具体值。并且时间参数是必需的！

作用：调节匹配元素的不透明度，用法有别于其他动画效果

// 第一个参数表示：时长

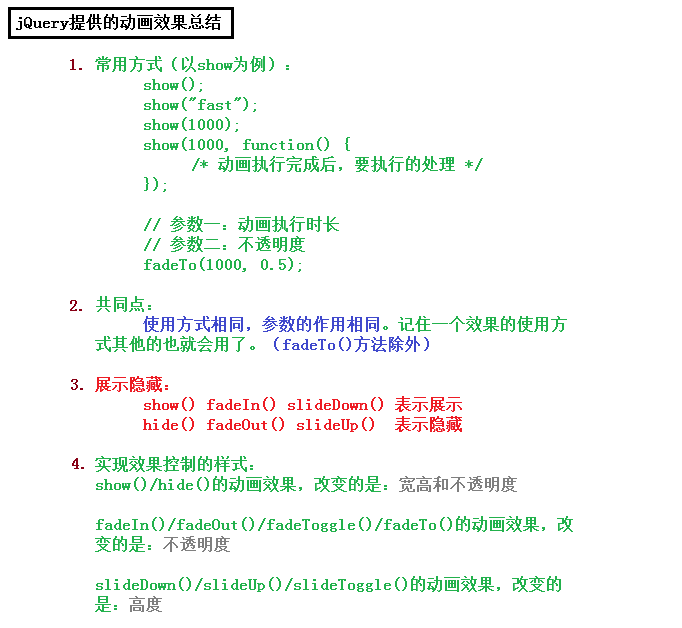
// 第二个参数表示：不透明度值，取值范围：0-1

$(selector).fadeTo(1000, .5); // 0全透，1全不透

// 第一个参数为0，此时作用相当于：.css(“opacity”, .5);

$(selector).fadeTo(0, .5);

### 7.5 jQuery动画总结



### 7.6 自定义动画animate

作用：执行一组CSS属性的自定义动画

// 第一个参数表示：要执行动画的CSS属性（必选）

// 第二个参数表示：执行动画时长（可选）

// 第三个参数表示：动画执行完后立即执行的回调函数（可选）

$(selector).animate({params},speed,callback);

//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');

    });

});

注意：所有能够执行动画的属性必须只有一个数字类型的值，比如：要改变字体大小，要使用：fontSize（font-size），不要使用：font

动画支持的属性：[http://www.w3school.com.cn/jquery/effect\_anima](http://www.w3school.com.cn/jquery/effect_animate.asp)[te.asp](http://www.w3school.com.cn/jquery/effect_animate.asp)

### 7.7 停止动画stop

stop()：停止当前正在执行的动画

为什么要停止动画？

如果一个以上的动画方法在同一个元素上调用，那么对这个元素来说，后面的动画将被放到效果队列中。这样就形成了动画队列， 动画队列里面的动画不会执行，直到第一个动画执行完成。

常用方式：

$(selector).stop();

带参数方式：第一个参数表示后续动画是否要执行

true：后续动画不执行

false：后续动画会执行

第二个参数表示当前动画知否要执行

true：立即执行完成当前动画

false：立即停止当前动画

参数全部不给，默认值都为false。

$(selector).stop(clearQueue,jumpToEnd);

解释：

当调用stop()方法后，队列里面的下一个动画将会立即开始。但是，如果参数clearQueue被设置为true，那么队列里剩余的动画就被删除了，并且永远也不会执行。

如果参数jumpToEnd被设置为true，那么当前动画会停止，但是参与动画的每一个CSS属性将被立即设置为它们的目标值。比如：slideUp()方法，那么元素会立即隐藏掉。如果存在回调函数，那么回调函数也会立即执行。

注意：如果元素动画还没有执行完，此时调用sotp()方法，那么动画将会停止。并且动画没有执行完成，那么回调函数也不会被执行。

$(function(){

    $('#div1').click(function(){

        $(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

    });

});