**Jquery**

## 概述

**1 jQuery是一个JavaScript函数库。**

是一个轻量级的"写的少，做的多"的JavaScript库。

**2 jQuery库包含以下功能：**

HTML 元素选取

HTML 元素操作

CSS 操作

HTML 事件函数

JavaScript 特效和动画

HTML DOM 遍历和修改

AJAX

Utilities

**3 为什么使用 jQuery**

很多大公司都在使用 jQuery， 例如:

Google

Microsoft

IBM

Netflix

**4 优势**

轻量级、体积小，使用灵巧(只需引入一个js文件)

强大的选择器

出色的DOM操作的封装

出色的浏览器兼容性

可靠的事件处理机制

完善的Ajax

链式操作、隐式迭代

方便的选择页面元素(模仿CSS选择器更精确、灵活)

动态更改页面样式/页面内容(操作DOM，动态添加、移除样式)

控制响应事件(动态添加响应事件)

提供基本网页特效(提供已封装的网页特效方法)

快速实现通信(ajax)

易扩展、插件丰富

**5 Ext对比Jquery**

1）.JQuery-EasyUI是仿照Ext做的。  
2）.Ext框架是一个整体，面向对象的编程思想，每个控件之间可以相互通讯。  
3）.JQuery的控件全都是分散的，没有整体性可言。你可以单独拿出来一个控件就能用。  
 4）.若要开发系统应用，首选Ext，控件库丰富，扩展和维护都方便。若是简单的页面动画和效果，首选JQuery  
 5）.jquery只是一个工具库，比较简单，相对容易。 Ext是一套真正的ria开发框架，甚至可以实现桌面应用一样的强大功能。本身代码质量极高，而且是高度的面向对象设计。jQuery 入门相对容易一些，实际用的过程中要用到各种插件，基本上用一个“学”一个。 ExtJS 入门稍难。

6）.大小比较：首先ExtJS是一个完整的Framework,是重量级别的，easy ui 是基于jquery库的一套UI组件库，是轻量级的，ExtJS是应用application级的，而jquery是page页面级的。当然application也是由page组成的，那就需要你自己去完成了，考虑你的需求，和使用框架的初衷，选择使用哪一种。同时ExtJs由于是重量级框架，完全面向对象风格，提供API非常完备也非常庞大，所以学习成本也想相对较大。

7）.兼容性比较：ExtJS兼容IE全系列浏览器和其他非IE现代浏览器，jquery UI向来不太考虑ie低版本浏览器的兼容，从态度上的鄙视。easy UI是基于jquery的，jquery2.X以上的版本不再支持IE6、7、8,，已郑重声明，请看官方网站，easyUI最新版本1.3.3使jQuery2.0,由于又很多HTML5特性，不再支持IE6\7\8，低版本由一些小部分的兼容不够好，请自己做技术选型的时候去测试，你要使用那个版本。在兼容问题上，他们都有瑕疵，看你的接收程度。

8）.使用许可license.  EXTJS  2.1以上版本，商用需要购买商业授权，jquery UI 使用MIT协议，开源。 jquery easyUI如果商用需遵循license commercial商业许可，也就是要购买使用权.

## 二、基础总结

1 查找id等于name的元素：

var nameTag = $("#name");

2 查找id等于name的标签的值：

var name = $("#name").val();

3 查找所有div元素：

var divTag = $("div");

4 查找所有样式类是 "myClass" 的元素

var myClassTag = $(".myClass");

5 找到每一个元素

var allTag = $("\*");

6 找到表单中所有的 input 元素

var inputTag = $("form input");

7 获取匹配的第一个元素和最后一个元素

var firstTag = $("li").first();

var lastTag = $("li").last();

8 查找所有未选中的 input 元素

var notCheckedTag = $("input:not(:checked)");

9 查找表格的1、3、5...行（奇数行）

var oddTrTag = $("tr:even");

10 查找表格的2、4、6...行（偶数行）

var evenTrTag = $("tr:odd");

11 查找表格第二行

var tempTag = $("tr:eq(1)");

12 查找大于第三的div元素

var tempTag = $("div:gt(2)");

13 查找小于第三行的行

var tempTag = $("tr:lt(2)");

14 查找所有包含"Wade"的div元素

var tempTag = $("div:contains('Wade')");

15 给所有 div 元素添加一个 text 类

$("div").addClass("text");

16 查找隐藏的tr/找出type为隐藏域的标签

var tempTag = $("tr:hidden");

var tempTag = $("input:hidden");

17 查找可见的tr

var tempTag = $("tr:visible");

18 查找所有含有id属性的div元素

var tempTag = $("div[id]");

19 给样式为sub\_title\_ico的标签下的strong标签赋值

$(".sub\_title\_ico strong").val("555");

20 给样式为sub\_title\_ico的标签下的b标签加html代码

$(".sub\_title\_ico b").html("555");

21 查找所有name属性不是wade的input元素默认为选中状态

$("input[name!='Wade']").attr("checked","true");

22 查找所有name以'news'开始的input元素

var tempTag = $("input[name^='news']");

23 查找所有name以 'news' 结尾的input元素

var tempTag = $("input[name$=news]");

24 查找所有name包含'news'的input元素

var tempTag = $("input[name\*=news]");

25 查找所有name包含'news'且有属性id的input元素

var tempTag = $("input[name\*=news][id]");

26 查找所有选中的选项元素

var tempTag = $("select option:selected");

27 返回文档中第一个图像的src属性值

var temp = $("img").first().attr("src");

28 为所有图像设置alt属性

$("img").attr("alt","select");

29 为所有图像设置src和alt属性

$("img").attr({"src":"test.jpg","alt":"select"});

30 将文档中第一个图像的src属性删除

$("img").first().removeAttr("src");

31 为匹配的元素加上 'selected' 类

$("img[id]").addClass("selected");

32 为匹配的元素删除 'selected' 类

$("img[id]").removeClass("selected");

33 取得第一个匹配元素的html内容

var temp = $("#name").html();

34 取得所有匹配元素的内容

var temp = $("#name").text();

35 设置所有匹配元素的文本内容

$("#name").text("555");

36 设置每一个匹配元素的值

$("input").val("hello world!");

37 要追加到目标中的内容

$("p[id=name]").append("<b>Hello</b>");

38 清空目标中的内容

$("p[id=name]").html("");

39 在所有段落之后插入一些HTML标记代码

$("p").after("<b>Hello</b>");

40 在所有段落之前插入一些HTML标记代码

$("p").before("<b>Hello</b>");

41 将所有匹配的元素替换成指定的HTML或DOM元素。

$("p").replaceWith("<b>Paragraph. </b>");

42 删除匹配的元素集合中所有的子节点。

$("p").empty();

43 从DOM中删除所有匹配的元素。

$("p").remove();

44 访问第一个匹配元素的样式属性

$("p").css("color");

45 在所有匹配的元素中，设置一个样式属性的值

$("p").css("color","red");

46 在所有匹配的元素中，设置多个样式属性的值

$("p").css({color:"red",background:"blue"});

$("p").css({color:"red","background-color":"blue"});

47 取得第一个匹配元素当前计算的高度值/宽度值（px）

$("p").height();

$(window).width();

$(document).height();

48 设置第一个匹配元素高度值/宽度值（px）

$("p").height(50);

$(window).height(50);

$(document).width(50);

49 onload事件

$(document).ready(function(){});

$("#deptName").ready(function(){});

$(function()

{

// 文档就绪

});

50 绑定与取消绑定事件

$(document).bind('click', function()

{

alert(5556);

});

$("#deptName").unbind("click");

blur, focus, focusin, focusout, load, resize, scroll, unload, click, dblclick, mousedown, mouseup,

mousemove, mouseover, mouseout, mouseenter, mouseleave, change, select, submit, keydown, keypress,

keyup, error

51 绑定一次事件

$(document).one('click', function()

{

alert(5556);

});

52 鼠标悬停的表格加上特定的类

$("td").hover(

function ()

{

$(this).addClass("hover");

},

function ()

{

$(this).removeClass("hover");

}

);

53 在每一个匹配元素的unload事件中绑定一个处理函数

$(window).unload( function () { alert("Bye now!"); } );

54 提交本页的第一个表单

$("form:first").submit();

55 在每一个匹配元素的submit事件中绑定的处理函数

$("form").submit( function ()

{

return false;

});

56 AJAX请求

$.ajax

({

type: "POST",

async: true,

url: "statistic.do?",

timeout:5000,

cache: false,

data: {"id":id, "commentPoint":commentPoint, "commentContents":commentContents},

beforeSend: function(XMLHttpRequest){}

//success: function(msg){}

success: processData

});

function processData(data)

{

}

57 触发一些事件

$("#deptName").click();

$("#deptName").click(function (){$(this).hide();});

$("#deptName").blur();

$("#deptName").keydown();

$("#deptName").keypress();

$("#deptName").keyup();

$("#deptName").load();

58 去掉左右的空格

var str = " 7788 877 ";

str = $.trim(str);

59 判断是否是数组

var newArray = new Array();

$.isArray(newArray);

60 判断是否是方法

$.isFunction("setColor");

function setColor(){};

61 查看对应元素的位置

var arr = [ 4, "Pete", 8, "John" ];

$.inArray("John", arr); //3

62 关于json和json数组

normalArray

[1,2,3,4,5]

jsonObj

{

"result":"success",

"errorCode":"12345"

}

jsonArray

[

{

"id":"324",

"tag":"link1234",

"objecteID":"HW\_076769",

"appNumber":"A17959558",

"countNum":"1",

"indexDB":"WWW\_CN",

"errorCode":"",

"errorDescription":""

},

{

"id":"322",

"tag":"link123",

"objecteID":"HW\_076769",

"appNumber":"A17959558",

"countNum":"1",

"indexDB":"WWW\_CN",

"errorCode":"",

"errorDescription":""

}

]

63 JSONObject

import org.json.JSONObject;

import org.json.JSONArray;

JSONObject jsonObj = new JSONObject();

jsonObj.put(1,'ONE');

jsonObj.put(2,'TWO');

JSONArray jsonArray = new JSONArray();

jsonArray.put(jsonObj);

$(normalArray).each(function (index)

{

alert(index+"===="+this);

});

var result = jsonObj.result;

alert("result==="+result);

$(jsonArray).each(function (index)

{

alert(index+"===="+this.appNumber);

});

var nbaWade = {};

var nbaMcgrady = {};

var nbaJames = {};

nbaWade.setting =

{

height:"193cm",

avoirdupois:"104.5",

club:"HEAT",

avgPoint:"28.5",

position:"Score guard",

inYear:"2003"

}

nbaWade.load = function (param)

{

var height = nbaWade.setting.height;

var avgPoint = nbaWade.setting.avgPoint;

document.getElementById("hiddenSpan").innerHTML=avgPoint;

if(typeof param == 'function')

{

alert("function");

}

if(typeof param == 'number')

{

alert("number");

}

if(typeof param == 'string')

{

alert("string");

}

}

var nbaArray = [nbaWade,nbaMcgrady,nbaJames];

var isearch = {};

isearch.isearch\_vo={};

isearch.isearch\_vo.TagValuesParamsClientVO = function(callback,appNumber)

{

this.callback = callback;

this.appNumber = appNumber;

}

isearch.isearch\_client={};

isearch.isearch\_client.getQueryTagValue = function(searchParams)

{

alert("searchParams.callback====="+searchParams.callback);

alert("searchParams.appNumber====="+searchParams.appNumber);

if(searchParams.callback!="")

{

}

return 6;

}

var searchParams = new isearch.isearch\_vo.TagValuesParamsClientVO();

searchParams.callback="processData";

searchParams.appNumber="A3579144";

var rrrr = isearch.isearch\_client.getQueryTagValue(searchParams);

alert("rrrr====="+rrrr);

function processData(data)

{

alert("data====="+data);

}

## 全面总结

**1 核心**

**$是jquery类的一个别名,$()构造一个jquery对象,jQuery 的核心功能都是通过这个函数实现的。 jQuery中的一切都基于这个函数，或者说都是在以某种方式使用这个函数。**

1） ${表达式}：根据这个表达式来查找所有匹配的元素。

      eg:$("div>p"); 查找所有p元素，且这些p元素都是div的子元素.

         $("input:radio",document.forms[0]); 查找文档第一个表单中，所有的单选按钮。

         $("div",xml.responseXML);在一个由ajax返回的xml文档中，查找所有的div元素。

2）$(html标记字符串) :根据提供的html字符串，创建jquery对象包装的dom元素。

       eg: $("<div><p>Hello</p></div>").appendTo("body");

          创建一个 <input> 元素必须同时设定 type 属性

          $("<input type='text'>")

3）$(dom元素):将一个或多个dom元素转换为jquery对象。

     eg: $(document.body).css( "background", "black" ); 设置页面背景色

4）$(document).ready(function(){})==$(function(){})；在页面加载完成后自动执行的代码。

**2 jquery对象访问。**

eq返回的是jquery对象，只能使用jquery方法，get返回的是dom元素对象，只能使用dom方法.

   1）$(dom元素).each(function(){}):以每一个匹配的元素作为上下文来执行一个函数

   2）$(dom元素).size()==$(dom元素).length;对象中元素的个数。eg:<img src="test1.jpg"/> <img src="test2.jpg"/>,jquery代码：$("img").size(); 返回2

   3）context：返回传给jQuery()的原始的DOM节点内容，即jQuery()的第二个参数。如果没有指定，那么context指向当前的文档(document)。

   4）$(dom元素).get()：取得所有匹配的dom元素的集合。如果你想要直接操作 DOM 对象而不是 jQuery 对象，这个函数非常有用。

   5）$(dom元素).get(index)：取得其中一个匹配的元素.

   6）$(dom元素).index():搜索与参数表示的对象匹配的元素，并返回相应元素的索引值。

**3 插件机制:**

1）jQuery.fn.extend(object);扩展 jQuery 元素集来提供新的方法,针对某一个dom元素。

      eg:jQuery.fn.extend({

          check: function() { return this.each(function() { this.checked = true; });},

          uncheck: function() { return this.each(function() { this.checked = false; });}

          });

      调用：

         $("input[type=checkbox]").check();

         $("input[type=radio]").uncheck();

2）$.extend(object);用来在jQuery命名空间上增加新函数,针对所有dom元素。

     $.extend({

       min:function(a,b){return a<b?a:b;},

       max:function(a,b){return a>b?a;b;}

     });

    调用：$.max(10,20);

**4 选择器**

**@基本：**

   1）$("#id"): 根据给定的ID匹配一个元素。

   2）$("dom元素名")：根据给定的元素名匹配所有元素

   3）$(".Class类名"): 根据给定的类匹配元素。

   4）$("\*") ：匹配所有元素。

   5）$("dom元素.class类名")：选择所有class属性为指定类名的dom元素。

   6）$(".stripe tr"): 获取class属性为stripe的table元素下的所有行

**@层级：**

   1）$("ancestor descendant"):在给定的祖先元素下匹配所有的后代元素

   2）$("parent > child"):在给定的父元素下匹配所有的子元素

   3）$("prev + next"):匹配所有紧接在 prev 元素后的 next 元素

   4）$("prev ~ siblings"):匹配 prev 元素之后的所有 siblings(同级) 元素

**@简单:**

1）$("tr:first") 匹配找到的第一个元素

2）$("tr:last") 匹配找到的最后一个元素

3）$("input:not(:checked)") 去除所有与给定选择器匹配的元素

4）$("tr:even") 匹配所有索引值为偶数的元素，从 0 开始计数

5）$("tr:odd") 匹配所有索引值为奇数的元素，从 0 开始计数

6）$("tr:eq(1)") 匹配一个给定索引值的元素

7）$("tr:gt(0)") 匹配所有大于给定索引值的元素

8）$("tr:lt(2)") 匹配所有小于给定索引值的元素

**@内容：**

1）$("div:contains('John')") 匹配包含给定文本的元素

2）$("div:has(p)") 匹配所有包含 p 元素的 div 元素

3）$("td:empty") 查找所有不包含子元素或者文本的空元素

**@可见性：**

1）$("tr:visible") 查找所有可见的 tr 元素

2）$("tr:hidden") 匹配所有的不可见元素，input 元素的 type 属性为 "hidden" 的话也会被匹配到

**@属性：**

1）$("div[id]") 查找所有含有 id 属性的 div 元素

2）$("input[name='newsletter']") 查找所有 name 属性是 newsletter 的 input 元素

3）$("input[name!='newsletter']") 查找所有 name 属性不是 newsletter 的 input 元素

4）$("input[name^='news']") 查找所有 name 以 'news' 开始的 input 元素

5）$("input[name$='letter']") 查找所有 name 以 'letter' 结尾的 input 元素

6）$("input[name\*='man']") 查找所有 name 包含 'man' 的 input 元素

7）$("input[id][name$='man']") 找到所有含有 id 属性，并且它的 name 属性是以 man 结尾的

**@子元素：**

1）nth-child(even)/:nth-child(odd)/:nth-child(3n)/:nth-child(2)/:nth-child(3n+1)/:nth-child(3n+2) 匹配其父元素下的第N个子或奇偶元素

2）$("ul li:first-child") 在每个 ul 中查找第一个 li

3）$("ul li:last-child")   在每个 ul 中查找最后一个 li

4）$("ul li:only-child")   在 ul 中查找是唯一子元素的 li

**@表单：**

1）$(":input") 查找所有的input元素

2）$("text")    匹配所有的单行文本框

3）$(":password") 匹配所有密码框

4）$("radio")   匹配所有单选按钮

5）$("checkbox") 匹配所有复选框

6）$("submit")   匹配所有提交按钮

7）$("image")    匹配所有图像域

8）$("button")   匹配所有按钮

9）$("file")     匹配所有文件域

10）$("hidden") 匹配所有不可见元素，或者type为hidden的元素

**@表单对象属性：**

1）$("input:checked") 匹配所有选中的被选中元素(复选框）单选框等，不包括select中的option)

2）$("input:enabled") 匹配所有可用元素

3）$("input:disabled") 匹配所有不可用元素

4）$("select option:selected") 匹配所有选中的option元素

**5 属性操作**

**@操作属性：**

    1）$("img").attr("src"); 返回文档中第一个图像的src属性值。

    2）$("img").attr("src","test.jpg"); 为所有匹配的元素设置一个属性值

    3）$("img").attr("title", function() { return this.src }); 所有匹配元素设置一个计算的属性值

    4）$("img").removeAttr("src"); 从每一个匹配的元素中删除一个属性

    5）$("img").attr({ src: "test.jpg", alt: "Test Image" }); 将一个“名/值”形式的对象设置为所有匹配元素的属性。

**@操作CSS:**

    1）$("p").addClass("selected"); 为每个匹配的元素添加指定的类名

    2）$("p").removeClass("selected"); 从所有匹配的元素中删除全部或者指定的类。一个或多个要删除的CSS类名，请用空格分开

$("p").removeClass();删除匹配元素的所有类

    3）$("p").toggleClass("selected"); 如果存在（不存在）就删除（添加）一个类

    4） var count = 0;

       $("p").click(function(){

      $(this).toggleClass("highlight", count++ % 3 == 0);

}); 每点击三下加上一次 'selected' 类

**@HTML代码:**

   1）$("div").html(); 取得第一个匹配元素的html内容

   2）$("div").html("<p>Hello Again</p>"); 设置每一个匹配元素的html内容

**@文本：**

1）$("p").text(); 取得所有匹配元素的内容

2）$("p").text("<b>Some</b> new text."); 设置所有匹配元素的文本内容

**@值：**

1） $("#single").val() 获得第一个匹配元素的当前值。

2）$("input").val("hello world!"); 设置每一个匹配元素的值

**6 筛选**

**@过滤：**

   1）$("p").eq(1) 获取第N个元素，这个元素的位置是从0算起

   2） if ( $(this).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。

   3）$("p").filter(".selected") 筛选出与指定表达式匹配的元素集合

   4）is(expr) 用一个表达式来检查当前选择的元素集合，如果其中至少有一个元素符合这个给定的表达式就返回true。

   5）$("p").not( $("#selected")[0] ) 删除与指定表达式匹配的元素

**@查找：**

   1）$("div").children();取得一个包含匹配的元素集合中每一个元素的所有子元素的元素集合。

   2）$("p").find("span");搜索所有与指定表达式匹配的元素。这个函数是找出正在处理的元素的后代元素的好方法。

   3）$("p").next();取得一个包含匹配的元素集合中每一个元素紧邻的后面同辈元素的元素集合

   4）$("p").prev();取得一个包含匹配的元素集合中每一个元素紧邻的前一个同辈元素的元素集合。

   5）$("div").parent;取得一个包含着所有匹配元素的唯一父元素的元素集合。

   6）$("span").parents;找到每个span元素的所有祖先元素

   7）$("div").siblings();取得一个包含匹配的元素集合中每一个元素的所有唯一同辈元素的元素集合

**7 文档处理**

**@内部插入：**

   1）$("p").append("<b>Hello</b>"); 向每个匹配的元素内部尾部追加内容

   2）$("p").prepend("<b>Hello</b>"); 向每个匹配的元素内部前部追加内容

   3）$("p").appendTo("#foo"); 把所有段落追加到ID值为foo的元素中。

**@外部插入：**

   1）$("p").after("<b>Hello</b>"); 在每个匹配的元素之后插入内容

   2）$("p").before("<b>Hello</b>"); 在每个匹配的元素之前插入内容

**@包裹**

   1）$("p").wrap("<div class='wrap'></div>"); 把所有匹配的元素用其他元素的结构化标记包裹起来。

**@替换：**

     1）$("p").replaceWith("<b>Paragraph. </b>"); 将所有匹配的元素替换成指定的HTML或DOM元素。

**@删除：**

   1）$("p").empty(); 删除匹配的元素集合中所有的子节点。

   2）$("p").remove(); 从DOM中删除所有匹配的元素

**8 CSS操**作

**@CSS:**

    1）$("p").css("color"); 取得第一个段落的color样式属性的值

2）$("p").css({ color: "#ff0011", background: "blue" });

把一个“名/值对”对象设置为所有匹配元素的样式属性。

    3）$("p").css("color","red"); 在所有匹配的元素中，设置一个样式属性的值

**@位置：**

    1）var offset = p.offset();获取匹配元素在当前视口的相对偏移，返回的对象包含两个整形属性：top 和 left。此方法只对可见元素有效。

    2）var position = p.position();获取匹配元素相对父元素的偏移

**@尺寸：**

    1）$("p").height(); 取得第一个匹配元素当前计算的高度值（px）。

    2）$("p").height(20); 为每个匹配的元素设置CSS高度(hidth)属性的值

3）$("p").width(); 取得第一个匹配元素当前计算的宽度值（px）。

$(window).width(); 获取当前窗口的宽

    4）$("p").width(20); 为每个匹配的元素设置CSS宽度(width)属性的值

**9 效果**

**@基本：**

    1）$("p").show() 显示隐藏的匹配元素

    2）$("p").hide() 隐藏显示的元素

    3）$("p").show("slow"); 以优雅的动画显示所有匹配的元素，并在显示完成后可选地触发一个回调函数。

      speed (String,Number) : 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)

callback (Function) : (Optional) 在动画完成时执行的函数，每个元素执行一次。

   用迅速的动画将隐藏的段落显示出来，历时200毫秒。并在之后执行反馈！

   <p style="display: none">Hello</p>

**@jQuery 代码:**

   $("p").show("fast",function(){

   $(this).text("Animation Done!");

   });

   4）$("p").hide("slow"); 以优雅的动画隐藏所有匹配的元素，并在显示完成后可选地触发一个回调函数。

   5）$("p").toggle() 切换元素的可见状态

   6）$("p").toggle("slow"); 以优雅的动画切换所有匹配的元素，并在显示完成后可选地触发一个回调函数

**@淡入淡出:**

1）$("p").fadeIn("slow"); 通过不透明度的变化来实现所有匹配元素的淡入效果，并在动画完成后可选地触发一个回调函数

2）$("p").fadeOut("slow"); 通过不透明度的变化来实现所有匹配元素的淡出效果，并在动画完成后可选地触发一个回调函数。

3）$("p").fadeTo("slow", 0.66);$("p").fadeTo("slow", 0.66); 把所有匹配元素的不透明度以渐进方式调整到指定的不透明度，并在动画完成后可选地触发一个回调函数

**@自定义：**

   1）$("#block").animate({

    width: "90%",

    height: "100%",

    fontSize: "20em",

    borderWidth: 10

201}, 1000 );

用于创建自定义动画的函数。

**10 事件**

**@事件处理：**

   1）$("p").bind("click", function(){ alert( $(this).text() );}); 为每一个匹配元素的特定事件（像click）绑定一个事件处理器函数

   2）$("p").one("click",function(){alert(${this}.text());}) 当所有段落被第一次点击的时候，显示所有其文本

   3）trigger(type,[data])在每一个匹配的元素上触发某类事件

**@事件委派：**$("p").live("click", function(){$(this).after("<p>Another paragraph!</p>");});。给所有当前以及将来会匹配的元素绑定一个事件处理函数（比如click事件）。也能绑定自定义事件。与bind()不同的是，live()一次只能绑定一个事件

**@事件切换:**一个模仿悬停事件（鼠标移动到一个对象上面及移出这个对象）的方法

   $("td").hover(

function () {

    $(this).addClass("hover");

},

function () {

    $(this).removeClass("hover");

}

);

**@每次点击后依次调用函数。**

$("td").toggle(

function () {

    $(this).addClass("selected");

},

function () {

   $(this).removeClass("selected");

}

);

**@事件:**

   1）$("p").blur(); 触发每一个匹配元素的blur事件

   2）$("p").blur( function () { alert("Hello World!"); } ); 在每一个匹配元素的blur事件中绑定一个处理函数

   3）change()；change事件会在元素失去焦点的时候触发，也会当其值在获得焦点后改变时触

   4）change(fn);在每一个匹配元素的change事件中绑定一个处理函数。

   5）$("p").click(); 触发每一个匹配元素的click事件

   6）$("p").click(function(){});在每一个匹配元素的click事件中绑定一个处理函数

   7）dblclick()；dblclick事件会在元素的同一点双击时触发。

   8）$("p").dblclick( function () { alert("Hello World!"); }); 在每一个匹配元素的dblclick事件中绑定一个处理函数。

   9）$(document).ready(function(){ $("#login").focus();}); 触发每一个匹配元素的focus事件。

   10）$("input[type=text]").focus(function(){this.blur();}); 在每一个匹配元素的focus事件中绑定一个处理函数。

   11）keydown()；keydown事件会在键盘按下时触发。

   12）keydown(fn)；在每一个匹配元素的keydown事件中绑定一个处理函数。

   13）keypress()；触发每一个匹配元素的keypress事件

   14）keypress(fn)；在每一个匹配元素的keypress事件中绑定一个处理函数。

   15）keyup()；触发每一个匹配元素的keyup事件

   16）keyup(fn)；在每一个匹配元素的keyup事件中绑定一个处理函数。

   17）mousedown();

   18）mouseup();

   19）mousemove();

   20）mouseover();

   21）mouseout();

   22）$("form:first").submit();提交本页的第一个表单。

   23）$("form").submit(function(){return false;});阻止页面提交。

**11 ajax**

       1）通有方式：$.ajax(prop);prop是一个hash表，它可以传递的key/value有以下几种:

         (String)type：数据传递方式(get或post)。

         ((String)url：数据请求页面的url

         ((String)data：传递数据的参数字符串，只适合post方式

         ((String)dataType：期待数据返回的数据格式(例如 "xml", "html", "script",或 "json")

         ((Boolean)ifModified： 当最后一次请求的相应有变化是才成功返回，默认值是false

         ((Number)timeout:设置时间延迟请求的时间。可以参考$.ajaxTimeout

         ((Boolean)global：是否为当前请求触发ajax全局事件，默认为true

         ((Function)error：当请求失败时触发的函数。

         ((Function)success：当请求成功时触发函数

         ((Function)complete：当请求完成后出发函数

      2）$.get(url, params, callback) 用get方式向远程页面传递参数，请求完成后处理函数，除了url外，其它参数任意选择！

         eg:$.get(   "ajax.asp",

            { name: "young", age: "25" },

            function(data){ alert("Data Loaded: " + data); }

          )

      3）$.getJSON(url, params, callback) 用get方式向远程json对象传递参数，请求完成后处理函数callback

      4）$.post(url, params, callback) 用post方式向远程页面传递参数，请求完成后处理函数callback

**12 工具类**

**@数组和对象操作：**

     1）$.each(obj,callback); 通用例遍方法，可用于例遍对象和数组。回调函数拥有两个参数：第一个为对象的成员或数组的索引，第二个为对应变量或内容。

       $.each( [0,1,2], function(i, n){alert( "Item #" + i + ": " + n );});

     2）jQuery.grep(array,callback,[invert])；使用过滤函数过滤数组元素，此函数至少传递两个参数：待过滤数组和过滤函数

     3）$.map(array,callback);将一个数组中的元素转换到另一个数组中

     4）$.inArray(value,array);确定第一个参数在数组中的位置(如果没有找到则返回 -1 ),从0开始。

     5）$.merge( [0,1,2], [2,3,4] ) ；合并两个数组。返回的结果会修改第一个数组的内容——第一个数组的元素后面跟着第二个数组的元素，不去掉重复项。

     6）$.unique(array);删除数组中重复元素

     7）$.trim(" hello, how are you? "); 去掉字符串起始和结尾的空格

     8）$.isArray(obj);测试对象是否为数组。

     9）$.isFunction(obj); 测试对象是否为函数

**@URL:**

   1）$.param(obj);将表单元素数组或者对象序列化。是.serialize()的核心方法。

     eg:var params = { width:1680, height:1050 }; var str = jQuery.param(params);

   2）$("form").serialize() ；序列表单内容为字符串，用于 Ajax 请求

**13 jquery操作单选框**

   if($("input[type='radio']:checked")){

     var tt=$("input[name='chk']:checked").val();

}

**14 jquery操作复选框**

    var chkArr="";

$(".cc:checked").each(function(){

       chkArr+=$(this).val()+"|";

});

**15 jquery实现复选框全选**

   var checked=$("#allchk").attr("checked");

$(".ww").each(function(){

   if($(this).attr("checked")!=checked){$(this).click();}

});