# jQuery简介

## jQuery简介

### 什么是jQuery?

jQuery的字面意思其实就是JavaScript和查询（Query），即用于辅助开发[JavaScript](http://www.w3cschool.cn/javascript" \t "http://www.w3cschool.cn/jquery/_blank)的库。

jQuery的作者是John Resig，2006年1月，jQuery第一个版本发布，它以简洁、灵活的编程风格让人一见倾心。随后有越来越多的开发者加入。

jQuery已经集成了JavaScript、css、DOM和Ajax于一体的强大功能。其语法设计可以使开发更加便捷，例如操作文档对象、选择DOM元素、制作动画效果、事件处理、使用Ajax以及其他功能。除此之外，jQuery提供API让开发者编写插件。使开发者可以很轻松的开发出功能强大的静态或动态网页。

总结：

* jQuery 是一个基于JavaScript的免费、开源的库。
* jQuery 极大地简化了 JavaScript 编程。
* jQuery 功能强大，但很容易学习。
* jQuery 是一个"写的更少，但做的更多”的轻量级 JavaScript 库。

### 与JS比较

1、jQuery的优势:

* 简化 JavaScript 代码操作
* 像 CSS 那样访问和操作 DOM
* 修改 CSS 控制页面外观
* 事件处理更加容易
* 各种动画效果使用方便
* 让 Ajax 技术更加完美

2、jQuery的缺陷:

* 不能向后兼容。每一个新版本不能兼容早期的版本。
* 插件兼容性不是太好，与上一点类似。
* 在同一页面上使用多个插件时，很容易碰到冲突现象，尤其是这些插件依赖相同事件或selector时最为明显

### jQuery的版本发展

参考当天资料resource目录下的《jQuery 的版本.docx》

## 下载、引入jQuery

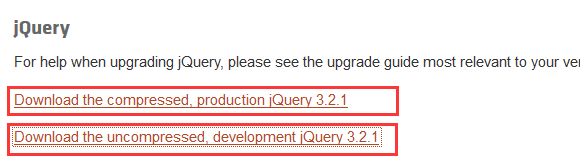
### 下载 jQuery

jQuery的使用非常简单，我们只需要引入jQuery库的js文件，然后直接使用即可。你可以前往[官方网站](http://jquery.com/" \t "http://www.365mini.com/page/_blank)下载jQuery库的js文件。

这里有两个版本的 jQuery 可供下载：

* Production version - 用于实际的网站中，已被精简和压缩。
* Development version - 用于测试和开发（未压缩，是可读的代码）

以上两个版本都可以从 jquery.com 中下载。



### 引入jQuery

，你也可以直接如下在HTML页面中引入该文件的URI。

jQuery 库是一个 JavaScript 文件，您可以使用 HTML 的 <script> 标签引用它：

<head>

<script src="jquery-1.4.2.min.js"></script>

</head>

*补充：在<script>标签中可以省略type="text/javascript"，因为在HTML5中，JavaScript 是 HTML5 以及所有现代浏览器中的默认脚本语言！*

# jQuery基础语法

## jQuery基础语法

### 文档就绪事件

如果希望代码在html档加载完之后立即执行，那么可以将代码放置在一个 document ready 函数中：

$(document).ready(function(){  
 // 在这里编写我们希望在DOM准备就绪后执行的代码...  
 });

将代码放在document ready 函数中是为了防止文档在完全加载（就绪）之前运行 jQuery 代码。起作用相当于：

window.onload=function(){ //js代码 }

* 如果在文档没有完全加载之前就运行函数，操作可能失败。

**提示：**简洁写法（与以上写法效果相同）:

$(function(){  
 // 开始写 jQuery 代码...  
 });

以上两种方式你可以选择你喜欢的方式实现文档就绪后执行jQuery方法。

### jQuery的运行原理

这里我们仅简单地介绍一下jQuery的运行原理。在jQuery库中实际上定义了一个jQuery()方法，它是jQuery库的核心。我们调用该方法并传入指定的参数，就可以返回一个jQuery实例对象，该对象中包含匹配的一个或多个DOM元素。接着，我们就可以使用jQuery对象上的方法来操作它所匹配的DOM元素。

jQuery对象上提供的方法足够我们进行几乎所有的DOM操作。

为了尽可能地减少代码量，jQuery库还为函数jQuery定义了一个别名变量$，它们是完全相同的(jQuery === $)，因此我们也可以使用$来替代jQuery。

基础语法：**$(selector).action()**

* 美元符号定义 jQuery
* 选择符（selector）"查询"和"查找" HTML 元素
* jQuery 的 action() 执行对元素的操作

实例:

$(this).hide() - 隐藏当前元素

$("p").hide() - 隐藏所有段落

$("p.test").hide() - 隐藏所有 class="test" 的段落

$("#test").hide() - 隐藏所有 id="test" 的元素

*注意:既然获得的是jQuery对象，你就只能使用jQuery对象的方法，而不能在jQuery对象上调用DOM元素(Element对象)自身的方法(比如getElementById())，除非你已经通过某些方式将它转换成了DOM元素。*

### DOM对象和jQuery对象互相转化

* **dom对象转jQuery对象:**

格式：

var $jquery对象 = $(dom对象);

示例：

var dom = document.getElementById("username");//dom对象

alert(dom.value);//获取用户名

var $jquery = $(dom);//dom对象转jquery对象

alert($jquery.val());//获取用户名

* **jQuery对象转dom对象**

格式：

var dom对象 = $jquery对象.get(0);

或

var dom对象 = $jquery对象[0];

示例

var $jquery = $("#username");

alert($jquery.val());//获取用户名

var dom1 = $jquery.get(0);

alert(dom1.value);//获取用户名

var dom2 = $jquery[0];

alert(dom2.value);//获取用户名

# jQuery核心：选择器

## jQuery选择器

### 选择器简介

在JS原生DOM中，我们想要对DOM元素进行操作，首先得获取到对应的元素(getElementById()、 getElementsByTagName()等)，然后再对这些元素进行操作。

同样的，jQuery也需要先选取所需的DOM元素，然后再针对这些元素进行操作。我们先来看看jQuery如何获取所需的元素。

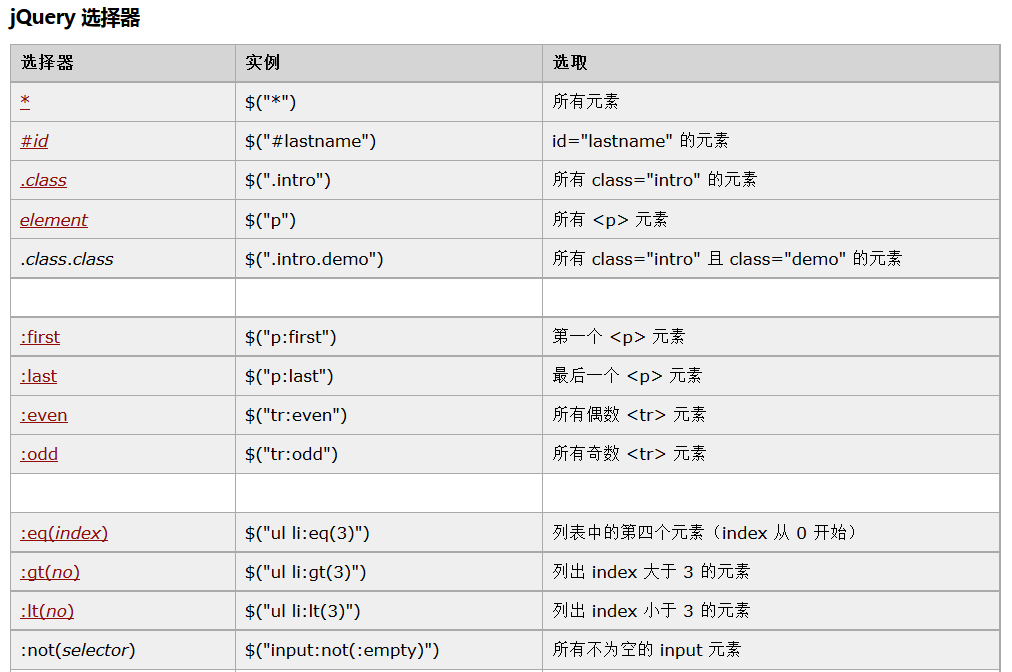
在jQuery中，我们一般通过一个字符串来标识匹配的元素，例如：

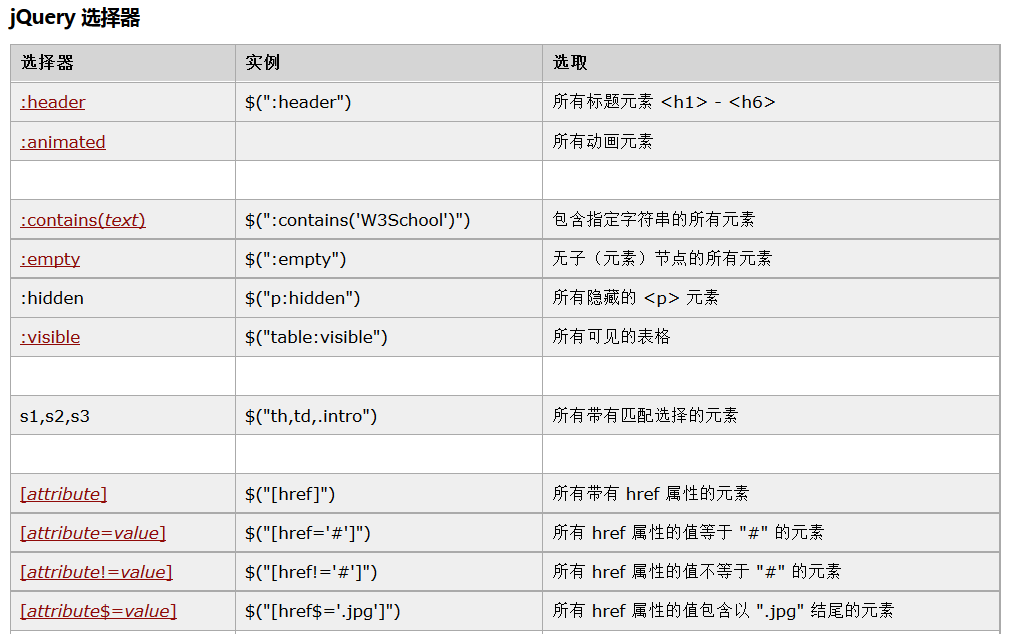
$("#uid"); // 选取id属性为"uid"的单个元素  
 $("p"); // 选取所有的p元素  
 $(".test"); // 选择所有带有CSS类名"test"的元素  
 $("[name=books]"); // 选择所有name属性为"books"的元素

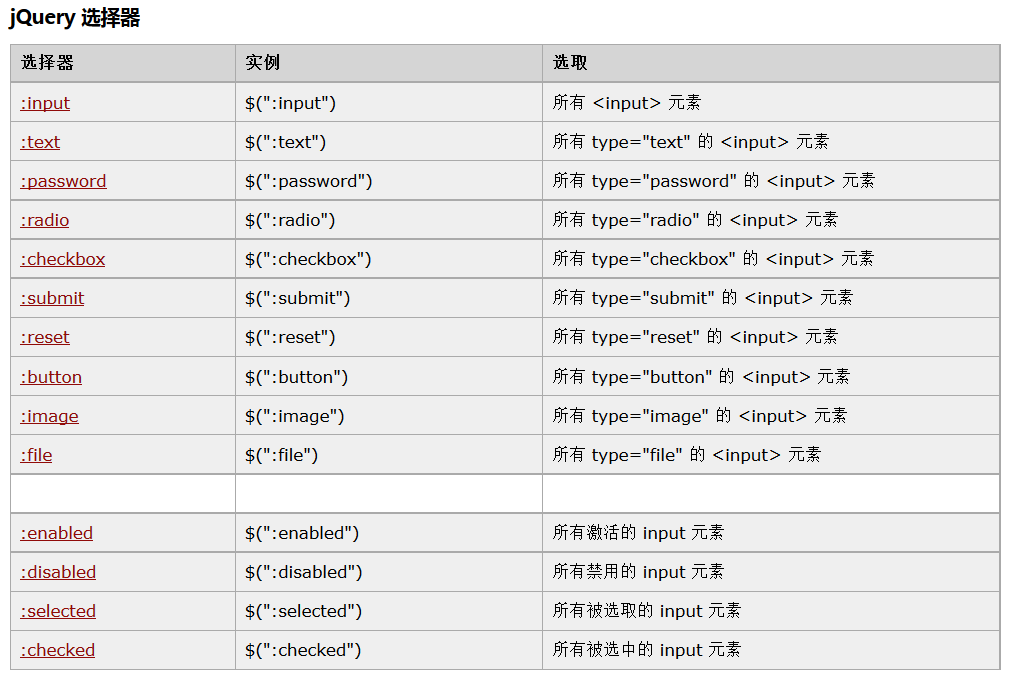
是否感觉这和CSS选择器非常相似？是的，jQuery获取元素就是通过类似于CSS选择器的字符串来匹配对应的元素，我们一般将其称作jQuery选择器(selector)。几乎所有的CSS选择器都可以当做jQuery选择器来使用，只要CSS选择器对哪些元素生效，对应的jQuery选择器就可以选取到哪些元素。

和CSS选择器一样，jQuery选择器也支持多个选择器任意组合使用。

### 选择器一览表







## 基本选择器

### 元素名选择器

选取所有的 <p> 元素;

$("p")

### .class选择器

选取所有 class="c1" 的 <p> 元素。

$("p.c1")

如果不限定元素名，选取所有的class="c1" 的元素，可以改为

$(".c1")。

### #id选择器

选取所有 id="d1" 的 <p> 元素。

$("p#d1")

如果不限定元素名，可以改为

$("#d1")。

### \*选择器

用法: $(\*)

返回值: 匹配所有标签, 封装成jQuery对象

### selector1,selector2,…

用法: $(”div,span,.myClass”)

返回值: 所有匹配选择器的标签, 封装成jQuery对象

## 层级选择器

如果想通过DOM元素之间的层次关系来获取特定元素。例如子元素、兄弟元素等。则需要通过层次选择器。

### ancestor descendant

用法: $("form input")

说明: 在给定的祖先元素下匹配所有后代元素

### parent > child

用法: $("form > input")

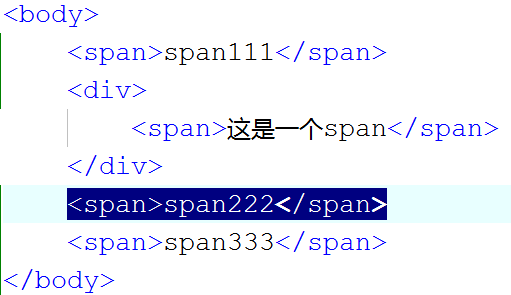
说明: 在指定父元素下匹配所有子元素。注意：要区分好后代元素与子元素

### prev + next

用法: $("div+span")

说明: 匹配所有紧接在div元素后的span元素，如：

$(“div + span”)

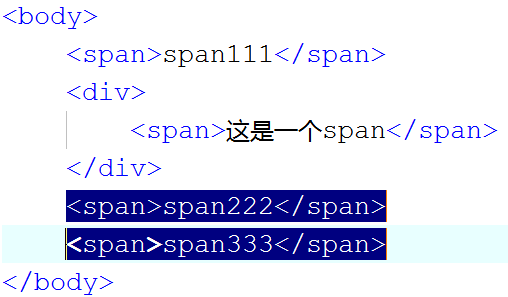


### prev ~ siblings

用法: $("form ~ input")

说明: 匹配form 元素之后的所有 input元素。如：

$(“div ~ span”)



## 基本过滤选择器

过滤选择器主要是通过特定的过滤规则来筛选出所需的DOM元素, 该选择器都以 ":" 开头

按照不同的过滤规则, 过滤选择器可以分为基本过滤, 内容过滤, 可见性过滤, 属性过滤, 子元素过滤, 表单过滤和表单属性过滤选择器.

### :first

用法: $("tr:first") ;

说明: 匹配找到的第一个元素

### :last

用法: $("tr:last")

说明: 匹配找到的最后一个元素.与 :first 相对应

### :not(selector)

用法: $("input:not(:checked)")

说明: 去除所有与给定选择器匹配的元素.有点类似于"非",意思是没有被选中的input(当input的type="checkbox").

### :even

用法: $("tr:even")

说明: 匹配所有索引值为偶数的元素，从0开始计数.js的数组都是从0开始计数的.

### :odd

用法: $("tr:odd")

说明: 匹配所有索引值为奇数的元素,和:even对应,从 0 开始计数.

### :eq(index)

用法: $("tr:eq(0)")

说明: 匹配一个给定索引值的元素.eq(0)就是获取第一个tr元素.括号里面的是索引值,不是元素排列数.

### :gt(index)

用法: $("tr:gt(0)")

说明: 匹配所有大于给定索引值的元素.

### :lt(index)

用法: $("tr:lt(2)")

说明: 匹配所有小于给定索引值的元素.

## 内容过滤选择器

内容过滤选择器的过滤规则主要体现在它所包含的子元素和文本内容上

### :contains(text)

用法: $("div:contains(’John’)")

说明: 匹配包含给定文本的元素.这个选择器比较有用，当我们要选择的不是dom标签元素时,它就派上了用场了,它的作用是查找被标签"围"起来的文本内容是否符合指定的内容的.

### :empty

用法: $("td:empty")

说明: 匹配所有不包含子元素或者文本的空元素

### :has(selector)

用法: $("div:has(p)").addClass("test")

说明: 匹配含有选择器所匹配的元素的元素。例如：给所有包含p元素的div标签加上class="test".

### :parent

用法: $("td:parent")

说明: 匹配含有子元素或者文本的元素.注意:这里是":parent",可不是".parent"哦!感觉与上面讲的":empty"形成反义词.

## 可见过滤选择器

根据元素的可见和不可见状态来选择相应的元素

### :hidden

用法: $("tr:hidden")

说明: 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.意思是css中display:none和input type="hidden"的都会被匹配到.同样,要在脑海中彻底分清楚冒号":", 点号"."和逗号","的区别.

### :visible

用法: $("tr:visible")

说明: 匹配所有的可见元素.

## 属性过滤选择器

属性过滤选择器的过滤规则是通过元素的属性来获取相应的元素

### [attribute]

用法: $("div[id]")

说明: 匹配包含给定属性的元素. 例子中是选取了所有带id属性的div标签.

### [attribute=value]

用法: $("input[name='newsletter']").attr("checked", true)

说明: 匹配给定的属性是某个特定值的元素.例子中选取了所有name属性是newsletter的 input 元素.

### [attribute!=value]

用法: $("input[name!='newsletter']").attr("checked", true)。

说明：匹配所有不含有指定的属性，或者属性不等于特定值的元素.此选择器等价于:not([attr=value]),要匹配含有特定属性但不等于特定值的元素,请使用[attr]:not([attr=value]).之前看到的 :not 派上了用场.

### [attribute^=value]

用法: $("input[name^=‘news’]")

说明: 匹配给定的属性是以某些值开始的元素.,我们又见到了这几个类似于正则匹配的符号.现在想忘都忘不掉了吧?!

### [attribute$=value]

用法: $("input[name$=‘letter’]")

说明: 匹配给定的属性是以某些值结尾的元素.

### [attribute\*=value]

用法: $("input[name\*=’man’]")

说明: 匹配给定的属性是以包含某些值的元素.

## 子元素过滤选择器

### :nth-child(index/even/odd/equation)

用法: $("ul li:nth-child(2)")

说明: 匹配其父元素下的第N个子或奇偶元素.这个选择器和之前说的[基础过滤(Basic Filters)](http://flex.joelove.cn/jquery-selectors-basicfilters/)中的 eq() 有些类似,不同的地方就是前者是从0开始,后者是从1开始.

### :first-child

用法: $("ul li:first-child")

说明: 匹配第一个子元素.’:first’ 只匹配一个元素,而此选择符将为每个父元素匹配一个子元素.这里需要特别点的记忆下区别.

### :last-child

用法: $("ul li:last-child")

说明: 匹配最后一个子元素.’:last’只匹配一个元素,而此选择符将为每个父 元素匹配一个子元素.

### :only-child

用法: $("ul li:only-child")

说明: 如果某个元素是父元素中唯一的子元素,那将会被匹配.如果父元素中含有其他元素,那将不会被匹配.意思就是:只有一个子元素的才会被匹配!

## 表单选择器

### :input

用法: $(":input")

说明:匹配所有 text, textarea, select 和 button 元素

### :text

用法: $(":text")

说明: 匹配所有的单行文本框.

### :password

用法: $(":password")

说明: 匹配所有密码框.

### :radio

用法: $(":radio")

说明: 匹配所有单选按钮.

### :checkbox

用法: $(":checkbox")

说明: 匹配所有复选框

### :submit

用法: $(":submit")

说明: 匹配所有提交按钮

### :image

用法: $(":image")

说明: 匹配所有图像域.

### :reset

用法: $(":reset")

说明: 匹配所有重置按钮.

### :button

用法: $(":button")

说明: 匹配所有按钮.这个包括直接写的元素button.

### :file

用法: $(":file")

说明: 匹配所有文件域.

### :hidden

用法: $("input:hidden")

说明: 匹配所有不可见元素，或者type为hidden的元素.这个选择器就不仅限于表单了,除了匹配input中的hidden外,那些style为hidden的也会被匹配.

## 表单对象属性过滤选择器

此选择器主要对所选择的表单元素进行过滤

### :enabled

用法: $("input:enabled")

说明: 匹配所有可用元素.意思是查找所有input中不带有disabled="disabled"的input.不为disabled,当然就为enabled啦.

### :disabled

用法: $("input:disabled")

说明: 匹配所有不可用元素.与上面的那个是相对应的.

### :checked

用法: $("input:checked")

说明: 匹配所有的单选框/复选框/select中的option被选中元素(复选框、单选框等).

### :selected

用法: $("select option:selected")

说明: 匹配所有选中的option元素.

## CSS 选择器

jQuery CSS 选择器可用于改变 HTML 元素的 CSS 属性。

下面的例子把所有 p 元素的背景颜色更改为红色：

实例

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

# jQuery文档操作(CRUD)

## 内部插入节点

### append(content)

向每个匹配的元素的内部的结尾处追加内容

### appendTo(content)

把所有匹配的元素追加到另一个指定的元素元素集合中

### prepend(content)

向每个匹配的元素的内部的开始处插入内容

### prependTo(content)

将每个匹配的元素插入到指定的元素内部的开始处

## 外部插入节点

### after(content) :

在每个匹配的元素之后插入内容

### before(content)

在每个匹配的元素之前插入内容

### insertAfter(content)

把所有匹配的元素插入到另一个、指定的元素元素集合的后面

### insertBefore(content)

把所有匹配的元素插入到另一个、指定的元素元素集合的前面

## 查找节点

### 使用jQuery选择器查询

$(selector)

得到一个包含所有匹配的dom节点对象的jQuery对象

### 查询jQuery对象内部数据

$object.find(selector)

在Jquery对象中根据selector查找其中匹配的后代节点

### 遍历jQuery对象包含的数据

$(selector1).each(function(index, itemDom){ })

遍历jQuery对象所包含的所有节点, 每取一个dom节点对象都去调用设置的回调函数, 并将取出的节点在数组中的下标和节点对象传入函数

## 创建节点

### 创建节点

$(htmlString)

动态创建的新元素节点不会被自动添加到文档中, 需要使用其他方法将其插入到文档中;

当创建单个元素时, 需注意闭合标签和使用标准的 XHTML 格式. 例如创建一个<p>元素, 可以使用 $(“<p/>”) 或 $(“<p></p>”), 但不能使用 $(“<p>”) 或 $(“</P>”)

创建文本节点就是在创建元素节点时直接把文本内容写出来; 创建属性节点也是在创建元素节点时一起创建。

## 删除节点

### empty():

删除匹配的元素集合中所有的子节点(不包括本身)。

### remove():

删除匹配的元素及其子元素(包括本身)

## 属性操作

### attr(name [,value])

根据属性名获取属性值或者设置一个属性

### removeAttr(name)

根据属性名删除对应的属性

## 操作HTML代码/值

### html([val])

得到元素的内容或者设置元素的内容

### val([value])

获得匹配元素的当前值或者设置其值

## 操作CSS样式

### addClass(className)

添加class属性

### removeClass()

移除class属性

### css(name,[value])

查看某个样式属性, 或设置某个样式属性

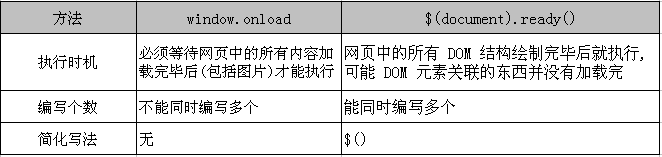
# 事件

## 常用的事件

### ready(fn)

当DOM载入就绪可以查询及操纵时绑定一个要执行的函数

它与window.onload是有区别的



### click([fn])

触发每一个匹配元素的click事件

### blur([fn])

触发每一个匹配元素的blur事件

### focus([fn])

触发每一个匹配元素的fucus事件

### change([fn])

触发每一个匹配元素的change事件

## 绑定与解绑事件

### bind(type, fn)

为每个匹配元素的特定事件绑定事件处理函数。

### unbind(type)

bind()的反向操作，从每一个匹配的元素中删除绑定的事件

## 事件切换

### hover(over,out)

当鼠标移动到一个匹配的元素上面时，会触发指定的第一个函数。当鼠标移出这个元素时，会触发指定的第二个函数。

# 效果

## 基本

### show()

显示当前标签

### hide()

隐藏当前标签

### toggle()

切换当前标签的可见性

## 滑动

### slideDown()

这个动画效果只调整元素的高度，可以使匹配的元素以“滑动”的方式显示出来

### slideUp()

这个动画效果只调整元素的高度，可以使匹配的元素以“滑动”的方式隐藏起来

### slideToggle()

通过高度变化来切换所有匹配元素的可见性

## 淡入淡出

### fadeIn()

通过不透明度的变化来实现所有匹配元素的淡入效果

### fadeOut()

通过不透明度的变化来实现所有匹配元素的淡出效果

### fadeToggle()

通过不透明度的变化来开关所有匹配元素的淡入和淡出效果

*注意: 低版本jQuery未提供.*

# jQuery之Ajax应用

## Ajax介绍

### 什么是Ajax？

AJAX（Asynchronous Javascript And XML）翻译成中文就是“异步Javascript和XML”。即使用Javascript语言与服务器进行异步交互，传输的数据为XML（当然，现在传输的数据不仅仅是XML）。

传统的网页（不使用 AJAX）如果需要更新内容，必需重载整个网页面。

而在AJAX中，通过在后台与服务器进行少量数据交换，可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。这一特点给用户的感受是在不知不觉中完成请求和响应过程。

总结:

* 与服务器异步交互；
* 浏览器页面局部刷新；

Jquery对ajax操作进行了封装。在jquery中提供了完整的ajax兼容套件，其中最底层的方法是$.ajax(),第二层是 load()，$.get() 和 $.post(), 第三层是 $.getScript() 和 $.getJSON()。

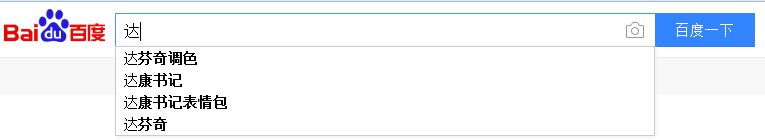
### 同步交互与异步交互

同步交互：客户端发出一个请求后，需要等待服务器响应结束后，才能发出第二个请求；

异步交互：客户端发出一个请求后，无需等待服务器响应结束，就可以发出第二个请求。

## AJAX常见应用情景

### 百度搜索框候选词提示



当我们在百度中输入一个“达”字后，会马上出现一个下拉列表！列表中显示的是包含“达”字的若干个关键字。

其实这里就使用了AJAX技术！当文件框发生了输入变化时，浏览器会使用AJAX技术向服务器发送一个请求，查询包含“达”字的前几个关键字，然后服务器会把查询到的结果响应给浏览器，最后浏览器把这几个关键字显示在下拉列表中。

* 在请求发出后，浏览器不用等待服务器响应结果就可以进行其他操作；
* 整个过程中页面没有刷新，只是局部刷新了！

### 用户名是否存在校验



当输入用户名后，把光标移动到其他表单项上时，浏览器会使用AJAX技术向服务器发出请求，服务器会查询名为”韩少云”的用户是否存在，最终服务器返回的数据表示名为”韩少云”的用户已经存在了，浏览器在得到结果后显示“用户名已存在”。

* 在请求发出后，浏览器不用等待服务器响应结果就可以进行其他操作；
* 整个过程中页面没有刷新，只是局部刷新了；

## AJAX的优缺点

### 优点：

* AJAX使用Javascript技术向服务器发送异步请求；
* AJAX无须刷新整个页面；
* 因为服务器响应内容不再是整个页面，而是页面中的局部，所以AJAX性能高；

### 缺点：

* AJAX并不适合所有场景，很多时候还是要使用同步交互；
* AJAX虽然提高了用户体验，但无形中向服务器发送的请求次数增多了，导致服务器压力增大；
* 因为AJAX是在浏览器中使用Javascript技术完成的，所以还需要处理浏览器兼容性问题；

## jQuery实现Ajax

### jQuery之Ajax实现—ajax方法

$.ajax(url, [data], [async], [callback]);

url -- 发送请求的URL地址

data -- 可选, 发送至服务器的key/value数据

async -- 可选, 默认为true, 表示异步交互

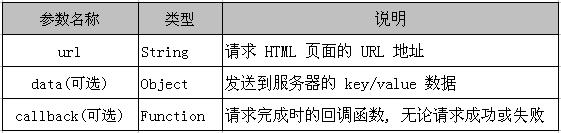
type -- 请求方式 , 默认为"GET"。

success -- 可选, 请求成功后执行的函数, 函数参数:

result -- 服务器返回的数据

### jQuery之Ajax实现—load方法

load()方法是 jQuery 中最为简单和常用的 Ajax 方法, 能载入远程的 HTML 代码并插入到 DOM 中. 它的结构是: load(url[, data][,callback])。



$(selector).load(url,data,callback);

selector -- 选择器, 将从服务器获取到的数据加载到指定的元素中

url -- 发送请求的URL地址

data -- 可选, 向服务器发送的数据 key/value数据

callback -- 可选, load方法完成后所执行的函数

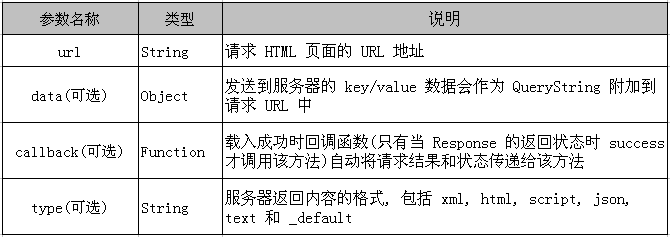
实例：

response.write("This is some text from an external ASP file.")

### jQuery之Ajax实现—get和post方法

$.get() 和$.post()是分别使用get提交方式和POST提交方式进行异步请求，用法类似，所以这里只列一种，即get方式。

$.get() 方法使用 GET 方式来进行异步请求. 它的结构是: $.get(url[, data][, callback][, type]);



$.get(url, [data], [callback]);

url -- 发送请求的URL地址

data -- 可选, 向服务器发送的数据

callback -- 可选, 请求成功后所执行的函数

type-- 可选, 服务器响应结果的数据格式

* $.get() 方法的回调函数只有两个参数: data 代表返回的内容, 可以是 XML 文档, JSON 文件, HTML 片段等; textstatus 代表请求状态, 其值可能为: succuss, error, notmodify, timeout 4 种.
* 方法的返回值：XMLHttpRequest对象

# JSON

## JSON介绍

### 什么是JSON

JSON(JavaScript Object Notation) 是一种JS提供的轻量级的数据交换格式。

JSON是用字符串来表示Javascript对象，例如可以在Servlet中发送一个JSON格式的字符串给客户端JavaScript，JavaScript可以执行这个字符串，得到一个Javascript对象。

### JSON与XML比较

* 可读性：XML比JSON具有更好的可读性；
* 解析难度：JSON本身就是JS对象，所以解析、操作非常简单；
* 流行度：XML已经流行好多年，但在某些领域，特别是在ajax中，JSON更受欢迎。

### JSON对象语法

JSON 语法：

* 数据在名称/值对中
* 数据由逗号分隔
* 花括号保存对象
* 方括号保存数组

var person = {"name":"zhangSan", "age":"18", "sex":"male"};

alert(person.name + ", " + person.age + ", " + person.sex);

　　注意，key也要在双引号中！

JSON值：

* [数字](http://baike.baidu.com/view/37407.htm" \t "_blank)（整数或浮点数）
* 字符串（在双引号中）
* 逻辑值（true 或 false）
* 数组（在方括号中）
* 对象（在花括号中）
* null

|  |
| --- |
| var person = {"name":"zhangSan", "age":"18", "sex":"male", "hobby":["cf", "sj", "ddm"]};  alert(person.name + ", " + person.age + ", " + person.sex + ", " + person.hobby); |

带有方法的JSON对象：

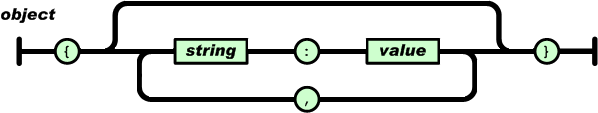
|  |
| --- |
| var person = {"name":"zhangSan", "getName":function(){return this.name;}};  alert(person.name);  alert(person.getName()); |

### JSON格式

JSON具有以下这些形式：

**形式一:**

对象是一个无序的“‘名称/值’对”集合。一个对象以“{”（左括号）开始，“}”（右括号）结束。每个“名称”后跟一个“:”（冒号）；“‘名称/值’ 对”之间使用“,”（逗号）分隔。



示例:

var person = {

"name":"张三",

"age": 18,

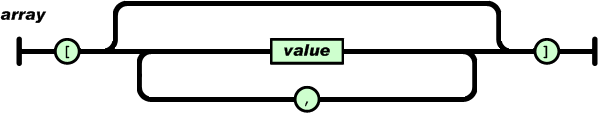
"gender": "男"

}

**形式二:**

数组是值（value）的有序集合。一个数组以“[”（左中括号）开始，“]”（右中括号）结束。值之间使用“,”（逗号）分隔。

{“key1”:value1,’key2’:value2,key3:value3}串



示例:

var person = {

"name":"张三",

"age": 18,

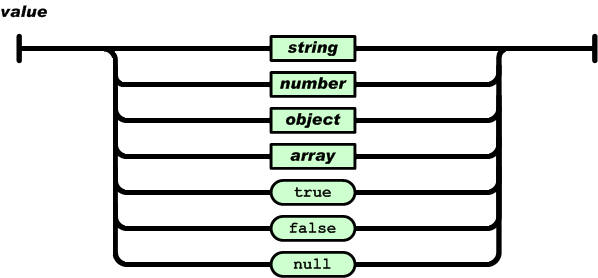
"friends":["小明", "小红", "小刘"]

}

**形式三:**

值（value）可以是双引号括起来的字符串（string）、数值(number)、true、false、 null、对象（object）或者数组（array）。这些结构可以嵌套。

[“value1”,”value2”,”value3”]



示例:

var person = {

"name":"张三",

"age": 18,

"friends":[

{

"name":"小明",

"age": 17,

"friends":["小李", "小兰"]

},

{

"name":"小刘",

"age": 19,

"friends":["小陈", "小红"]

}

]

}